Machine Learning for Risk Prediction using Oblique Random Survival Forests

A demo using the aorsf package

Byron C. Jaeger

April 20, 2023

Hello!

Slides are here: https://www.byronjaeger.com/talk/

Outline

Background and Jargon

- Machine Learning & Supervised Learning
- Decision Trees & Random Forests: Axis-based & Oblique
- Censoring & Random Survival Forests
- Oblique Random Survival Forests

Demo with `aorsf`

- Fit
- Interpret
- Benchmark
- Extend

Machine Learning

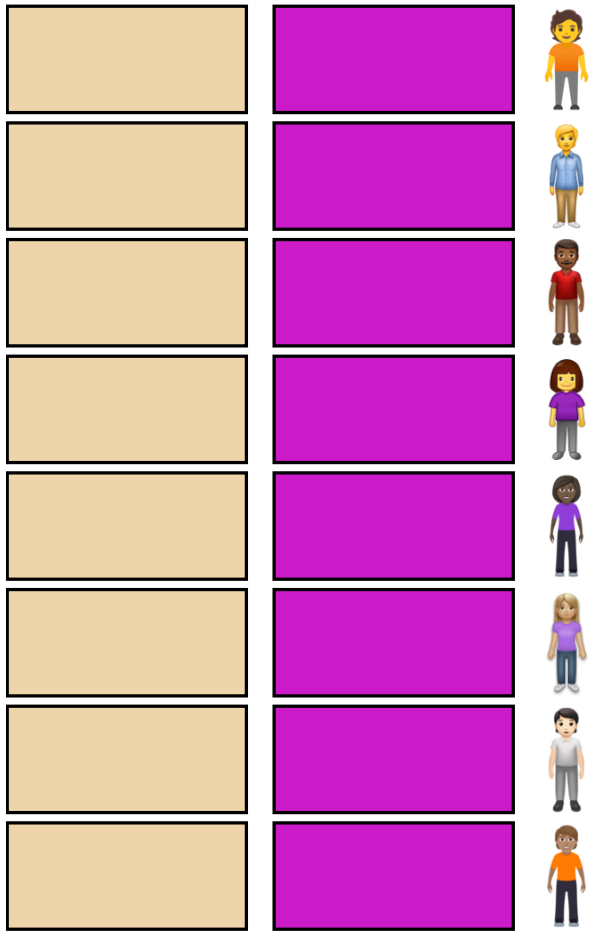
Supervised learning

- Labeled data
- Predict an outcome
- Learners
- Risk prediction

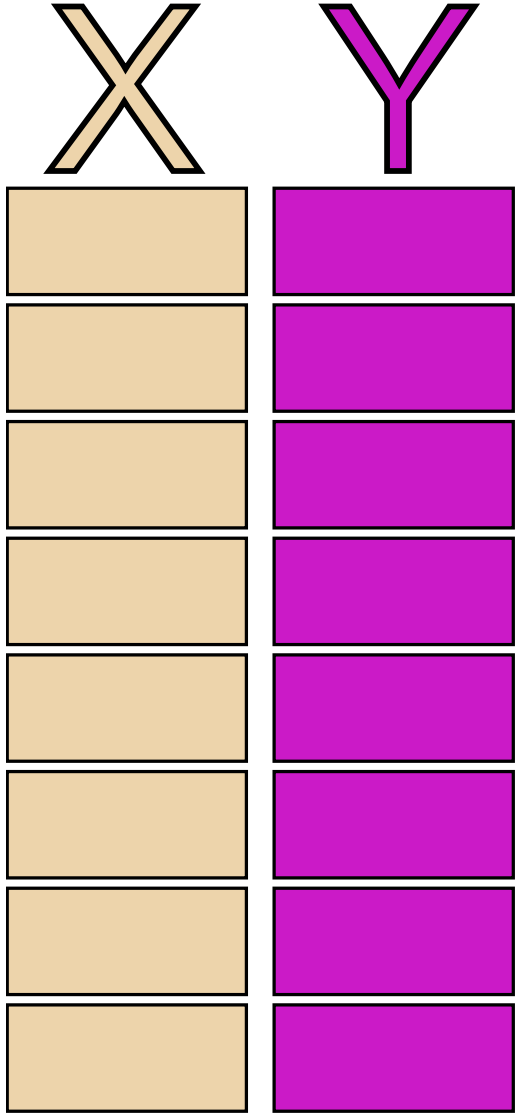
Unsupervised learning

- Unlabeled data
- Find structure
- Clusters
- Organize medical records

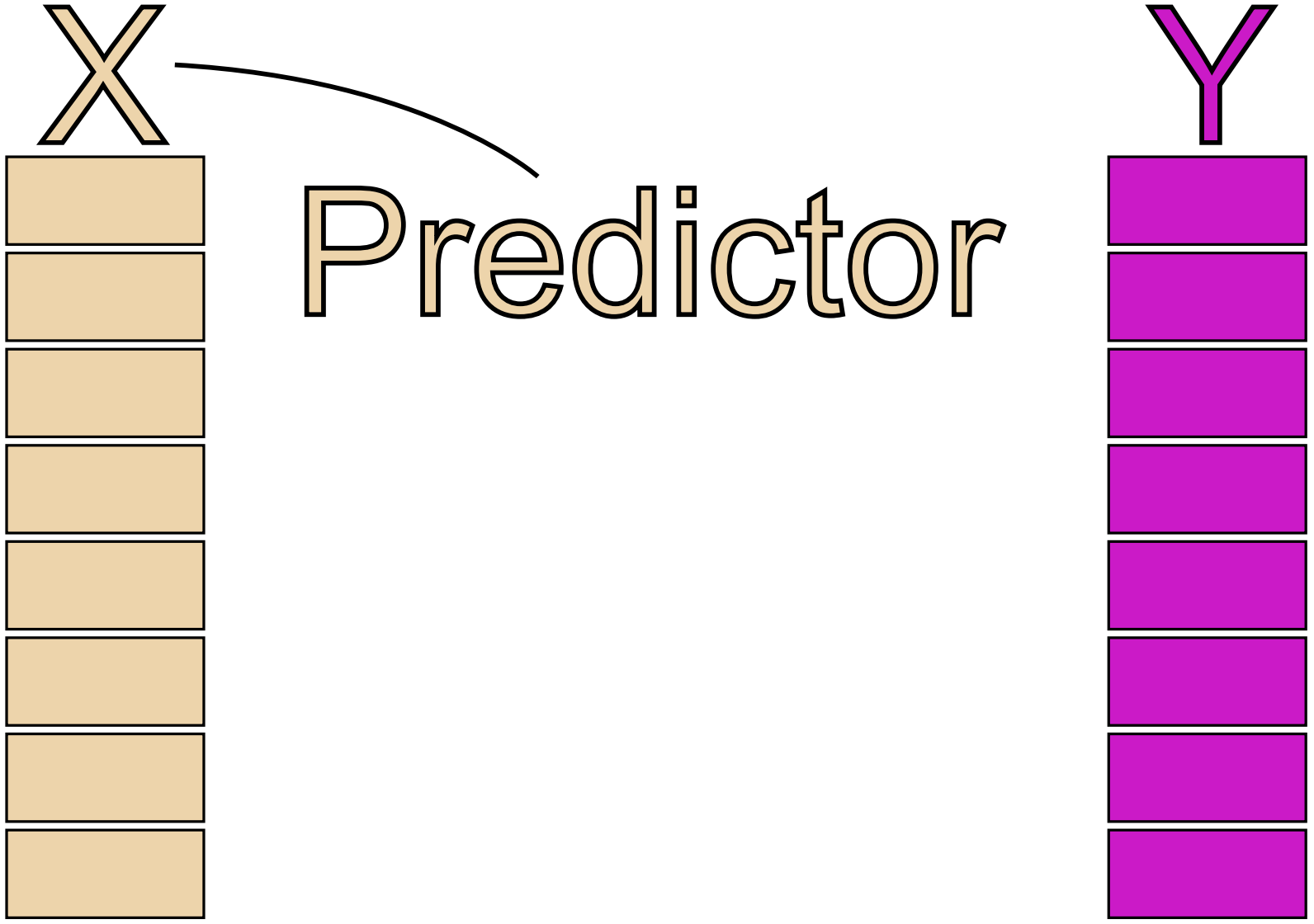
Supervised learning

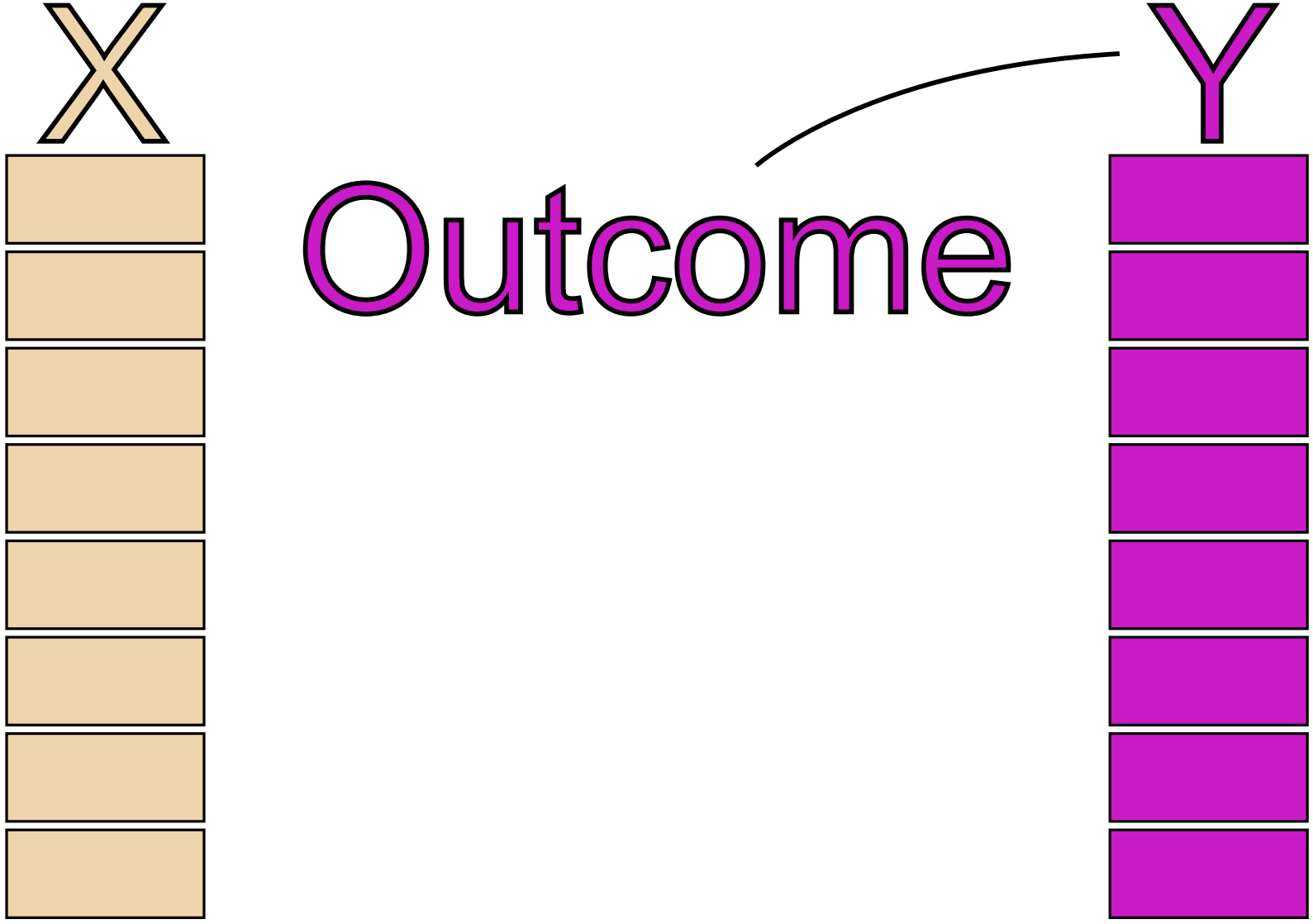


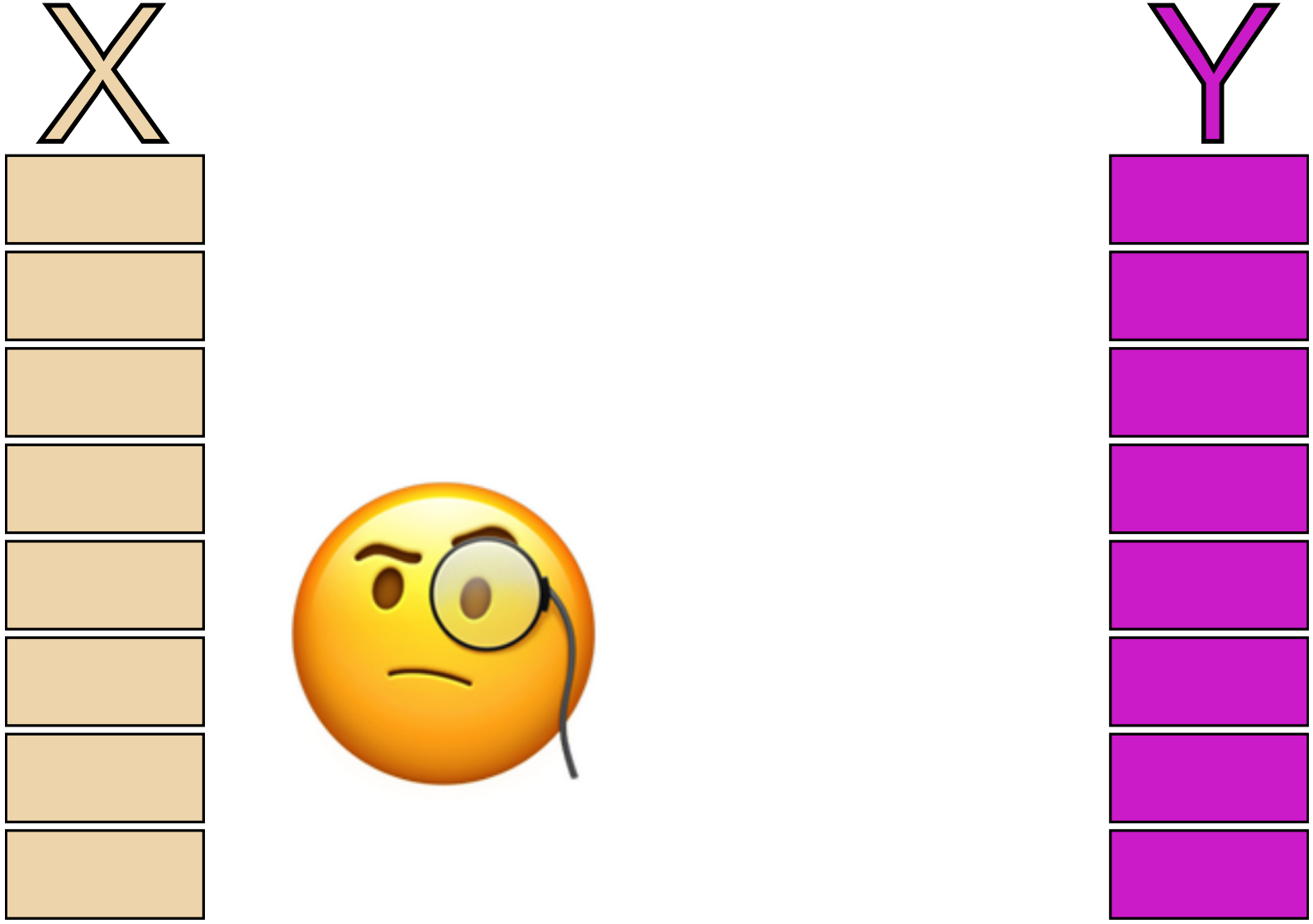
Each row is data from one person

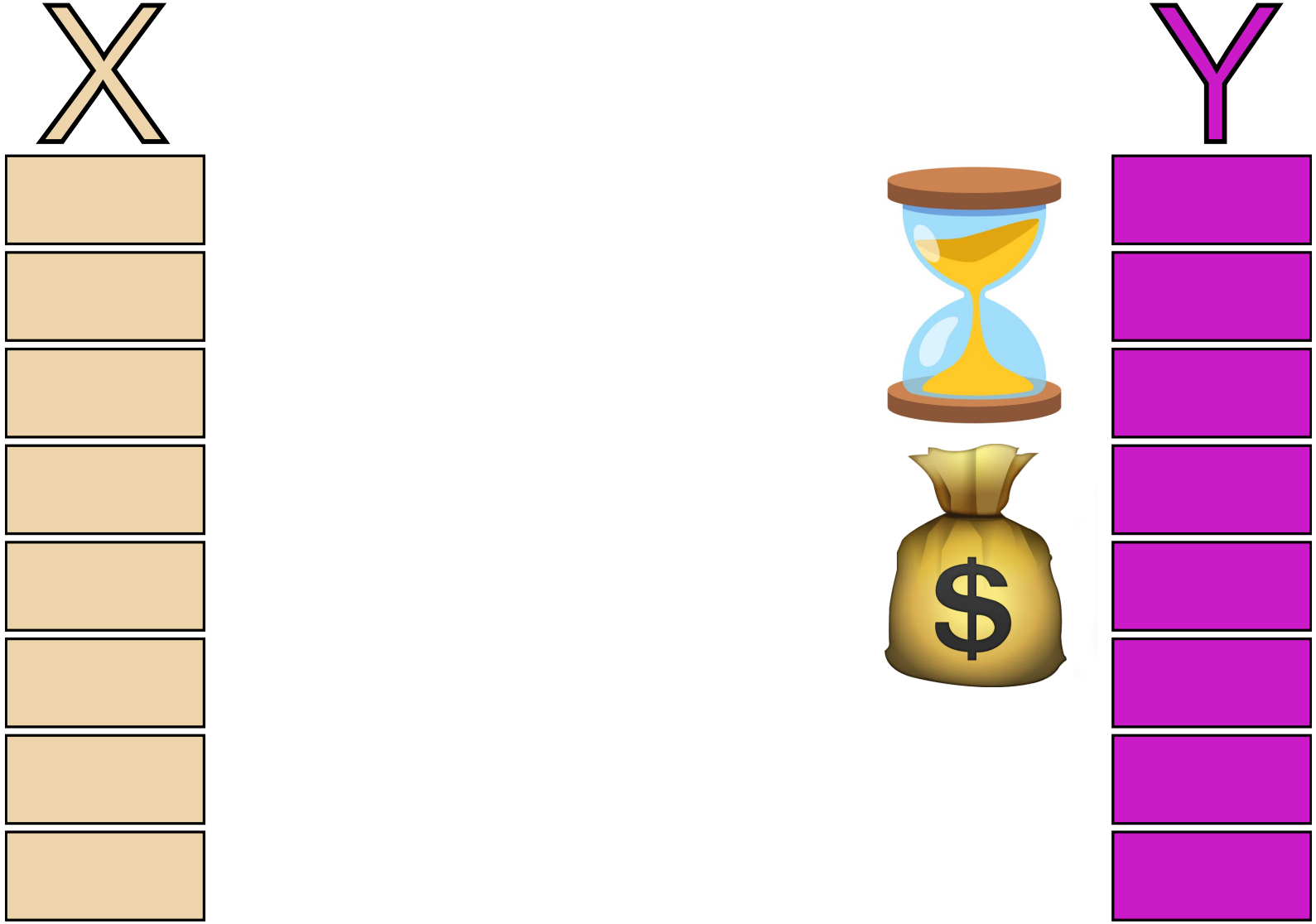


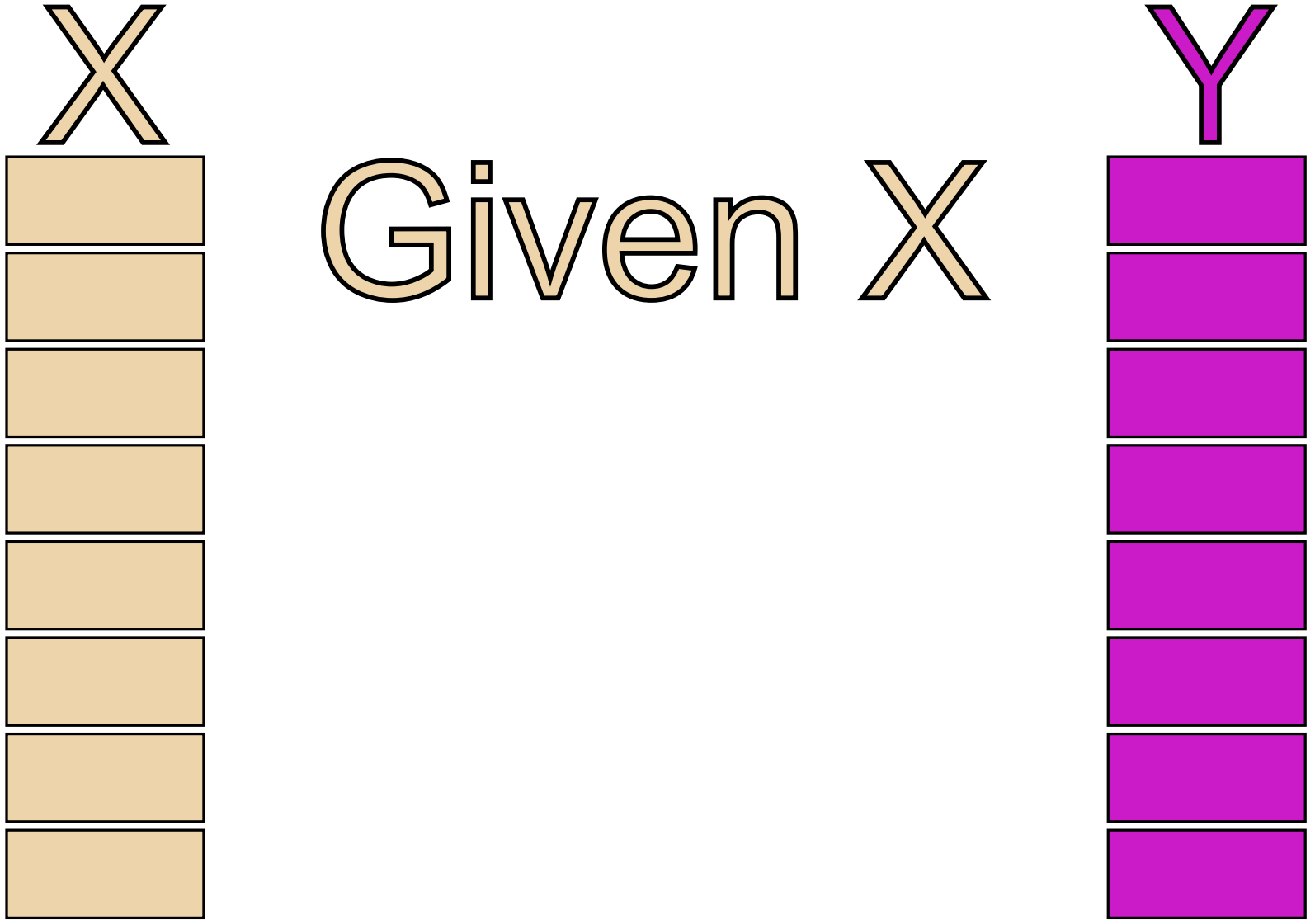
Each column is a variable


X
Predictor
Y


X
Outcome
Y

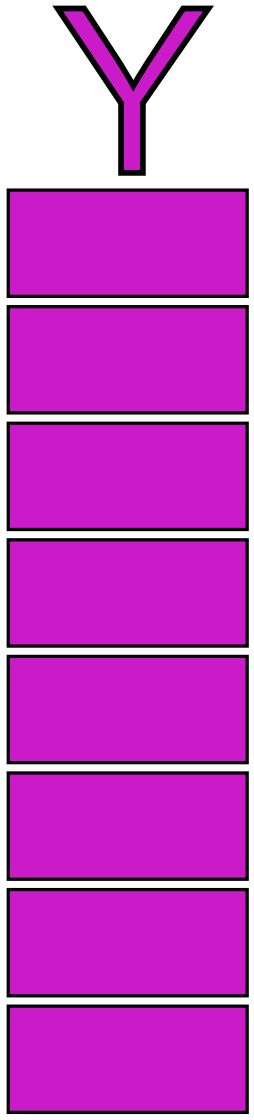

X
Y


X
Y
$


X
Given X
Y

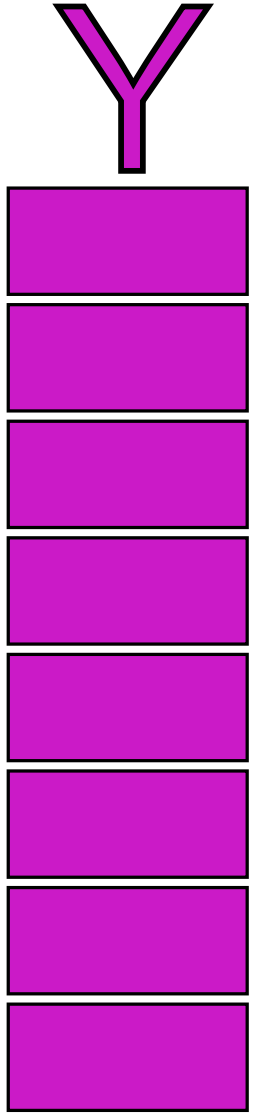
Given X

Predict Y



Learner

$$f\ (x) = y$$

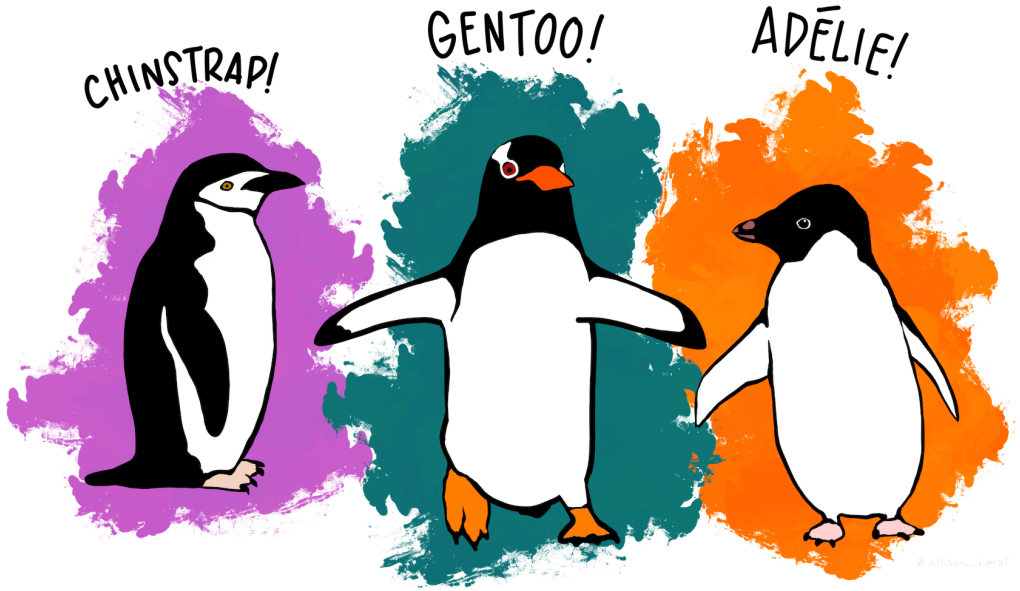


Decision Trees and Random Forests

Decision trees

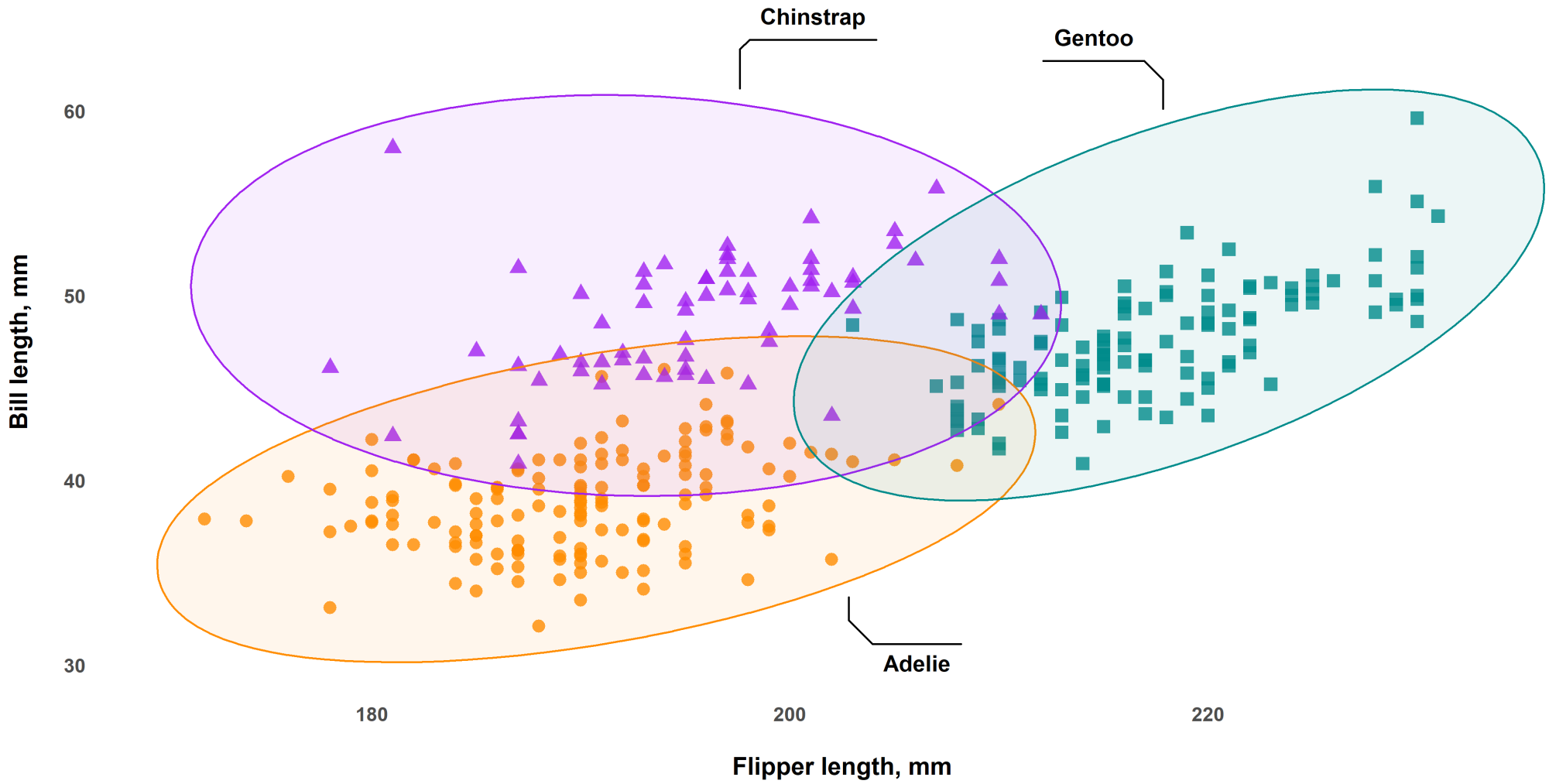
- Partition the space of predictor variables.
- Used in classification, regression, and survival analysis.
- May be **axis-based** or **oblique** (more on this soon)

We'll demonstrate the mechanics of decision trees by developing a prediction rule to classify penguin[1] species (chinstrap, gentoo, or adelie) based on bill depth and bill length.

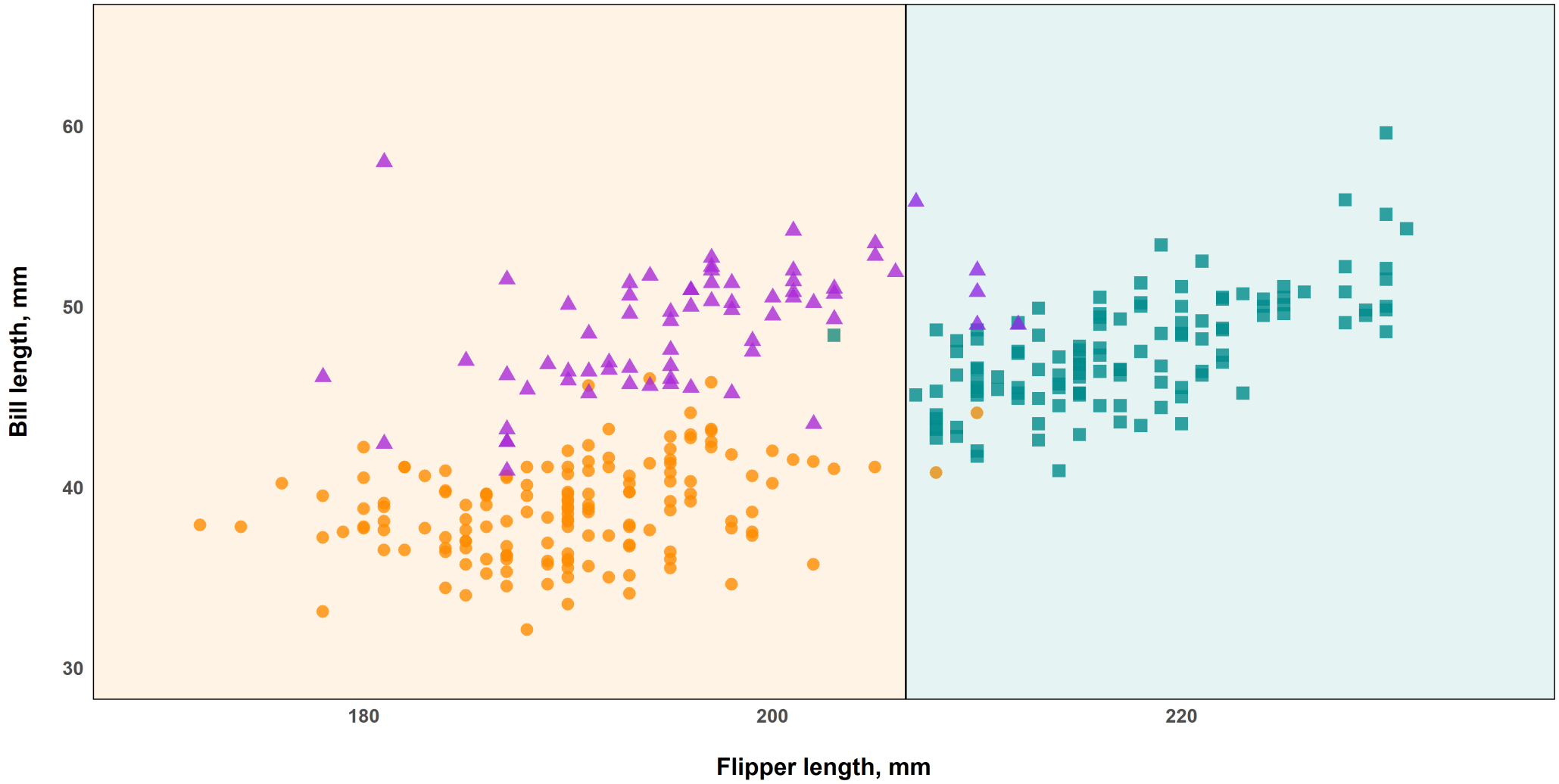


[1]Data were collected and made available by Dr. Kristen Gorman and the Palmer Station, a member of the Long Term Ecological Research Network.

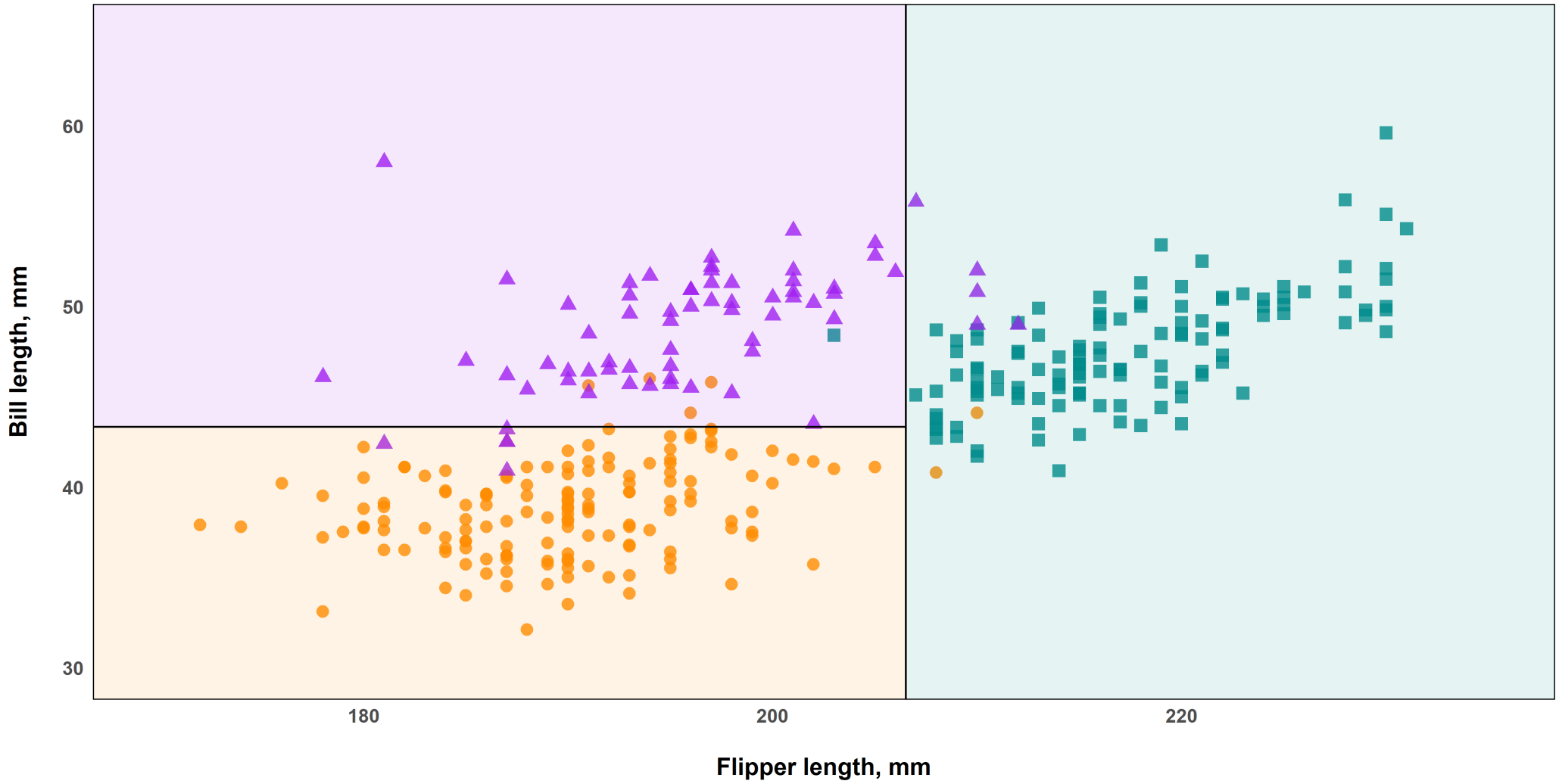
Dimensions for Adelie, Chinstrap and Gentoo Penguins at Palmer Station



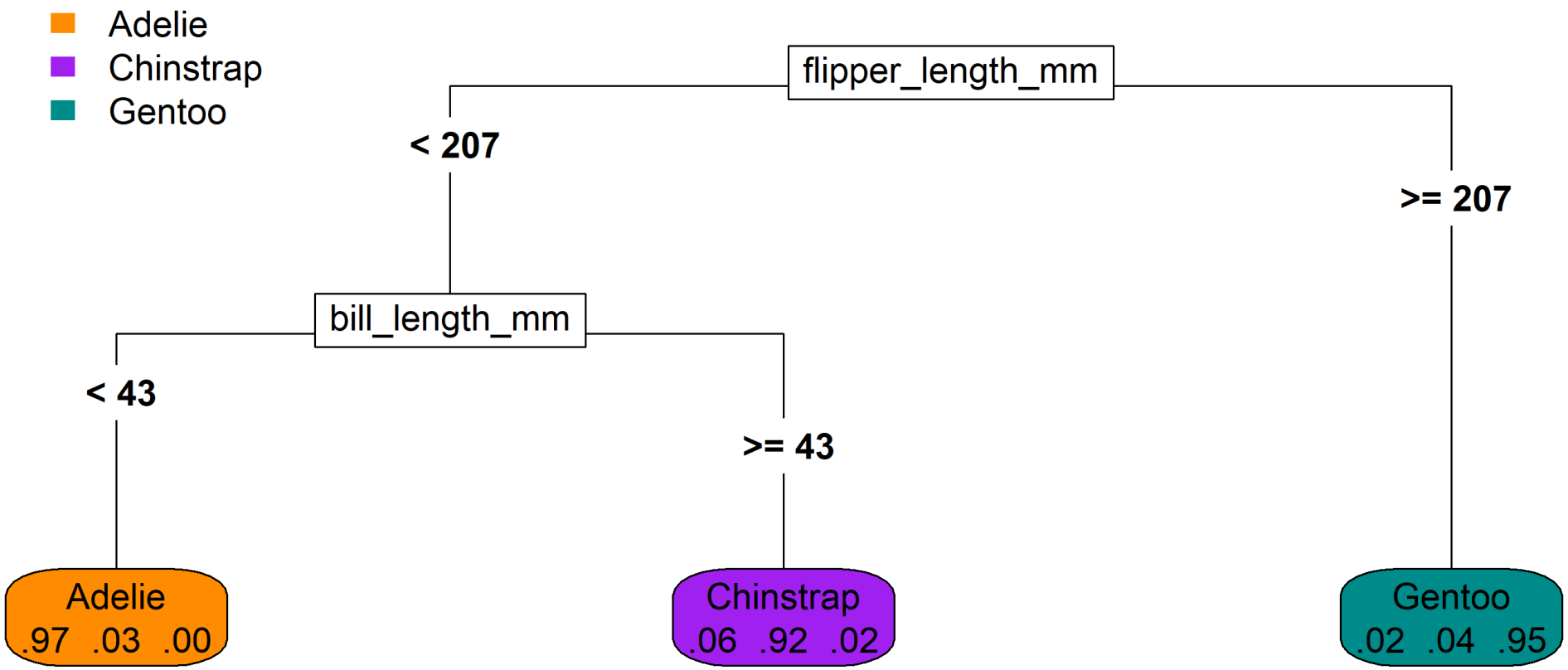
Partition all the penguins into flipper length < 207 or ≥ 207



Partition penguins on the left into bill length < 43 or ≥ 43



The same partitions visualized as a binary tree



Random Forest

Each tree is grown through a randomized process:

- tree-specific bootstrapped replicate of the training data
- node-specific random subsets of predictors

Randomness makes the trees more independent

- Each randomized tree is *individually weaker* than a standard decision tree.
- The *average prediction* from many randomized trees is more accurate than the prediction from a single tree.

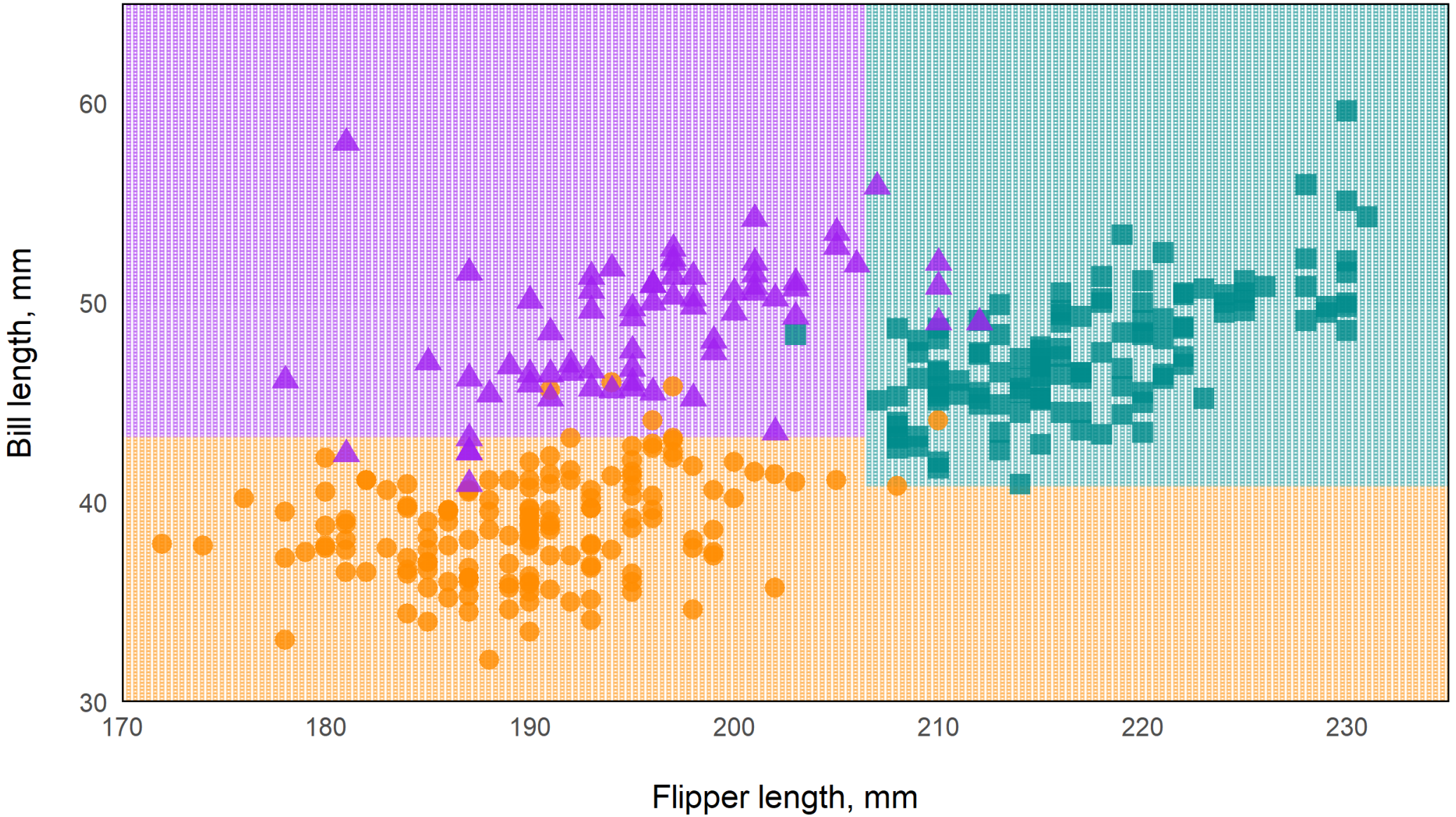
Why? Consider 100 classification trees, each with 55% chance of choosing the right answer, 45% chance of choosing wrong.

- if the trees are all the same, the ensemble has 45% chance of being wrong
- if they are completely independent and we use the majority vote:

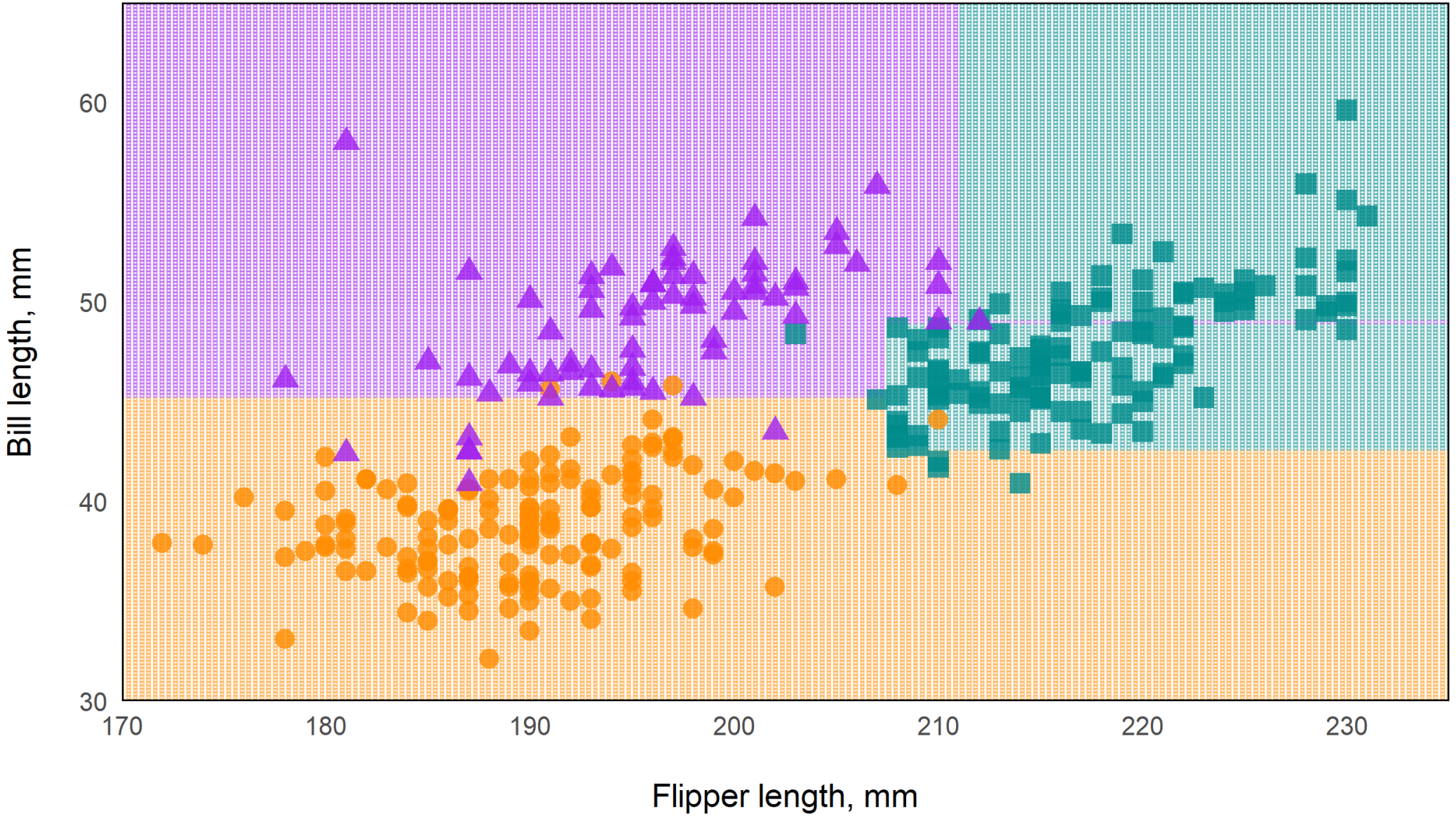
```
# chance of 49 or fewer trees being right
pbinom(q = 49, size = 100, prob = 0.55)
```

```
## [1] 0.1345762
```

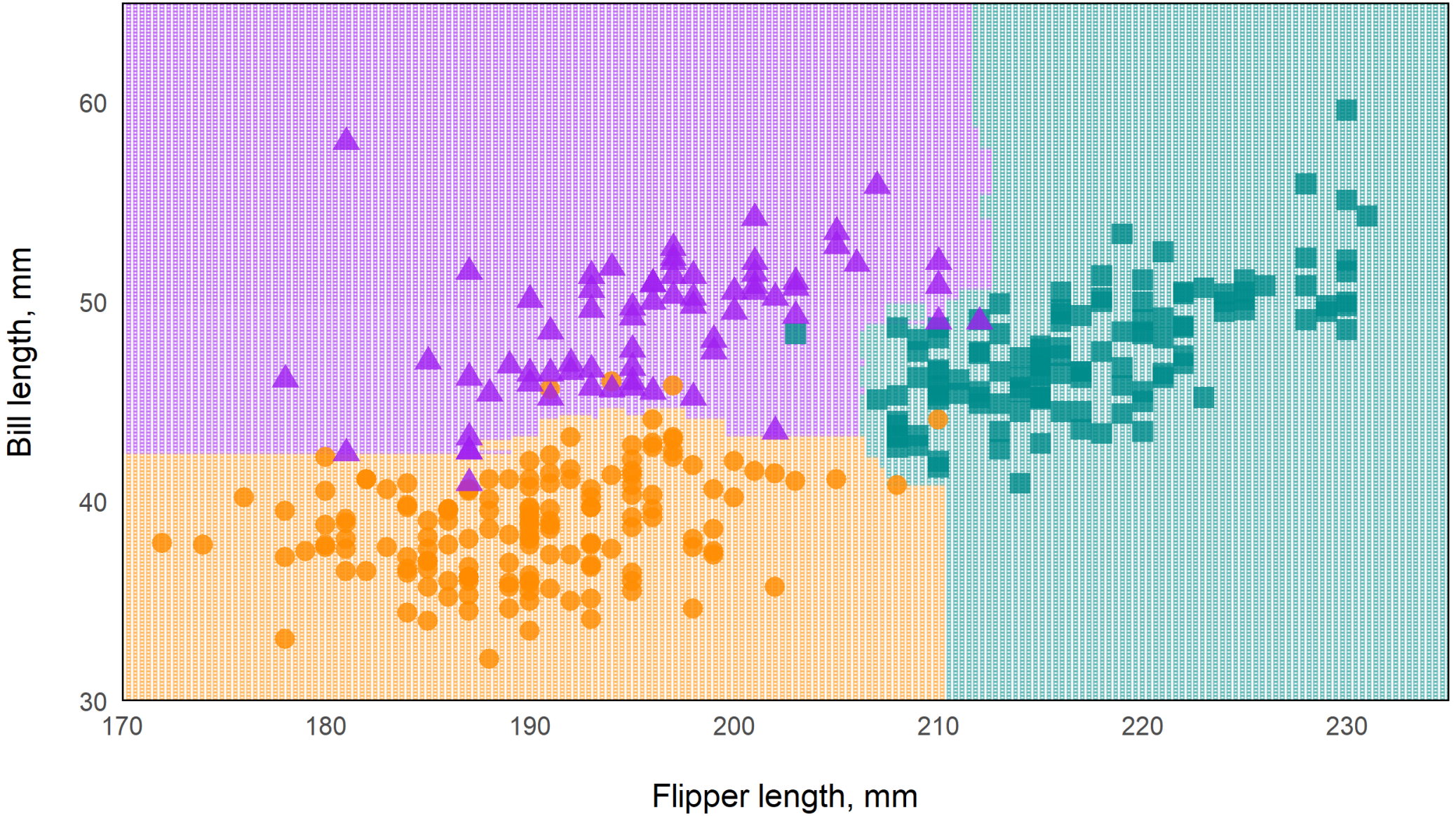
Predictions from a non-random, **individual tree**



Predictions from **randomized individual tree**



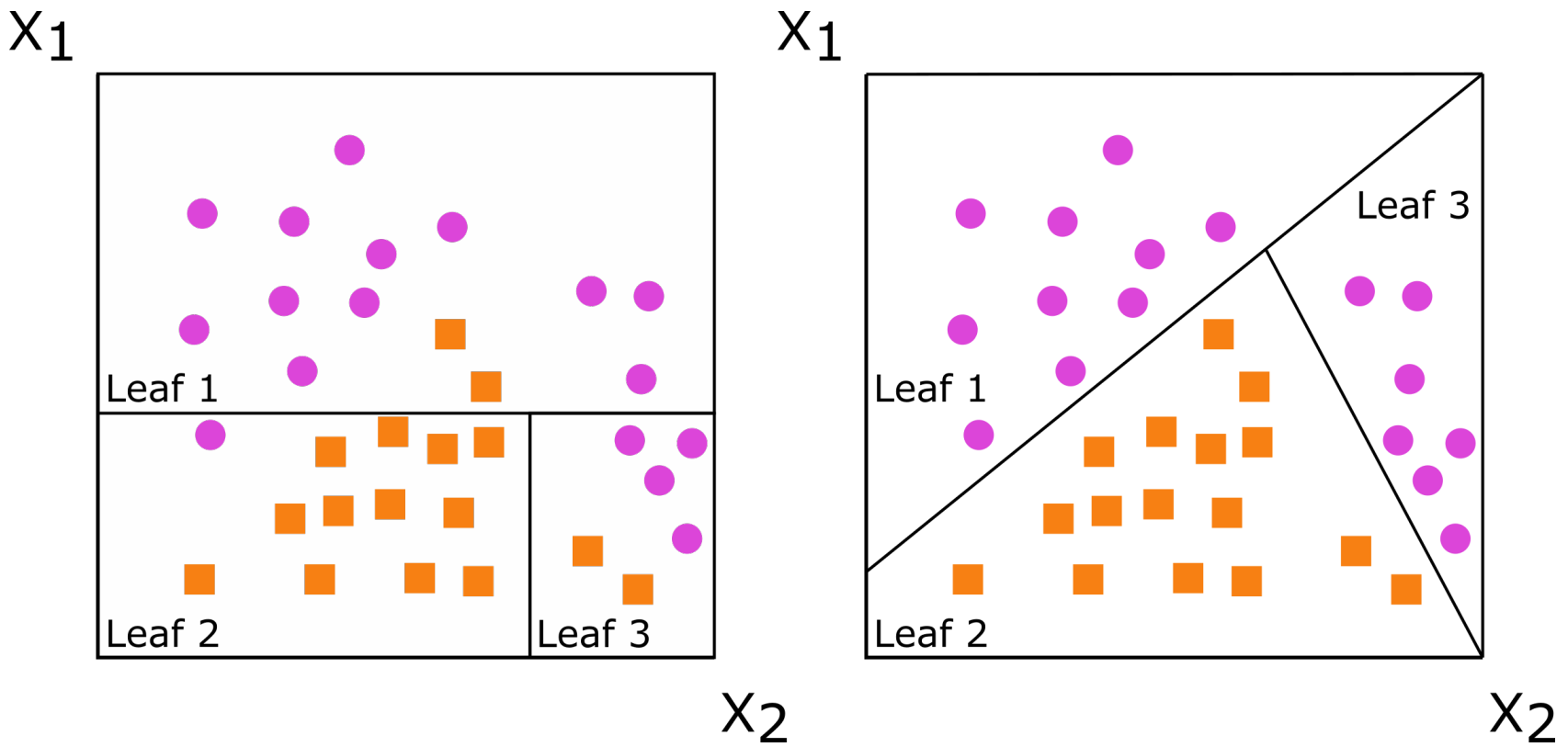
Ensemble predictions from a random forest



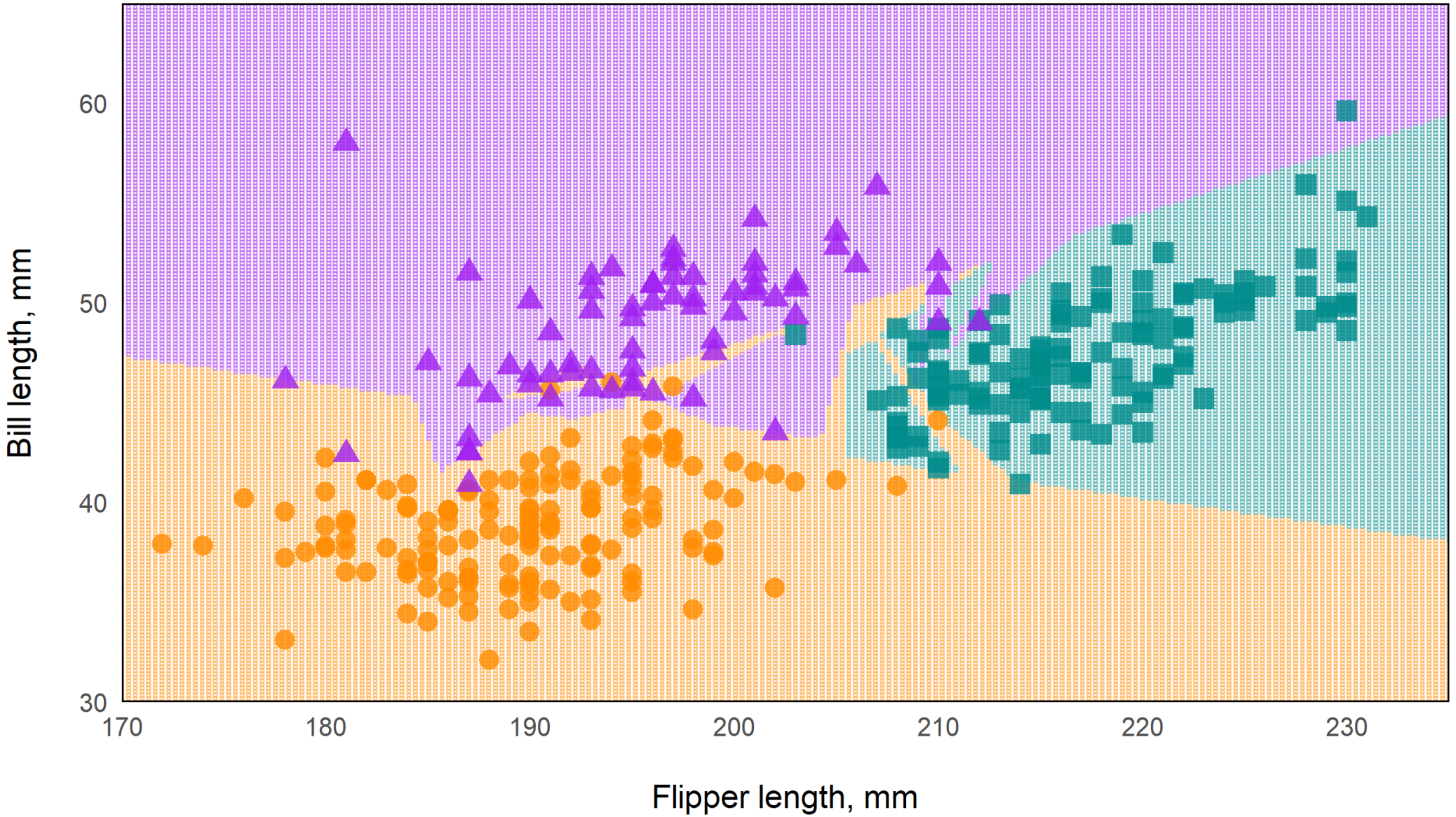
Oblique trees

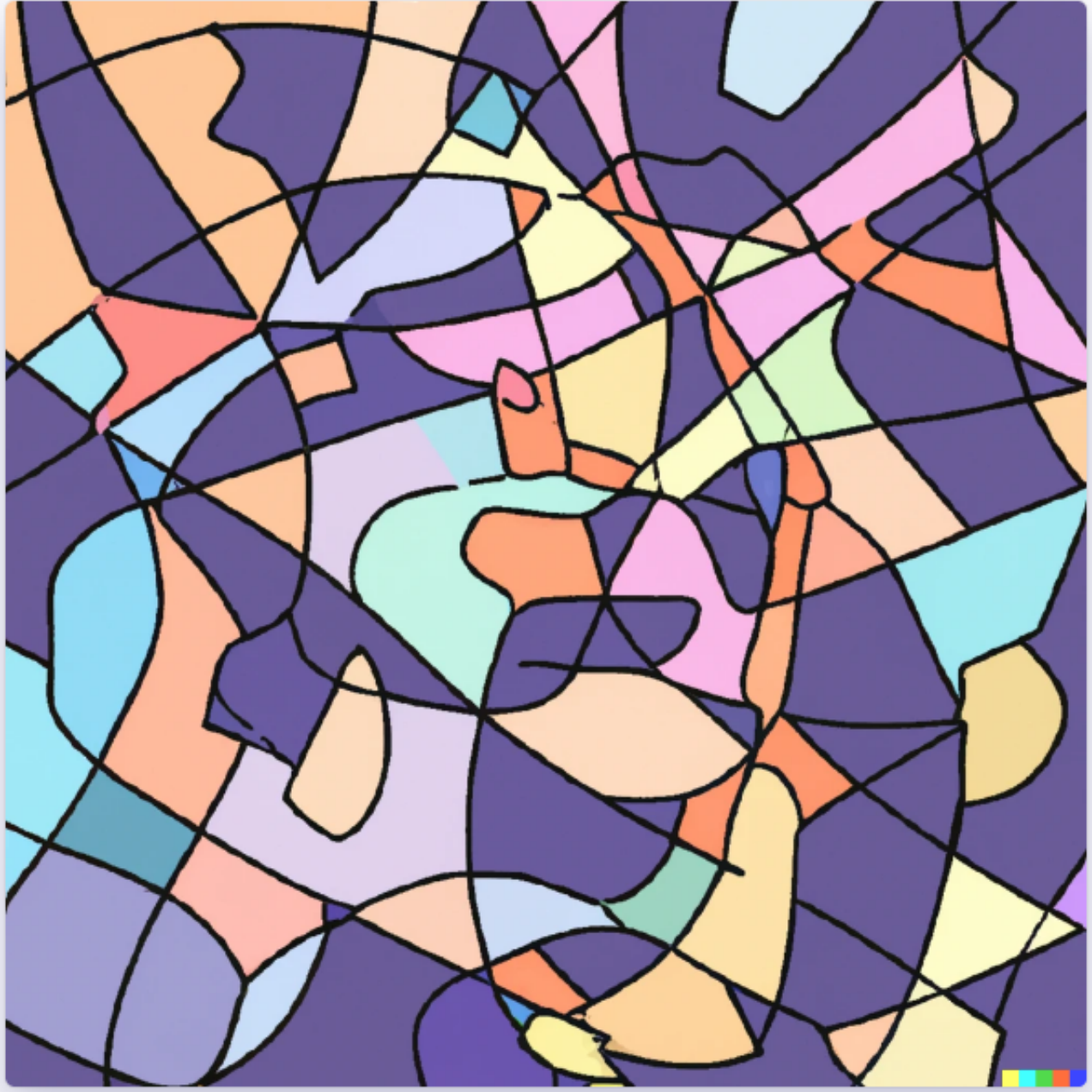
Instead of using one variable to split the data, use a weighted combination of variables.

I.e., instead of $x_1 <$ cutpoint, use $c_1 * x_1 + c_2 * x_2 <$ cutpoint



Predictions from **randomized individual oblique tree**



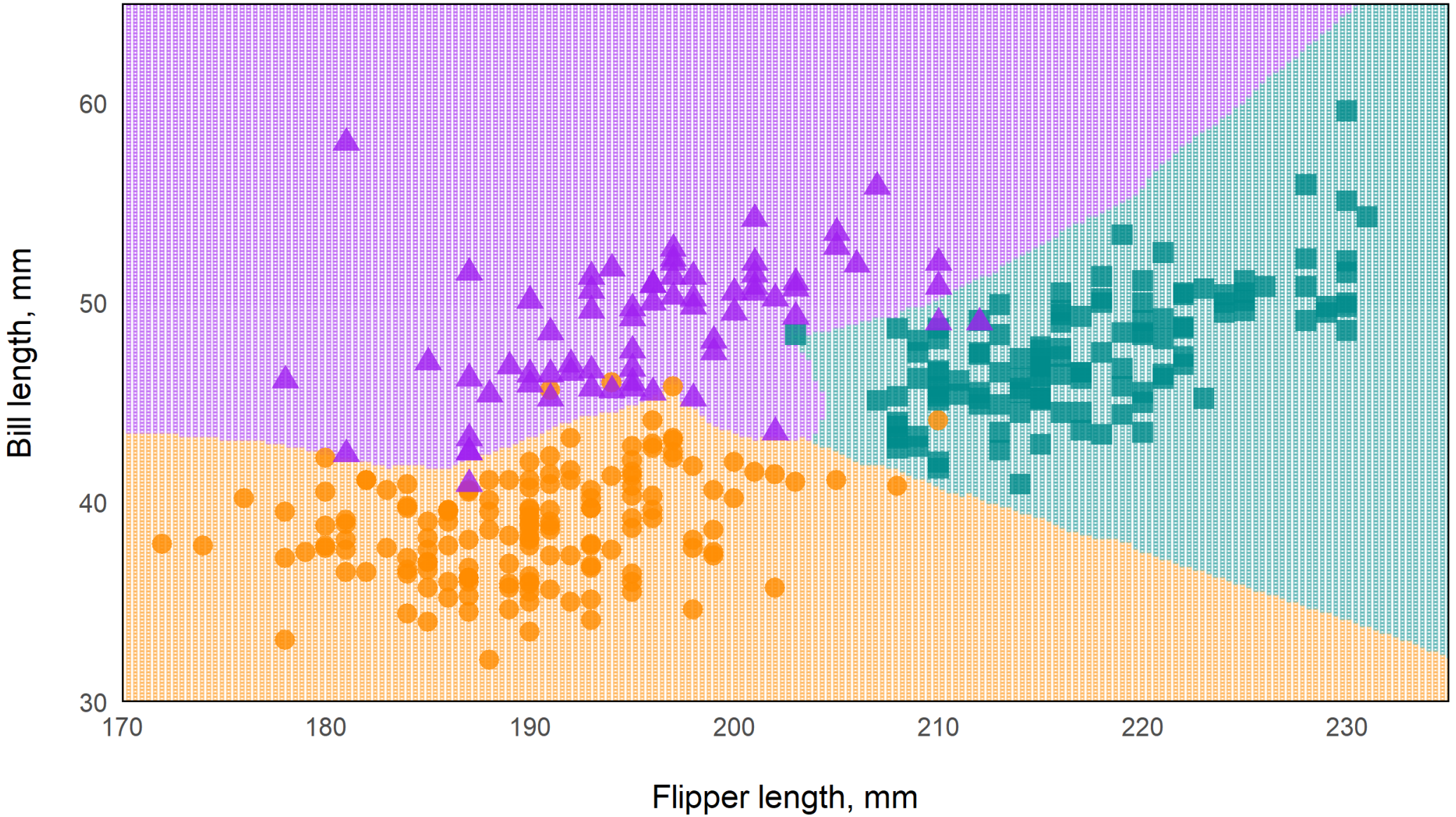


Created with DALL·E, an AI system by OpenAI

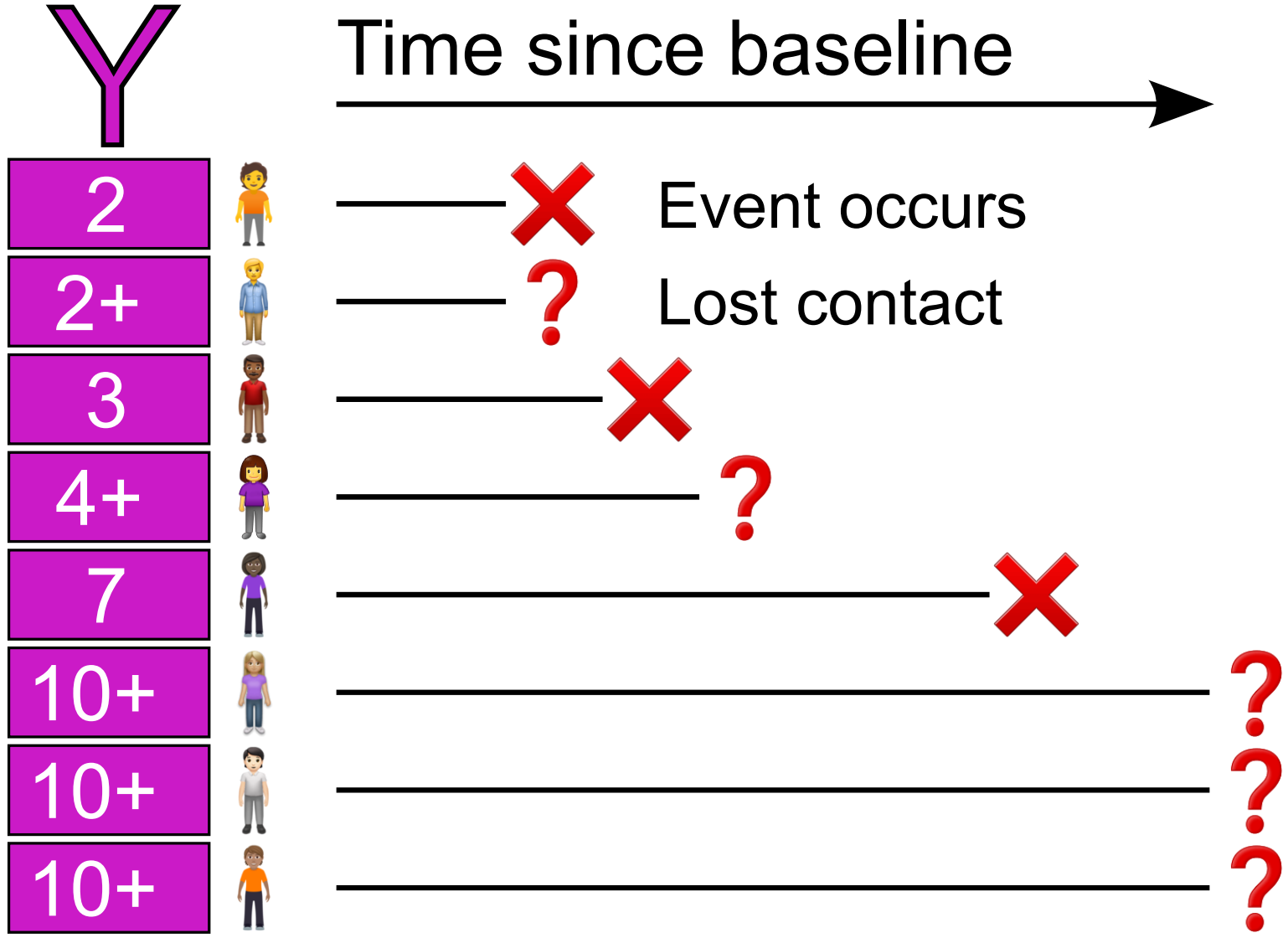
"a scatter plot in the style of pablo picasso, with purple, orange, and cyan colors"

Byron × DALL·E
Human & AI

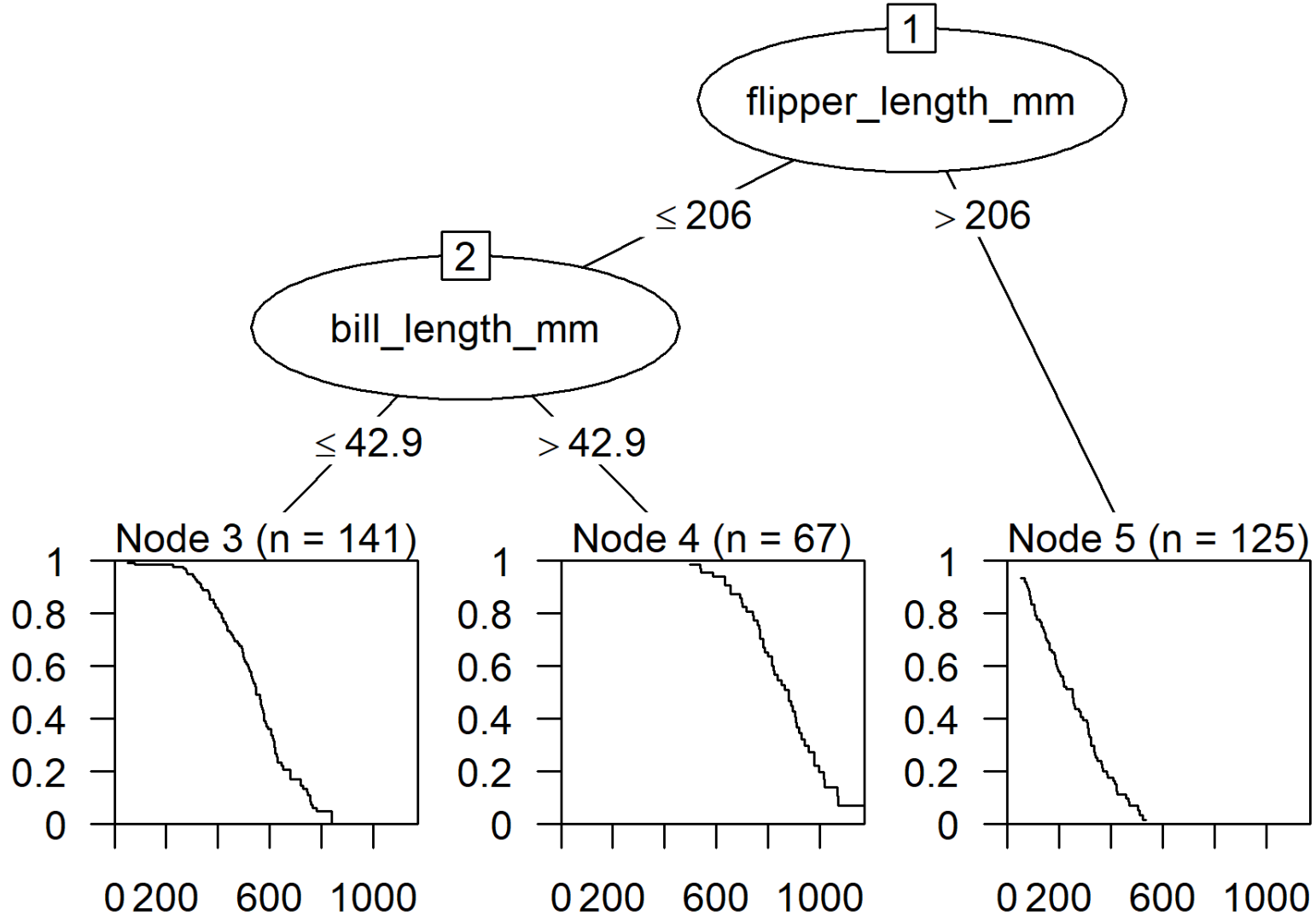
Ensemble predictions from an oblique random forest



Censoring


Y
Time since baseline
2
2+
3
4+
7
10+
10+
10+
Event occurs
Lost contact

Random Survival Forests (RSFs)


1
flipper_length_mm
≤ 206
> 206
2
bill_length_mm
≤ 42.9
> 42.9
Node 3 (n = 141)
Node 4 (n = 67)
Node 5 (n = 125)
1
0.8
0.6
0.4
0.2
0
0 200 600 1000

Oblique RSFs

Around 2018, I wrote some code to fit RSFs with oblique splits.

- Published a paper to share the idea + code.[1]
- TLDR: Cox regression in each non-leaf node, use the regression coefficients to make oblique splits.

Around 2020, that code got picked up and used for heart failure risk prediction.[2]

- oblique RSF did better than boosting, penalized, and stepwise regression! 😄
- oblique RSF was slow and hard to interpret. 😔

1. Jaeger BC, Long DL, Long DM, Sims M, Szychowski JM, Min YI, Mcclure LA, Howard G, Simon N. OBLIQUE RANDOM SURVIVAL FORESTS. Ann Appl Stat. 2019 Sep;13(3):1847-1883. doi: 10.1214/19-aoas1261. Epub 2019 Oct 17. PMID: 36704751; PMCID: PMC9875945.

2. Segar MW, Jaeger BC, Patel KV, Nambi V, Ndumele CE, Correa A, Butler J, Chandra A, Ayers C, Rao S, Lewis AA, Raffield LM, Rodriguez CJ, Michos ED, Ballantyne CM, Hall ME, Mentz RJ, de Lemos JA, Pandey A. Development and Validation of Machine Learning-Based Race-Specific Models to Predict 10-Year Risk of Heart Failure: A Multicohort Analysis. Circulation. 2021 Jun 15;143(24):2370-2383. doi: 10.1161/CIRCULATIONAHA.120.053134. Epub 2021 Apr 13. PMID: 33845593; PMCID: PMC9976274.


Tell me the truth...I'm...I'm ready
to hear it.
NOBODY USES SLOW R PACKAGES
imgflip.com

Oblique RSFs

Around 2021, I committed to `aorsf`, a re-write of my oblique RSF code

- Open review by rOpenSci and published in JOSS.[1]
- Large benchmark and more details in pre-print.[2]
- Named after the software my Dad wrote while I was growing up: AORSA[3]

1. Jaeger BC, Welden S, Lenoir K, Pajewski NM. aorsf: An R package for supervised learning using the oblique random survival forest. Journal of Open Source Software. 2022 Sep 28;7(77):4705.
2. Jaeger BC, Welden S, Lenoir K, Speiser JL, Segar MW, Pandey A, Pajewski NM. Accelerated and interpretable oblique random survival forests. arXiv preprint arXiv:2208.01129. 2022 Aug 1.
3. Article from Oak Ridge National Laboratory Cray User group featuring AORSA

Demo with aorsf

Fit an oblique RSF

Note: Copy/paste my code and run it locally[1]

Step 1: install/load the `aorsf` R package

```
# un-comment line below to install aorsf if needed
# install.packages('aorsf')

library(aorsf)
library(tidyverse) # will be used throughout
```

[1]Thanks to Garrick Aden-Buie's xaringanExtra for this awesome feature.

Fit an oblique RSF

Step 2: Inspect the `pbc_orsf` data

```
as_tibble(pbc_orsf)
```

```
## # A tibble: 276 x 20
##       id  time status trt     age sex   ascites hepato spiders edema  bili  chol
##    <int> <int>  <dbl> <fct> <dbl> <fct> <fct>   <fct>  <fct>   <fct> <dbl> <int>
##  1     1   400      1 d_pe~  58.8 f     1       1      1       1      14.5   261
##  2     2  4500      0 d_pe~  56.4 f     0       1      1       0       1.1   302
##  3     3  1012      1 d_pe~  70.1 m     0       0      0       0.5     1.4   176
##  4     4  1925      1 d_pe~  54.7 f     0       1      1       0.5     1.8   244
##  5     5  1504      0 plac~  38.1 f     0       1      1       0       3.4   279
##  6     7  1832      0 plac~  55.5 f     0       1      0       0       1     322
##  7     8  2466      1 plac~  53.1 f     0       0      0       0       0.3   280
##  8     9  2400      1 d_pe~  42.5 f     0       0      1       0       3.2   562
##  9    10    51      1 plac~  70.6 f     1       0      1       1      12.6   200
## 10    11  3762      1 plac~  53.7 f     0       1      1       0       1.4   259
## # i 266 more rows
## # i 8 more variables: albumin <dbl>, copper <int>, alk.phos <dbl>, ast <dbl>,
## #   trig <int>, platelet <int>, protime <dbl>, stage <ord>
```

Fit an oblique RSF

Step 3: Fit and inspect your oblique RSF

```
fit_orsf <- orsf(data = pbc_orsf,
                 formula = time + status ~ . - id)

fit_orsf
```

```
## ---------- Oblique random survival forest
##
##      Linear combinations: Accelerated
##           N observations: 276
##                 N events: 111
##                  N trees: 500
##       N predictors total: 17
##    N predictors per node: 5
##  Average leaves per tree: 25
## Min observations in leaf: 5
##       Min events in leaf: 1
##           OOB stat value: 0.84
##            OOB stat type: Harrell's C-statistic
##      Variable importance: anova
##
## -----------------------------------------
```

Fit an oblique RSF

Do you prefer the classic syntax?

```
library(survival)

fit_cph <- coxph(formula = Surv(time, status) ~ . - id,
                 data = pbc_orsf)
```

`aorsf` supports that:

```
fit_orsf <- orsf(formula = Surv(time, status) ~ . - id,
                 data = pbc_orsf)
```

Prefer pipes? `aorsf` supports that too.

```
set.seed(329) # use this seed to make your results match slides

fit_orsf <- pbc_orsf |>
  orsf(formula = time + status ~ . - id)
```

Fit an oblique RSF

Fan of `tidymodels`? (me too!) `aorsf` is a valid engine for censored regression.

```
library(parsnip)
library(censored) # must be version 0.2.0 or higher

rf_spec <-
  rand_forest(trees = 200) %>%
  set_engine("aorsf") %>%
  set_mode("censored regression")

fit_tidy <- rf_spec %>%
  parsnip::fit(data = pbc_orsf,
               formula = Surv(time, status) ~ . - id)

# fit printed on next slide for space
```

Fit an oblique RSF

Fan of `tidymodels`? (me too!) `aorsf` is a valid engine for censored regression.

```
fit_tidy
```

```
## parsnip model object
##
## ---------- Oblique random survival forest
##
##      Linear combinations: Accelerated
##           N observations: 276
##                 N events: 111
##                  N trees: 200
##       N predictors total: 17
##    N predictors per node: 5
##  Average leaves per tree: 25
## Min observations in leaf: 5
##       Min events in leaf: 1
##           OOB stat value: 0.84
##            OOB stat type: Harrell's C-statistic
##      Variable importance: anova
##
## ----------------------------------------
```

Interpret an oblique RSF

Get expected risk (i.e., partial dependence [pd]) in descending order of importance

```
orsf_summarize_uni(fit_orsf, n_variables = 1)
```

```
##
## -- bili (VI Rank: 1) ----------------------------
##
##         |---------------- risk ----------------|
##   Value       Mean    Median     25th %    75th %
##  <char>     <num>     <num>      <num>     <num>
##    0.80 0.2343668 0.1116206 0.04509389 0.3729834
##     1.4 0.2547884 0.1363122 0.05985486 0.4103148
##     3.5 0.3698634 0.2862611 0.16196924 0.5533383
##
##  Predicted risk at time t = 1788 for top 1 predictors
```

Interpret an oblique RSF

`orsf_pd()` functions give full control over expected risk

- `orsf_pd_inb()`: computes expected risk using all training data
- `orsf_pd_oob()`: computes expected risk using out-of-bag only
- `orsf_pd_new()`: computes expected risk using new data

```
pd_oob <- orsf_pd_oob(fit_orsf, pred_spec = list(bili = 1:5))

pd_oob
```

```
##    pred_horizon  bili      mean        lwr      medn       upr
##           <num> <num>     <num>      <num>     <num>     <num>
## 1:         1788     1 0.2390257 0.01072737 0.1137410 0.8746058
## 2:         1788     2 0.2898135 0.04198550 0.1849255 0.8930730
## 3:         1788     3 0.3466884 0.07529034 0.2593297 0.9112996
## 4:         1788     4 0.3906128 0.10570410 0.3148943 0.9213005
## 5:         1788     5 0.4346375 0.15449708 0.3647937 0.9305869
```

Why is `pred_horizon` set at 1788? If you don't set a `pred_horizon`, `orsf` functions will set it for you as the median follow-up time

Interpret an oblique RSF

Use `pred_horizon` to make expected risk more interpretable.

- expected risk for men and women at 1, 2, 3, 4, and 5 years:

```
pd_by_gender <- orsf_pd_oob(fit_orsf,
                            pred_spec = list(sex = c("m", "f")),
                            pred_horizon = 365 * 1:5)

pd_by_gender %>%
  dplyr::select(pred_horizon, sex, mean) %>%
  tidyr::pivot_wider(names_from = sex, values_from = mean) %>%
  dplyr::mutate(ratio = m / f)
```

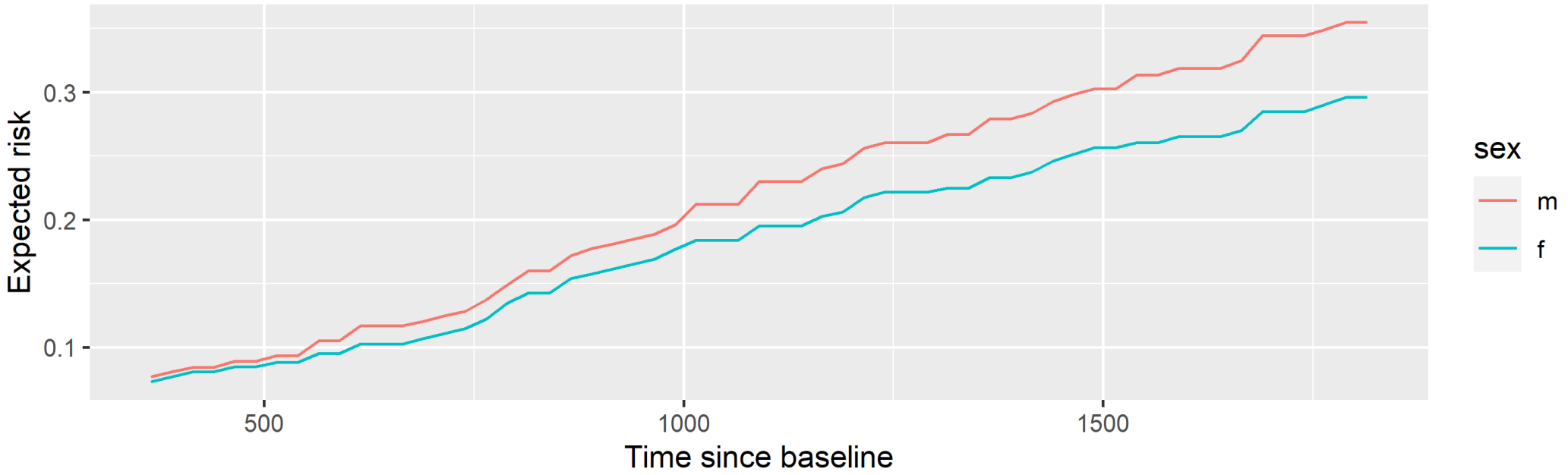
```
## # A tibble: 5 x 4
##   pred_horizon      m      f ratio
##          <dbl>  <dbl>  <dbl> <dbl>
## 1          365 0.0768 0.0728  1.06
## 2          730 0.125  0.111   1.13
## 3         1095 0.230  0.195   1.18
## 4         1460 0.298  0.251   1.19
## 5         1825 0.355  0.296   1.20
```

Does expected risk increase faster over time for men?

Interpret an oblique RSF

Use `pred_horizon` to investigate expected risk profiles over time

```
fit_orsf %>%
  orsf_pd_oob(pred_spec = list(sex = c("m", "f")),
              pred_horizon = seq(365, 365*5, by = 25)) %>%
  ggplot(aes(x = pred_horizon, y = mean, color = sex)) +
  geom_line() +
  labs(x = 'Time since baseline', y = 'Expected risk')
```



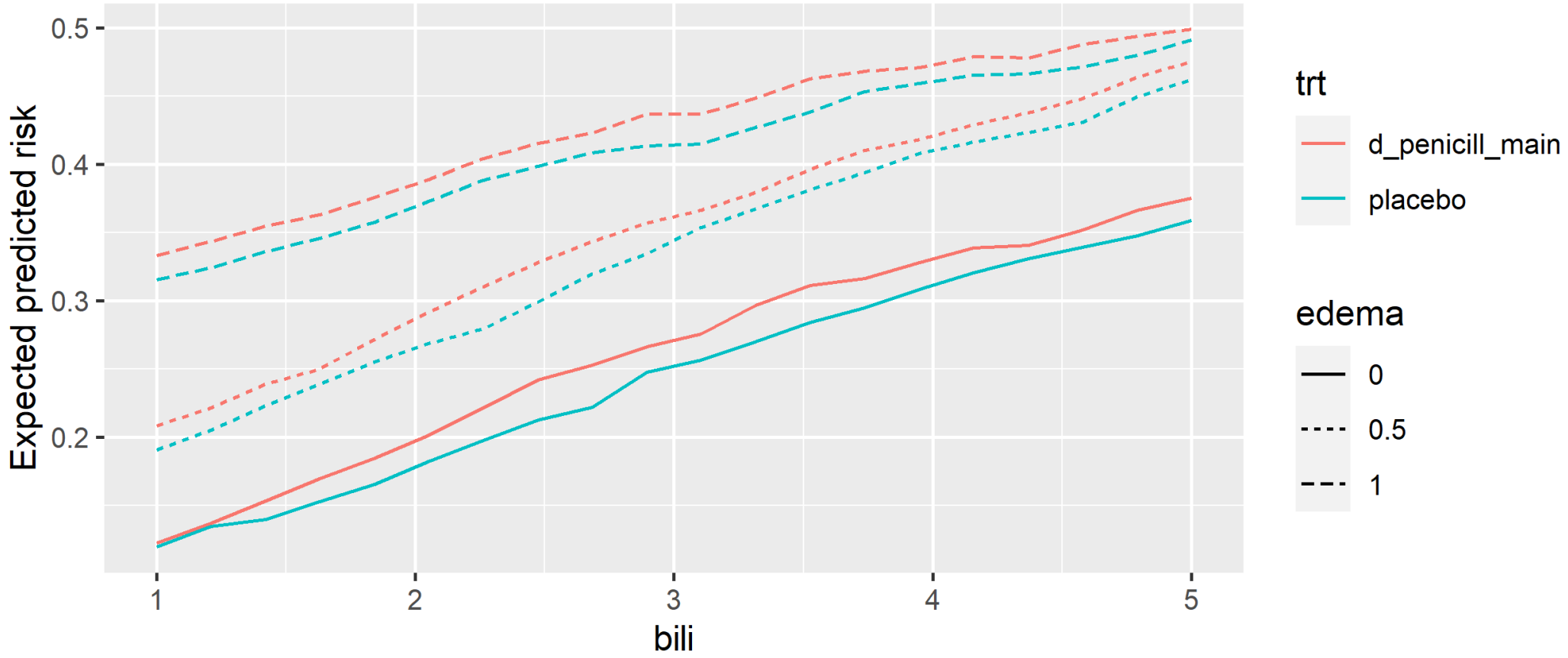
Interpret an oblique RSF

Or, fix `pred_horizon` and look at risk profiles over other variables

```
pred_spec = list(bili = seq(1, 5, length.out = 20),
                 edema = levels(pbc_orsf$edema),
                 trt = levels(pbc_orsf$trt))

pd_bili_edema <- orsf_pd_oob(fit_orsf, pred_spec)

fig <- ggplot(pd_bili_edema) +
  aes(x = bili, y = medn, col = trt, linetype = edema) +
  geom_line() +
  labs(y = 'Expected predicted risk')

# fig on next slide for space
```

Interpret an oblique RSF

Or, fix `pred_horizon` and look at risk profiles over other variables



Variable importance

ANOVA importance

A fast method to compute variable importance with (some) oblique random forests:

For each predictor:

1. Compute p-values for each coefficient
2. Compute the proportion of p-values that are low (<0.01)

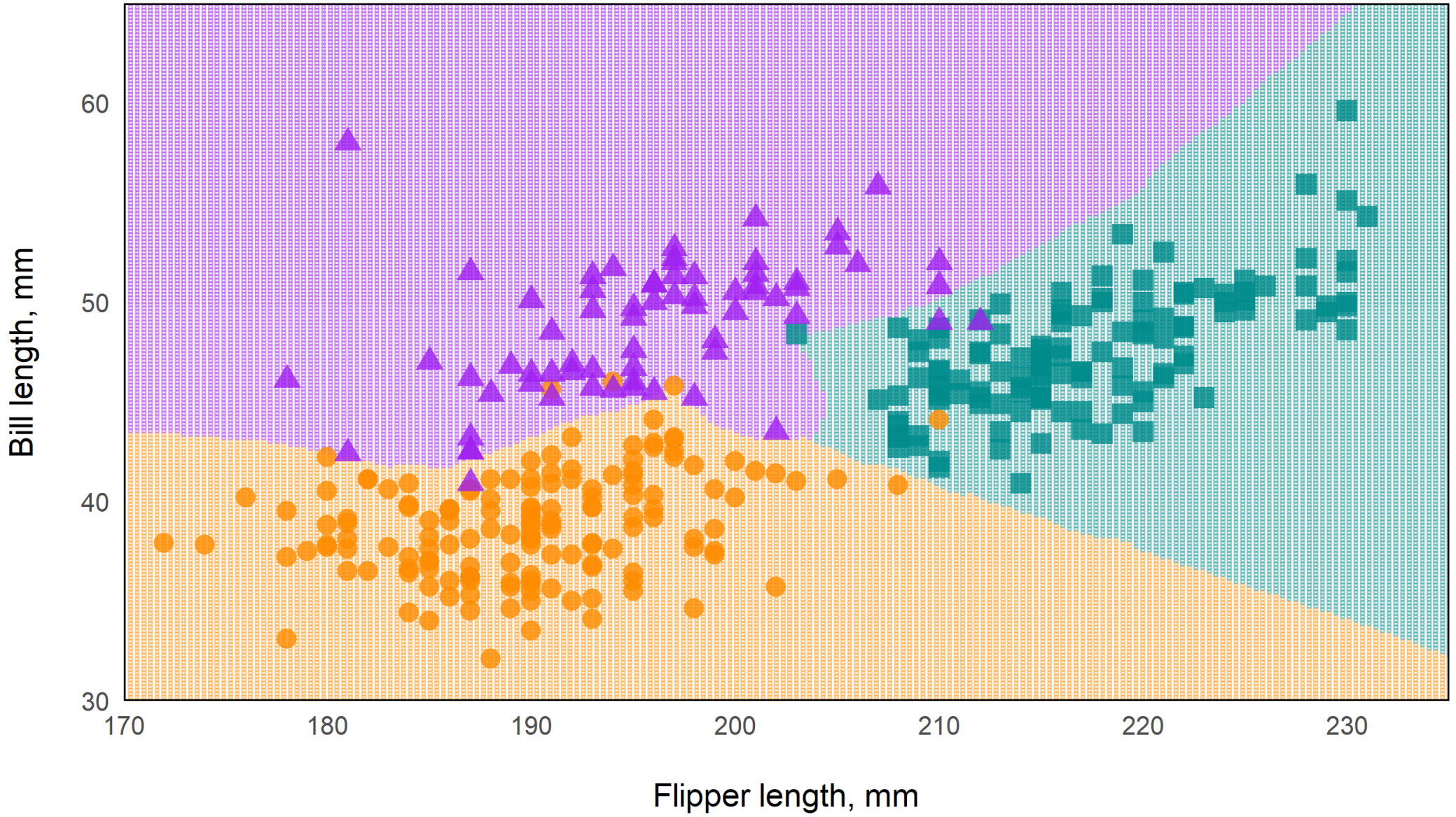
Importance of a predictor = the proportion of times its p-value is low.

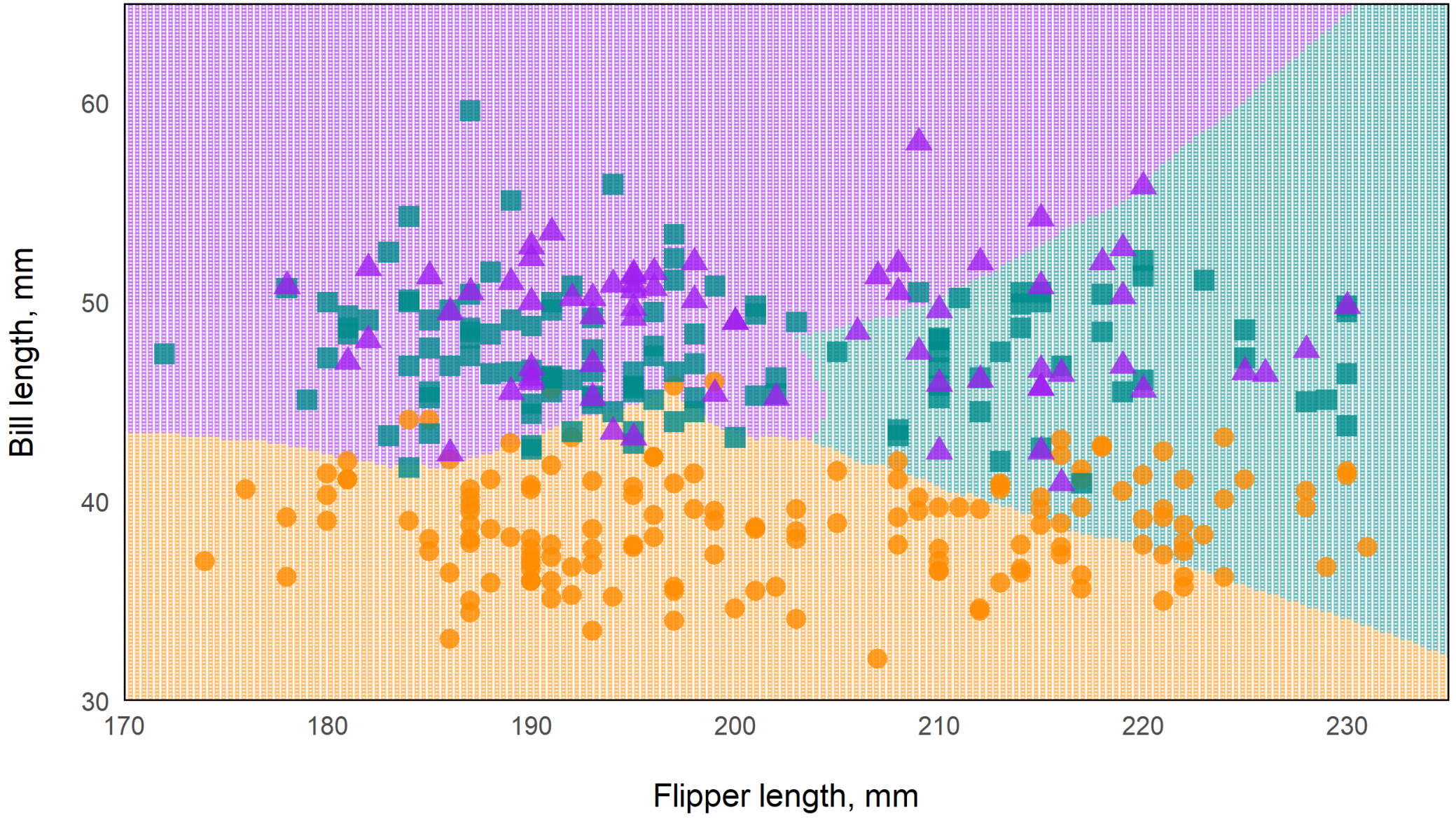
Permutation importance

For each predictor:

1. Permute predictor values
2. Measure prediction error with permuted values

Importance of a predictor = increase in prediction error after permutation


Bill length, mm
60
50
40
30
170
180
190
200
210
220
230
Flipper length, mm

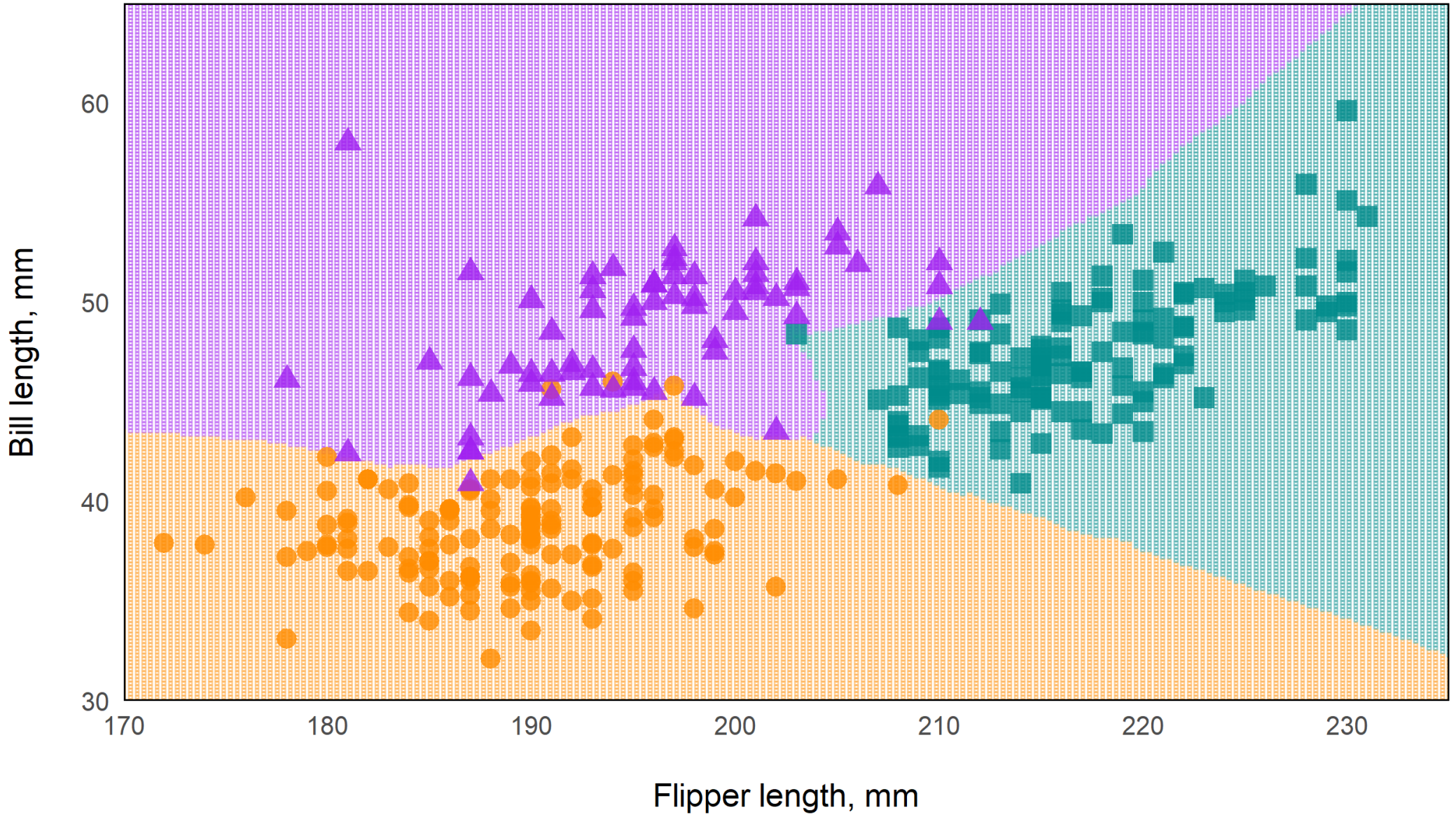

Bill length, mm
60
50
40
30
170
180
190
200
210
220
230
Flipper length, mm

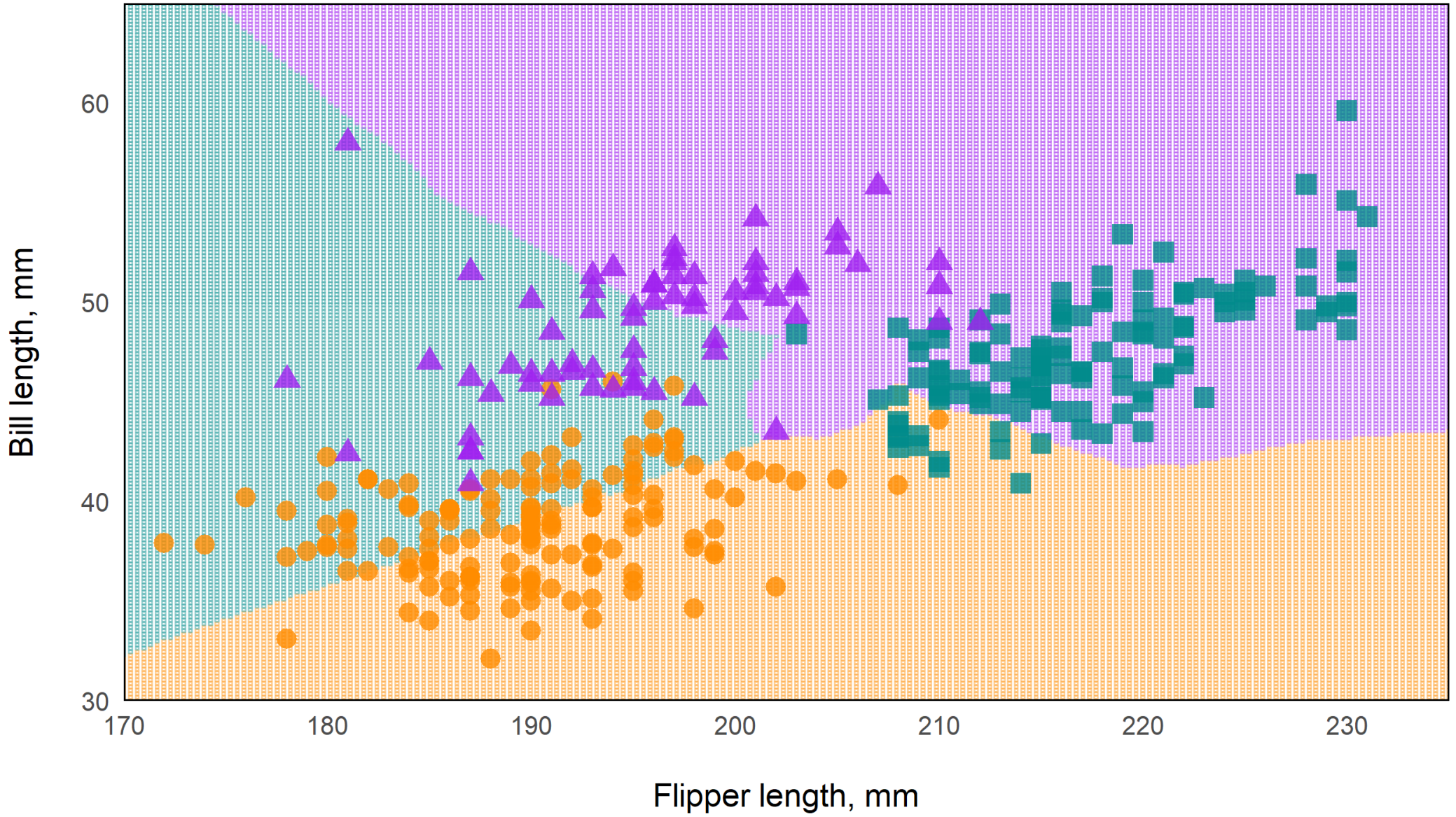
Negation importance

For each predictor:

1. Multiply coefficient in linear combination by -1
2. Measure prediction error with permuted values

Importance of a predictor = increase in prediction error after negation


Bill length, mm
60
50
40
30
170
180
190
200
210
220
230
Flipper length, mm


Bill length, mm
60
50
40
30
170
180
190
200
210
220
230
Flipper length, mm

Variable importance

```
orsf_vi_anova(fit_orsf)[1:5]
```

```
##   ascites      bili     edema    copper   albumin
## 0.3832659 0.2720345 0.2383323 0.2016109 0.1750125
```

```
orsf_vi_permute(fit_orsf)[1:5]
```

```
##        bili         age      copper     albumin     protime
## 0.016670140 0.008908106 0.005521984 0.005365701 0.003594499
```

```
orsf_vi_negate(fit_orsf)[1:5]
```

```
##       bili        age     copper    protime    albumin
## 0.07637008 0.02740154 0.02505730 0.01302355 0.01000208
```

But how accurate is `aorsf`'s predicted risk?

Benchmarking `aorsf`

Let's run a benchmark using `mlr3`[1] comparing `aorsf`'s oblique RSF to:

- axis-based RSFs in `randomForestSRC`
- axis-based RSFs in `ranger`

We'll run an experiment where

1. each approach is used to make a prediction model with the same training data
2. each model is validated in the same external data.

Here are the packages we'll use:

```
library(mlr3verse)
library(mlr3proba)
library(mlr3extralearners)
library(mlr3viz)
library(mlr3benchmark)
library(OpenML) # for some of the prediction tasks
```

[1] `aorsf` is compatible with `mlr3` too. Find more about `mlr3` here: https://mlr3.mlr-org.com/

Benchmarking `aorsf`

Retrieve some public data and set it up as a `mlr3` task object.

- Mayo Clinic Primary Biliary Cholangitis Data

```
task_pbc <-
 TaskSurv$new(
  id = 'pbc',
  backend = dplyr::select(pbc_orsf, -id) %>%
   dplyr::mutate(stage = as.numeric(stage)),
  time = "time",
  event = "status"
 )
```

Benchmarking `aorsf`

Retrieve some public data and set it up as a `mlr3` task object.

- Veteran's Administration Lung Cancer Trial

```
data(veteran, package = "randomForestSRC")

task_veteran <-
 TaskSurv$new(
  id = 'veteran',
  backend = veteran,
  time = "time",
  event = "status"
 )
```

Benchmarking `aorsf`

Retrieve some public data and set it up as a `mlr3` task object.

- NKI 70 gene signature

```
data_nki <- OpenML::getOMLDataSet(data.id = 1228)

task_nki <-
 TaskSurv$new(
  id = 'nki',
  backend = data_nki$data,
  time = "time",
  event = "event"
 )
```

Benchmarking `aorsf`

Retrieve some public data and set it up as a `mlr3` task object.

- Gene Expression-Based Survival Prediction in Lung Adenocarcinoma

```
data_lung <- OpenML::getOMLDataSet(data.id = 1245)

task_lung <-
 TaskSurv$new(
  id = 'nki',
  backend = data_lung$data %>%
   dplyr::mutate(OS_event = as.numeric(OS_event) -1),
  time = "OS_years",
  event = "OS_event"
 )
```

Benchmarking `aorsf`

Retrieve some public data and set it up as a `mlr3` task object.

- Chemotherapy for Stage B/C colon cancer

```
# there are two rows per person, one for death
# and the other for recurrence, hence the two tasks

task_colon_death <-
 TaskSurv$new(
  id = 'colon_death',
  backend = survival::colon %>%
   dplyr::filter(etype == 2) %>%
   tidyr::drop_na() %>%
   # drop id, redundant variables
   dplyr::select(-id, -study, -node4, -etype),
   dplyr::mutate(OS_event = as.numeric(OS_event) -1),
  time = "time",
  event = "status"
 )
```

Benchmarking `aorsf`

Retrieve some public data and set it up as a `mlr3` task object.

- Chemotherapy for Stage B/C colon cancer

```
# there are two rows per person, one for death
# and the other for recurrence, hence the two tasks

task_colon_recur <-
 TaskSurv$new(
  id = 'colon_death',
  backend = survival::colon %>%
   dplyr::filter(etype == 1) %>%
   tidyr::drop_na() %>%
   # drop id, redundant variables
   dplyr::select(-id, -study, -node4, -etype),
   dplyr::mutate(OS_event = as.numeric(OS_event) -1),
  time = "time",
  event = "status"
 )
```

Benchmarking `aorsf`

TLDR; we have 11 different prediction tasks for this benchmark.

```
# putting them all together
tasks <- list(task_pbc,
              task_veteran,
              task_nki,
              task_lung,
              task_colon_death,
              task_colon_recur,
              # add a few more pre-made ones
              tsk("actg"),
              tsk("gbcs"),
              tsk("grace"),
              tsk("unemployment"),
              tsk("whas"))
```

Benchmarking `aorsf`

With these tasks we can run a benchmark on our favorite learners:

```
# Learners with default parameters
learners <- lrns(c("surv.ranger", "surv.rfsrc", "surv.aorsf"))

# Brier (Graf) score, c-index and training time as measures
measures <- msrs(
  c("surv.graf", # Brier score (lower is better)
    "surv.cindex", # C-index (higher is better)
    "surv.calib_alpha", # Calibration slope (1 is best)
    "time_train") # time to fit model
)

# Benchmark with 5-fold CV
design <- benchmark_grid(
  tasks = tasks,
  learners = learners,
  resamplings = rsmps("cv", folds = 5)
)

benchmark_result <- benchmark(design)

bm_scores <- benchmark_result$score(measures, predict_sets = "test")
```

Benchmarking `aorsf`

Summarize by taking the mean over all 11 tasks, and all 5 folds:

```
bm_scores %>%
 select(task_id, learner_id, starts_with('surv'), time_train) %>%
 group_by(learner_id) %>%
 filter(!is.infinite(surv.graf)) %>%
 summarize(
  across(
   .cols = c(surv.graf, surv.cindex, surv.calib_alpha, time_train),
   .fns = mean
  )
 )
```

```
## # A tibble: 3 x 5
##   learner_id  surv.graf surv.cindex surv.calib_alpha time_train
##   <chr>           <dbl>       <dbl>            <dbl>      <dbl>
## 1 surv.aorsf      0.143       0.734            0.994      0.301
## 2 surv.ranger     0.156       0.718            1.07       2.08
## 3 surv.rfsrc      0.146       0.724            0.985      0.711
```

Extending aorsf

Extending aorsf

`aorsf` includes a family of control functions:

- `orsf_control_cph()` uses a full Cox regression
- `orsf_control_fast()` uses partial Cox regression (default)
- `orsf_control_net()` uses penalized Cox regression (slower)
- `orsf_control_custom()` uses whatever you give it

Extending aorsf

Make an oblique RSF that makes oblique splits randomly:

```
f_rando <- function(x_node, y_node, w_node){
 matrix(runif(ncol(x_node)), ncol=1)
}

fit_rando <- orsf(pbc_orsf,
                  Surv(time, status) ~ . - id,
                  control = orsf_control_custom(beta_fun = f_rando),
                  n_tree = 500)
```

Extending aorsf

Make an oblique RSF that makes oblique splits with principal component analysis:

```
f_pca <- function(x_node, y_node, w_node) {

 # estimate two principal components.
 pca <- stats::prcomp(x_node, rank. = 2)
 # use the second principal component to split the node
 pca$rotation[, 2L, drop = FALSE]

}

fit_pca <- orsf(pbc_orsf,
                Surv(time, status) ~ . - id,
                control = orsf_control_custom(beta_fun = f_pca),
                n_tree = 500)
```

Extending aorsf

Compare these to our initial fit:

```
library(riskRegression)

risk_preds <- list(dflt = 1 - fit_orsf$pred_oobag,
                   rando = 1 - fit_rando$pred_oobag,
                   pca = 1 - fit_pca$pred_oobag)

sc <- Score(object = risk_preds,
            formula = Surv(time, status) ~ 1,
            data = pbc_orsf,
            summary = 'IPA',
            times = fit_pca$pred_horizon)
```

Extending aorsf

Compare these to our initial fit:

```
##
## Results by model:
##
##     model times    AUC  lower  upper
##    <fctr> <num> <char> <char> <char>
## 1:   dflt  1788 90.820 86.599 95.041
## 2:  rando  1788 88.557 84.078 93.036
## 3:    pca  1788 89.754 85.310 94.198
##
## Results of model comparisons:
##
##    times  model reference delta.AUC  lower  upper      p
##    <num> <fctr>    <fctr>    <char> <char> <char>  <num>
## 1:  1788  rando      dflt    -2.263 -4.303 -0.222 0.0297
## 2:  1788    pca      dflt    -1.066 -2.702  0.571 0.2018
## 3:  1788    pca     rando     1.197 -1.120  3.515 0.3113
```

What we covered

Fitting oblique RSFs with `aorsf`

- Similar to `coxph` syntax, usable as `tidymodels` engine and `mlr3` learner

Interpreting oblique RSFs with `aorsf`

- Partial dependence and variable importance

Benchmarking `aorsf`

- Excellent discrimination and calibration compared to axis-based RSF
- `aorsf` is computationally efficient (caveat: no parallel computing)

Extending `aorsf`

- Customize your oblique RSF with `orsf_control` functions.

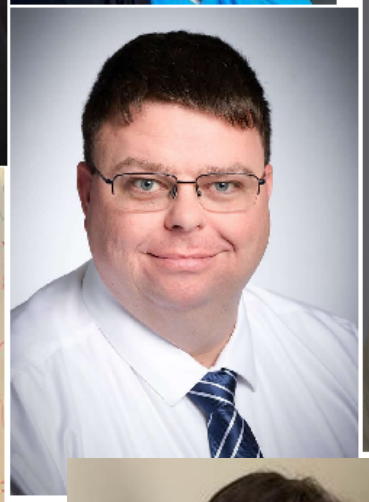
Acknowledgments

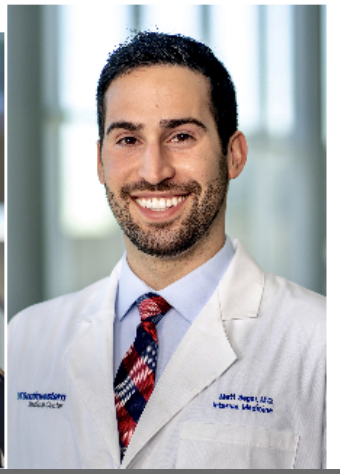
Research reported in this presentation was supported by

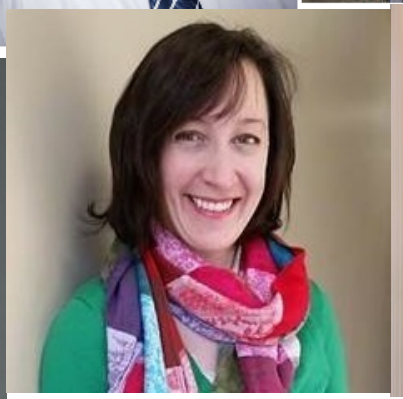
- Center for Biomedical Informatics, Wake Forest University School of Medicine.
- National Center for Advancing Translational Sciences (NCATS), National Institutes of Health, through Grant Award Number UL1TR001420.

The content is solely the responsibility of the authors and does not necessarily represent the official views of the NIH.

THANK YOU!!







Wake Forest University
School of Medicine