Ensemble Trees

Ravi Brenner

```
library(tidyverse)
```

```
## Warning: package 'ggplot2' was built under R version 4.5.2
```

```
## Warning: package 'readr' was built under R version 4.5.2
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.6
## v forcats   1.0.1     v stringr   1.6.0
## v ggplot2   4.0.1     v tibble    3.3.0
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.2.0
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(tidymodels)
```

```
## -- Attaching packages --------------------------------------- tidymodels 1.4.1 --
## v broom        1.0.10     v rsample      1.3.1
## v dials        1.4.2      v tailor       0.1.0
## v infer        1.0.9      v tune         2.0.1
## v modeldata    1.5.1      v workflows    1.3.0
## v parsnip      1.4.0      v workflowsets 1.1.1
## v recipes      1.3.1      v yardstick    1.3.2
```

```
## Warning: package 'parsnip' was built under R version 4.5.2
```

```
## -- Conflicts ------------------------------------------ tidymodels_conflicts() --
## x scales::discard() masks purrr::discard()
## x dplyr::filter()   masks stats::filter()
## x recipes::fixed()  masks stringr::fixed()
## x dplyr::lag()      masks stats::lag()
## x yardstick::spec() masks readr::spec()
## x recipes::step()   masks stats::step()
```

```
data(ames)
data(credit_data)
```

```
set.seed(2025)

ames_split <- initial_split(ames, prop = 0.8)
ames_train <- training(ames_split)
ames_test <- testing(ames_split)
ames_cv <- vfold_cv(ames_train)

credit_split <- initial_split(credit_data, prop = 0.8, strata = Status)
```

```r
credit_train <- training(credit_split)
credit_test <- testing(credit_split)
credit_cv <- vfold_cv(credit_train)


reg_recipe <-
  recipe(Sale_Price ~ ., data = ames_train)
```

```r
# Calculate the test set bias, variance, MSE with bootstrapping
bootstrap_predictions <- function(workflow_in, train_data, test_data){
  fit_model <- function(split, workflow) {
    fit(workflow, data = training(split))
  }

  set.seed(2025)
  ames_bootstraps <- bootstraps(train_data, times = 100)

  all_predictions <-
    ames_bootstraps |>
    mutate(
      model_fit = map(splits, fit_model, workflow = workflow_in),
      predictions = map(model_fit, ~ predict(.x, new_data = test_data) |> rename(pred = .pred))
    )

  prediction_df <-
    all_predictions |>
    select(id, predictions) |>
    unnest(predictions) |>
    group_by(id) |>
    mutate(Sale_Price = rep(test_data$Sale_Price, length.out = n()),
           obs_id = row_number()) |>
    ungroup()

  return(prediction_df)
}

bias_var_mse <- function(pred_df){
  mse_df <-
    pred_df |>
    group_by(obs_id) |>
    summarise(
      bias_sq = (mean(pred) - mean(Sale_Price))^2,
      variance = stats::var(pred),
      mse = mean((pred - Sale_Price)^2),
      sale_price = first(Sale_Price)
    ) |>
    ungroup() |>
    summarise(
      mean_bias_sq = mean(bias_sq),
      mean_variance = mean(variance),
      mean_mse = mean(mse),
      irr_error = mean_mse - mean_bias_sq - mean_variance
    )
  return(mse_df)
```

```
}
```

Regression case

Deep CART

as a baseline

```
set.seed(2025)
cart_mod <-
  decision_tree(mode = "regression",
                engine = "rpart",
                cost_complexity = 0,
                tree_depth = 30,
                min_n = 20)

reg_recipe <-
  recipe(Sale_Price ~ ., data = ames_train)

cart_workflow <-
  workflow() |>
  add_model(cart_mod) |>
  add_recipe(reg_recipe)

cart_fit <- fit(cart_workflow,
                data = ames_train)

cart_boot_preds <- bootstrap_predictions(cart_workflow, ames_train, ames_test)

cart_tradeoff <- bias_var_mse(cart_boot_preds) |>
  mutate(rmse = sqrt(mean_mse))
```

Shallow CART

```
set.seed(2025)
cart_mod_short <-
  decision_tree(mode = "regression",
                engine = "rpart",
                cost_complexity = 0,
                tree_depth = 3,
                min_n = 20)

cart_short_workflow <-
  workflow() |>
  add_model(cart_mod_short) |>
  add_recipe(reg_recipe)

cart_short_fit <- fit(cart_short_workflow,
                      data = ames_train)

cart_short_boot_preds <- bootstrap_predictions(cart_short_workflow, ames_train, ames_test)

cart_short_tradeoff <- bias_var_mse(cart_short_boot_preds) |>
```

```
  mutate(rmse = sqrt(mean_mse))
```

Bagging

```
set.seed(2025)
bag_mod <-
  rand_forest(mode = "regression",
              engine = "ranger",
              mtry = ncol(ames) - 1,
              trees = 100,
              min_n = 20)

bag_workflow <-
  workflow() |>
  add_model(bag_mod) |>
  add_recipe(reg_recipe)

bag_fit <-
  bag_workflow |>
  fit(data = ames_train)

bag_boot_preds <- bootstrap_predictions(bag_workflow, ames_train, ames_test)

bag_tradeoff <- bias_var_mse(bag_boot_preds) |>
  mutate(rmse = sqrt(mean_mse))
```

Show that variance is reduced relative to CART

Random Forests

should reduce variance relative to bagging, especially when predictors are correlated may need to sim this

```
set.seed(2025)
rf_mod <-
  rand_forest(mode = "regression",
              engine = "ranger",
              mtry = floor((ncol(ames) - 1)/3),
              trees = 100,
              min_n = 20)

rf_workflow <-
  workflow() |>
  add_model(rf_mod) |>
  add_recipe(reg_recipe)

rf_fit <-
  rf_workflow |>
  fit(data = ames_train)

rf_boot_preds <- bootstrap_predictions(rf_workflow, ames_train, ames_test)

rf_tradeoff <- bias_var_mse(rf_boot_preds) |>
  mutate(rmse = sqrt(mean_mse))
```

Too complex s.t. bias starts to increase more and mse starts to increase tooo

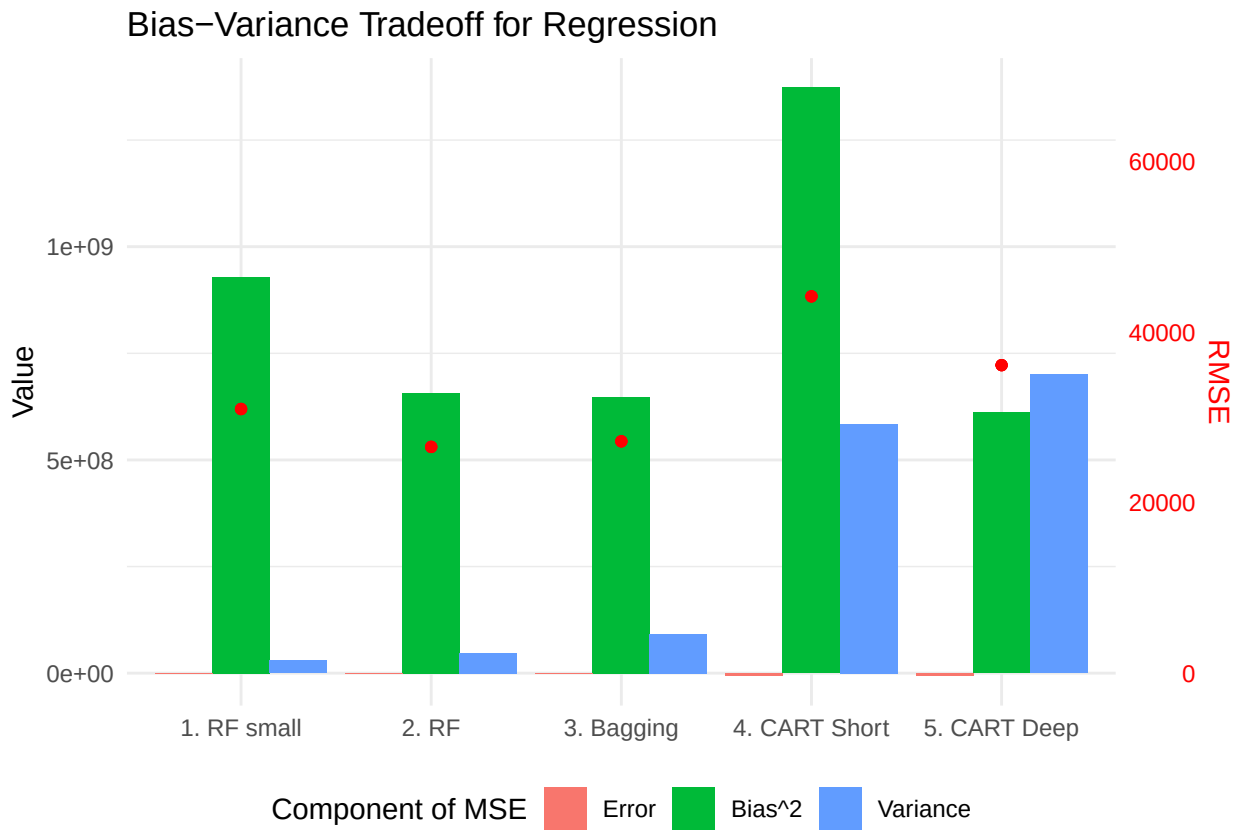
i think rf with a smaller mtry would work

```
set.seed(2025)
rf_short_mod <-
  rand_forest(mode = "regression",
              engine = "ranger",
              mtry = 3,
              trees = 100,
              min_n = 20)

rf_short_workflow <-
  workflow() |>
  add_model(rf_short_mod) |>
  add_recipe(reg_recipe)

rf_short_fit <-
  rf_short_workflow |>
  fit(data = ames_train)


rf_short_boot_preds <- bootstrap_predictions(rf_short_workflow, ames_train, ames_test)

rf_short_tradeoff <- bias_var_mse(rf_short_boot_preds) |>
  mutate(rmse = sqrt(mean_mse))
```

comparison

```
cart_short_tradeoff |>
  mutate(name = "4. CART Short") |>
  bind_rows(cart_tradeoff |>
              mutate(name = "5. CART Deep")) |>
  bind_rows(bag_tradeoff |>
              mutate(name = "3. Bagging")) |>
  bind_rows(rf_tradeoff |>
              mutate(name = "2. RF")) |>
  bind_rows(rf_short_tradeoff |>
              mutate(name = "1. RF small")) |>
  pivot_longer(cols = c(mean_bias_sq, mean_variance, irr_error),
               names_to = "variable",
               values_to = "value") |>
  ggplot(aes(x = name)) +
  geom_bar(aes(y = value, fill = variable),
           position = "dodge", stat = "identity") +
  geom_point(aes(y = 20000*rmse),
             color = "red") +
  scale_fill_discrete(name = "Component of MSE",
                      labels = c("Error",
                                 "Bias^2",
                                 "Variance")) +
  scale_y_continuous(sec.axis = sec_axis(transform = ~./20000,
                                         name = "RMSE")) +
  labs(x = NULL,
```

```
       y = "Value",
       title = "Bias-Variance Tradeoff for Regression") +
  theme_minimal() +
  theme(axis.text.y.right = element_text(color = "red"),
        axis.title.y.right = element_text(color = "red"),
        legend.position = "bottom")
```




```
ggsave("output/reg_bv_results.jpg",
       width = 6.5, height = 3)
```

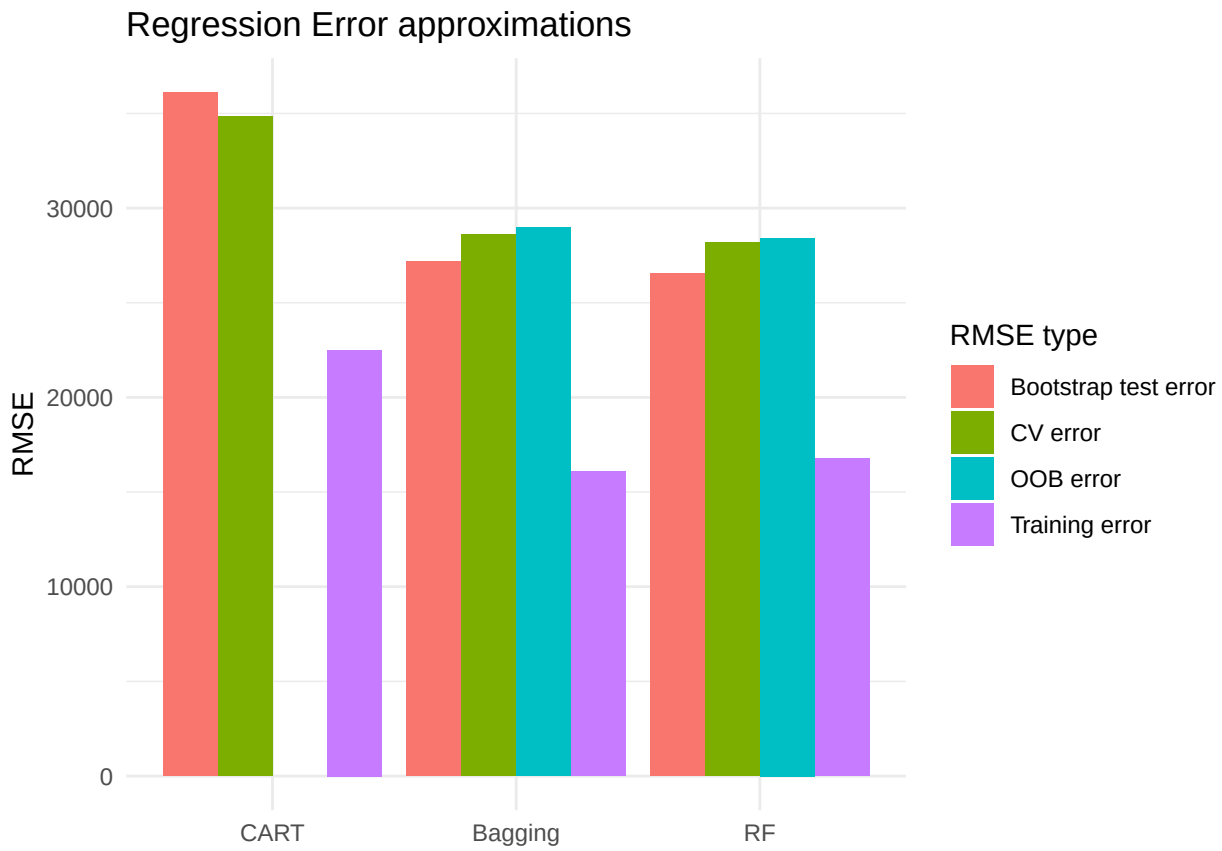
how well does CV and OOB approximate the error?

```
cart_res <- fit_resamples(cart_workflow,
                          control = control_resamples(save_workflow = TRUE),
                          metrics = metric_set(rmse),
                          resamples = ames_cv)

bag_res <- fit_resamples(bag_workflow,
                          control = control_resamples(save_workflow = TRUE),
                          metrics = metric_set(rmse),
                          resamples = ames_cv)

rf_res <- fit_resamples(rf_workflow,
                          control = control_resamples(save_workflow = TRUE),
                          metrics = metric_set(rmse),
```

```
                          resamples = ames_cv)

error_comp_df <- tibble(
  model = c("CART","Bagging","RF"),
  train_error = c(
    predict(cart_fit, new_data = ames_train) |>
      bind_cols(ames_train) |>
      rmse(truth = Sale_Price, estimate = .pred) |>
      pull(.estimate),
    predict(bag_fit, new_data = ames_train) |>
      bind_cols(ames_train) |>
      rmse(truth = Sale_Price, estimate = .pred) |>
      pull(.estimate),
    predict(rf_fit, new_data = ames_train) |>
      bind_cols(ames_train) |>
      rmse(truth = Sale_Price, estimate = .pred) |>
      pull(.estimate)
  ),
  cv_error = c(
    cart_res |>
      collect_metrics() |>
      pull(mean),
    bag_res |>
      collect_metrics() |>
      pull(mean),
    rf_res |>
      collect_metrics() |>
      pull(mean)
  ),
  oob_error = c(
    NA,
    bag_fit |>
      extract_fit_engine() |>
      pluck("prediction.error") |>
      sqrt(),
    rf_fit |>
      extract_fit_engine() |>
      pluck("prediction.error") |>
      sqrt()
  ),
  bootstrap_test_errorr = c(
    cart_tradeoff$rmse,
    bag_tradeoff$rmse,
    rf_tradeoff$rmse
  )
)

error_comp_df |>
  mutate(model = factor(model,levels=c("CART","Bagging","RF"))) |>
  pivot_longer(-model) |>
  ggplot(aes(x = model, y = value, fill = name)) +
  geom_bar(stat = "identity",
           position = "dodge") +
```

```
  scale_fill_discrete(name = "RMSE type",
                      labels = c("Bootstrap test error",
                                 "CV error",
                                 "OOB error",
                                 "Training error")) +
  labs(x = NULL,
       y = "RMSE",
       title = "Regression Error approximations") +
  theme_minimal()
```

```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_bar()`).
```




```
ggsave("output/reg_error_approx.jpg",
       width = 6.5, height = 3)
```

```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_bar()`).
```

classification

```
class_bootstrap <- function(workflow_in, train_data, test_data){
  fit_model <- function(split, workflow) {
    fit(workflow, data = training(split))
  }
```

```
  set.seed(2025)
  bootstrap_data <- bootstraps(train_data, times = 101)

  all_predictions <-
    bootstrap_data |>
    mutate(
      model_fit = map(splits, fit_model, workflow = workflow_in),
      predictions = map(model_fit, ~ predict(.x, new_data = test_data, type = "prob") |>
                          mutate(pred = if_else(.pred_good >= .pred_bad,"good","bad")))
    )

  prediction_df <-
    all_predictions |>
    select(id, predictions) |>
    unnest(predictions) |>
    group_by(id) |>
    mutate(Status = rep(test_data$Status, length.out = n()),
           obs_id = row_number()) |>
    ungroup()

  return(prediction_df)
}

bias_var_mse_class <- function(pred_df){
  mse_df <-
    pred_df |>
    mutate(status_int = if_else(Status == "good",1,0)) |>
    group_by(obs_id) |>
    summarise(
      mode_class = DescTools::Mode(pred),
      bias = mean(mode_class != Status),
      bias_2 = (mean(.pred_good) - mean(status_int))^2,
      variance = mean(pred != mode_class),
      variance_2 = stats::var(.pred_good),
      mse = mean((.pred_good - status_int)^2),
      loss = mean(pred != Status)
    ) |>
    ungroup() |>
    summarize(mean_bias = mean(bias),
              mean_var = mean(variance),
              mean_loss = mean(loss),
              mean_bias_2 = mean(bias_2),
              mean_var_2 = mean(variance_2),
              mean_mse = mean(mse),
              irr_error = mean_mse - mean_var_2 - mean_bias_2)

  return(mse_df)
}
```

Deep CART

as a baseline

```
set.seed(2025)
cart_mod <-
  decision_tree(mode = "classification",
                engine = "rpart",
                cost_complexity = 0,
                tree_depth = 30,
                min_n = 20)

class_recipe <-
  recipe(Status ~ ., data = credit_train)

cart_workflow <-
  workflow() |>
  add_model(cart_mod) |>
  add_recipe(class_recipe)

cart_fit <- fit(cart_workflow,
                data = credit_train)

cart_class_boot_preds <- class_bootstrap(cart_workflow, credit_train, credit_test)

cart_class_tradeoff <- bias_var_mse_class(cart_class_boot_preds)
```

Shallow CART

```
set.seed(2025)
cart_short_mod <-
  decision_tree(mode = "classification",
                engine = "rpart",
                cost_complexity = 0,
                tree_depth = 3,
                min_n = 20)

class_recipe <-
  recipe(Status ~ ., data = credit_train)

cart_short_workflow <-
  workflow() |>
  add_model(cart_short_mod) |>
  add_recipe(class_recipe)

cart_short_fit <- fit(cart_short_workflow,
                      data = credit_train)

cart_short_class_boot_preds <- class_bootstrap(cart_short_workflow, credit_train, credit_test)

cart_short_class_tradeoff <- bias_var_mse_class(cart_short_class_boot_preds)
```

Bagging

```
set.seed(2025)
bag_mod <-
```

```
  rand_forest(mode = "classification",
              engine = "ranger",
              mtry = ncol(credit_data) - 1,
              trees = 100,
              min_n = 20)  |>
  set_engine("ranger",
             probability = TRUE)

bag_workflow <-
  workflow() |>
  add_model(bag_mod) |>
  add_recipe(class_recipe)

bag_fit <-
  bag_workflow |>
  fit(data = credit_train)

bag_class_boot_preds <- class_bootstrap(bag_workflow, credit_train, credit_test)

bag_class_tradeoff <- bias_var_mse_class(bag_class_boot_preds)
```

RF

```
set.seed(2025)
rf_mod <-
  rand_forest(mode = "classification",
              engine = "ranger",
              mtry = floor(sqrt(ncol(credit_data) - 1)),
              trees = 100,
              min_n = 20)  |>
  set_engine("ranger",
             probability = TRUE)

class_recipe <-
  recipe(Status ~ ., data = credit_train)

rf_workflow <-
  workflow() |>
  add_model(rf_mod) |>
  add_recipe(class_recipe)

rf_fit <-
  rf_workflow |>
  fit(data = credit_train)

rf_class_boot_preds <- class_bootstrap(rf_workflow, credit_train, credit_test)

rf_class_tradeoff <- bias_var_mse_class(rf_class_boot_preds)
```

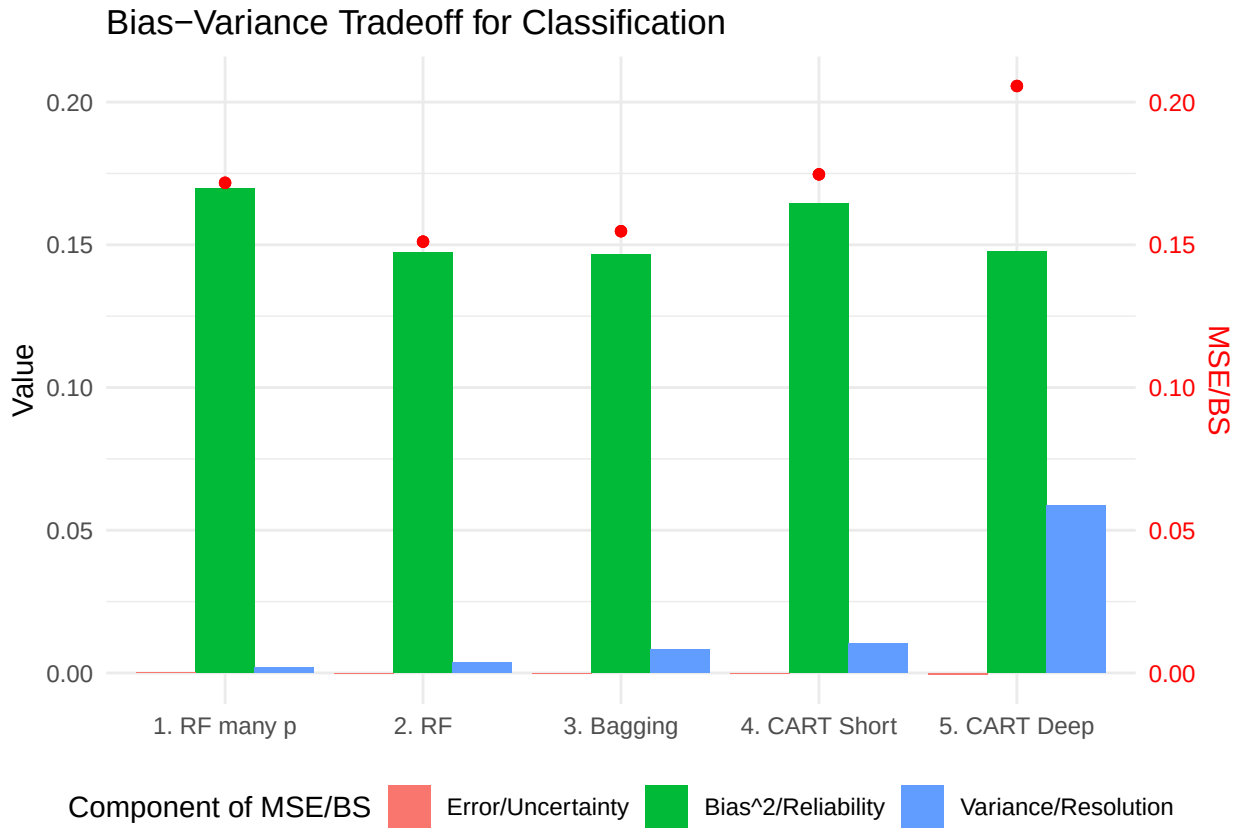
More complex - lots of predictors

```
set.seed(2025)
credit_train_rand <- credit_train |>
  bind_cols(vroom::gen_tbl(rows = nrow(credit_train),
                           cols = 50,
                           col_types = rep(times = 50,list("d"))))
credit_test_rand <- credit_test |>
  bind_cols(vroom::gen_tbl(rows = nrow(credit_test),
                           cols = 50,
                           col_types = rep(times = 50,list("d"))))
set.seed(2025)
rf_mod <-
  rand_forest(mode = "classification",
              engine = "ranger",
              mtry = floor(sqrt(ncol(credit_data) - 1)),
              trees = 100,
              min_n = 20)  |>
  set_engine("ranger",
             probability = TRUE)

class_recipe <-
  recipe(Status ~ ., data = credit_train_rand)

rf_p_workflow <-
  workflow() |>
  add_model(rf_mod) |>
  add_recipe(class_recipe)

rf_p_fit <-
  rf_p_workflow |>
  fit(data = credit_train_rand)

rf_small_class_boot_preds <- class_bootstrap(rf_p_workflow, credit_train_rand, credit_test_rand)

rf_small_class_tradeoff <- bias_var_mse_class(rf_small_class_boot_preds)
```

comparison

```
cart_short_class_tradeoff |>
  mutate(name = "4. CART Short") |>
  bind_rows(cart_class_tradeoff |>
              mutate(name = "5. CART Deep")) |>
  bind_rows(bag_class_tradeoff |>
              mutate(name = "3. Bagging")) |>
  bind_rows(rf_class_tradeoff |>
              mutate(name = "2. RF")) |>
  bind_rows(rf_small_class_tradeoff |>
              mutate(name = "1. RF many p")) |>
  pivot_longer(cols = c(mean_bias_2, mean_var_2, irr_error),
               names_to = "variable",
               values_to = "value") |>
  ggplot(aes(x = name)) +
  geom_bar(aes(y = value, fill = variable),
           position = "dodge", stat = "identity")+
```

```
  geom_point(aes(y = mean_mse),
             color = "red")  +
  scale_fill_discrete(name = "Component of MSE/BS",
                      labels = c("Error/Uncertainty",
                                 "Bias^2/Reliability",
                                 "Variance/Resolution")) +
  scale_y_continuous(sec.axis = sec_axis(transform = ~.,
                                         name = "MSE/BS")) +
  labs(x = NULL,
       y = "Value",
       title = "Bias-Variance Tradeoff for Classification") +
  theme_minimal() +
  theme(axis.text.y.right = element_text(color = "red"),
        axis.title.y.right = element_text(color = "red"),
        legend.position = "bottom")
```

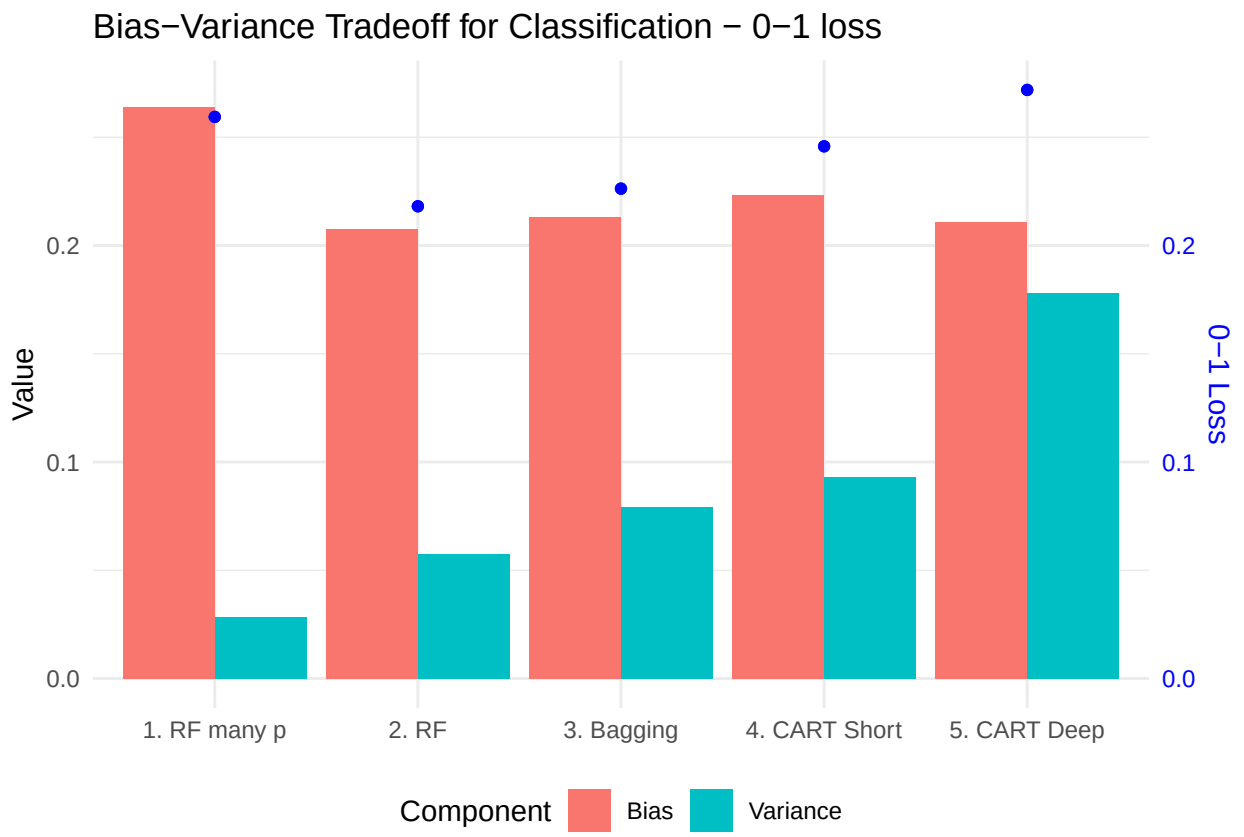



```
ggsave("output/class_bv_results.jpg",
       width = 6.5, height = 3)
```

with 0-1 loss

```
cart_short_class_tradeoff |>
  mutate(name = "4. CART Short") |>
  bind_rows(cart_class_tradeoff |>
              mutate(name = "5. CART Deep")) |>
  bind_rows(bag_class_tradeoff |>
              mutate(name = "3. Bagging")) |>
```

```r
bind_rows(rf_class_tradeoff |>
            mutate(name = "2. RF")) |>
bind_rows(rf_small_class_tradeoff |>
            mutate(name = "1. RF many p")) |>
pivot_longer(cols = c(mean_bias, mean_var),
             names_to = "variable",
             values_to = "value") |>
ggplot(aes(x = name)) +
geom_bar(aes(y = value, fill = variable),
         position = "dodge", stat = "identity")+
geom_point(aes(y = mean_loss),
           color = "blue")  +
scale_fill_discrete(name = "Component",
                    labels = c("Bias",
                               "Variance")) +
scale_y_continuous(sec.axis = sec_axis(transform = ~.,
                                       name = "0-1 Loss")) +
labs(x = NULL,
     y = "Value",
     title = "Bias-Variance Tradeoff for Classification - 0-1 loss") +
theme_minimal() +
theme(axis.text.y.right = element_text(color = "blue"),
      axis.title.y.right = element_text(color = "blue"),
      legend.position = "bottom")
```



```
ggsave("output/class_bv01_results.jpg",
       width = 6.5, height = 3)
```

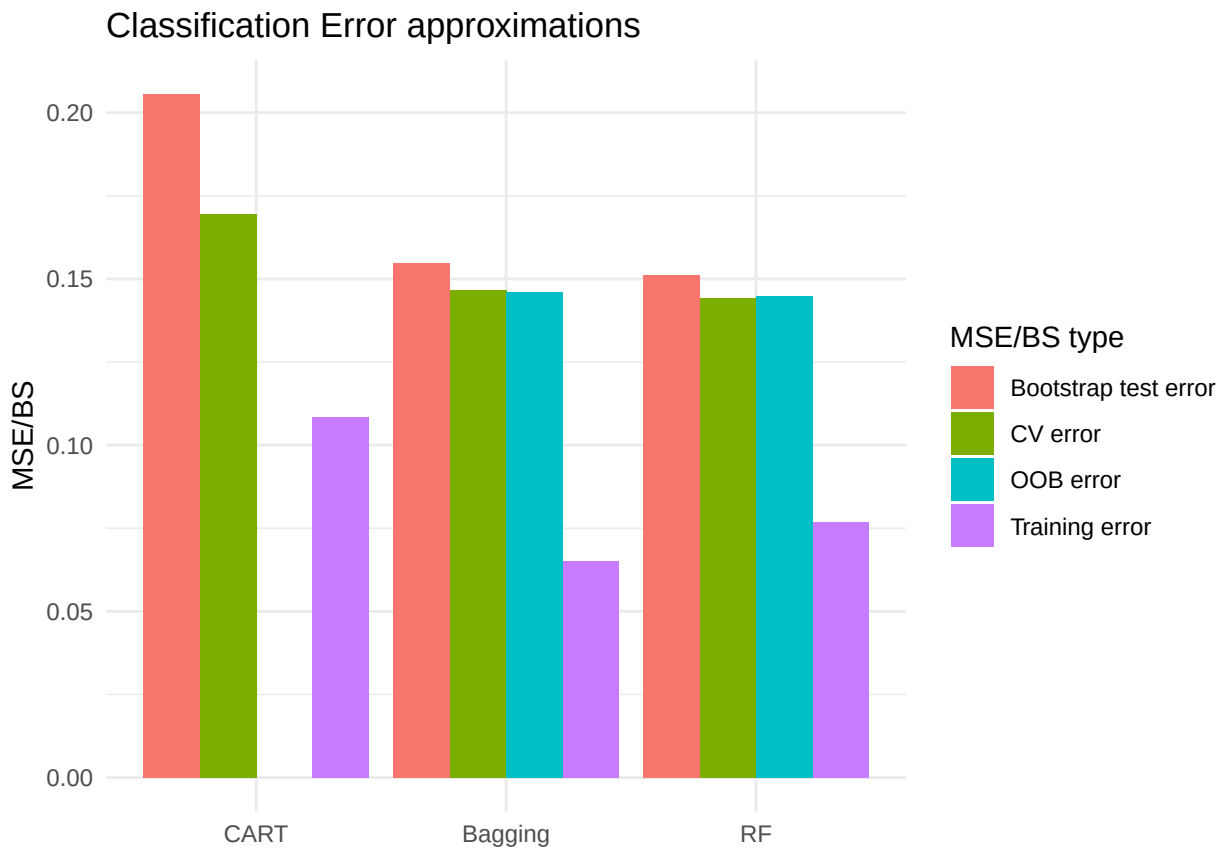
Bias is not changing much, all we are doing is smoothing out the variance

how well does CV and OOB approximate the error?

```
cart_res <- fit_resamples(cart_workflow,
                          control = control_resamples(save_workflow = TRUE),
                          metrics = metric_set(brier_class),
                          resamples = credit_cv)

bag_res <- fit_resamples(bag_workflow,
                          control = control_resamples(save_workflow = TRUE),
                          metrics = metric_set(brier_class),
                          resamples = credit_cv)

rf_res <- fit_resamples(rf_workflow,
                          control = control_resamples(save_workflow = TRUE),
                          metrics = metric_set(brier_class),
                          resamples = credit_cv)

class_error_df <- tibble(
  model = c("CART","Bagging","RF"),
   train_error = c(
    predict(cart_fit, new_data = credit_train, type = "prob") |>
      bind_cols(credit_train) |>
      brier_class(truth = Status,.pred_bad) |>
      pull(.estimate),
    predict(bag_fit, new_data = credit_train, type = "prob") |>
      bind_cols(credit_train) |>
      brier_class(truth = Status,.pred_bad) |>
      pull(.estimate),
   predict(rf_fit, new_data = credit_train, type = "prob") |>
      bind_cols(credit_train) |>
      brier_class(truth = Status,.pred_bad) |>
      pull(.estimate)
  ),
  cv_error = c(
    cart_res |>
      collect_metrics() |>
      pull(mean),
    bag_res |>
      collect_metrics() |>
      pull(mean),
    rf_res |>
      collect_metrics() |>
      pull(mean)
  ),
  oob_error = c(
    NA,
    bag_fit |>
      extract_fit_engine() |>
```

```
      pluck("prediction.error") ,
    rf_fit |>
      extract_fit_engine() |>
      pluck("prediction.error")
  ),
  bootstrap_test_errorr = c(
    cart_class_tradeoff$mean_mse,
    bag_class_tradeoff$mean_mse,
    rf_class_tradeoff$mean_mse
  )
)

class_error_df |>
  mutate(model = factor(model,levels=c("CART","Bagging","RF"))) |>
  pivot_longer(-model) |>
  ggplot(aes(x = model, y = value, fill = name)) +
  geom_bar(stat = "identity",
           position = "dodge") +
  scale_fill_discrete(name = "MSE/BS type",
                      labels = c("Bootstrap test error",
                                 "CV error",
                                 "OOB error",
                                 "Training error")) +
  labs(x = NULL,
       y = "MSE/BS",
       title = "Classification Error approximations") +
  theme_minimal()
```

```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_bar()`).
```




```
ggsave("output/class_error_approx.jpg",
       width = 6.5, height = 3)
```

```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_bar()`).
```

Consensus version of bagging and rf

```
class_bootstrap_cons <- function(workflow_in, train_data, test_data){
  fit_model <- function(split, workflow) {
    fit(workflow, data = training(split))
  }

  set.seed(2025)
  bootstrap_data <- bootstraps(train_data, times = 101)

  all_predictions <-
    bootstrap_data |>
    mutate(
      model_fit = map(splits, fit_model, workflow = workflow_in),
      predictions = map(model_fit, ~ predict(.x, new_data = test_data, type = "class"))
    )

  prediction_df <-
    all_predictions |>
    select(id, predictions) |>
```

```
    unnest(predictions) |>
    group_by(id) |>
    mutate(Status = rep(test_data$Status, length.out = n()),
           obs_id = row_number()) |>
    ungroup()

  return(prediction_df)
}
```

```
set.seed(2025)
bag_mod <-
  rand_forest(mode = "classification",
              engine = "ranger",
              mtry = ncol(credit_data) - 1,
              trees = 100,
              min_n = 20)  |>
  set_engine("ranger",
             probability = FALSE)

class_recipe <-
  recipe(Status ~ ., data = credit_train)

bag_workflow <-
  workflow() |>
  add_model(bag_mod) |>
  add_recipe(class_recipe)

bag_cons_fit <-
  bag_workflow |>
  fit(data = credit_train)

bag_consensus_boot_preds <- class_bootstrap_cons(bag_workflow,credit_train, credit_test)

bag_consensus_tradeoff <- bag_consensus_boot_preds |>
  group_by(obs_id) |>
  summarise(
    mode_class = DescTools::Mode(.pred_class),
    bias = mean(mode_class != Status),
    variance = mean(.pred_class != mode_class),
    loss = mean(.pred_class != Status)
  ) |>
  ungroup() |>
  summarize(mean_bias = mean(bias),
            mean_var = mean(variance),
            mean_loss = mean(loss))
```

```
set.seed(2025)
rf_mod <-
  rand_forest(mode = "classification",
              engine = "ranger",
              mtry = floor(sqrt(ncol(credit_data) - 1)),
              trees = 100,
              min_n = 20)  |>
  set_engine("ranger",
```

```
              probability = FALSE)

rf_workflow <-
  workflow() |>
  add_model(rf_mod) |>
  add_recipe(class_recipe)

rf_cons_fit <-
  rf_workflow |>
  fit(data = credit_train)

rf_consensus_boot_preds <-class_bootstrap_cons(rf_cons_fit,credit_train, credit_test)

rf_consensus_tradeoff <- rf_consensus_boot_preds |>
  group_by(obs_id) |>
  summarise(
    mode_class = DescTools::Mode(.pred_class),
    bias = mean(mode_class != Status),
    variance = mean(.pred_class != mode_class),
    loss = mean(.pred_class != Status)
  ) |>
  ungroup() |>
  summarize(mean_bias = mean(bias),
            mean_var = mean(variance),
            mean_loss = mean(loss))
```

comparison with probability version