Simulation of Bias-Variance Tradeoff with Tree-Based Models

P9120 Final Project

Ravi Brenner

1 Introduction

The bias-variance tradeoff is a common framework for understanding the error of a prediction model in statistics and machine learning. Once a model is trained on some training data and evaluated on some test data, the mean square error (MSE) can be calculated, which is composed of the bias and the variance in the following way:

$MSE(\hat{f}(x)) = bias^2(f(x), \hat{f}(x)) + var(\hat{f}(x))$

The goal of most modeling pipelines is to minimize this MSE in some way, either by reducing the variance for simpler models, or reducing the bias for more complicated models. This is necessary because bias and variance have a fundamentally inverse relationship. Take an extremely simple model as an example, line a linear regression with only an intercept term. This model will make the same prediction $\hat{f}(x)$ for all test data points x. Thus the *variance* of the predictions will be low. At the same time, the difference between the predictions and the true values will be quite high, so we say the *bias* is high. On the other end of the spectrum, think of a very complex model like a neural network. The network can be trained to exactly predict every one of the training data points, so the variance of the predictions will be high. Meanwhile the difference between the predictions and the true values will be relatively low, so the bias will be low.

Many examples of the bias-variance tradeoff treat this as mainly a theoretical construct, which it is. While the MSE can be mathematically decomposed into bias and variance terms, there are also error terms that are not accounted for. Furthermore, most textbook examples rely on simulated data in the regression setting only. Here, I will demonstrate the bias-variance tradeoff using real datasets, and apply it to both regression and classification. I will use various types of tree-based and ensemble models to do so, since these are relatively easy to understand but provide enough flexibility in their settings to demonstrate the changes in bias and variance. Finally, I will explore how well the MSE can be estimated by cross validation and out-of-bag error.

2 Methods

All analysis was conducted in R software using the `tidymodels` framework. Full code is available on the github repo here. I used the Ames housing data for regression tasks, and the Credit data for classification tasks, both provided as part of the `modeldata` subpackage of `tidymodels`. The workflow for regression and classification was similar, although a few key differences are discussed here. In brief, various models of varying complexity were fit, and their error, bias, and variance calculated via boostrapping.

For regression, the data were split with 80% going to training and 20% to testing. Five models were fit: a deep CART tree, a shallow CART tree, a bagging model, a typical random forest model with the number of predictors = p/3, and random forest model with just 3 randomly chosen predictors. For all models, the minimum number of data points for a split was 20. The maximum depth of the deep tree was 30, and for the shallow tree was 3. For bagging and RF models, 100 trees were created and averaged.

For each model, 100 bootstrap samples of the training data were drawn, the model was trained on the 100 different training datasets, and then it was evaluated on the test set. This yielded 100 predictions for each test subject with which to calculate the bias^2, variance, and MSE. The bias^2 was calculated as the square of the difference between mean prediction for each subject and the subjects true value. The variance was calculated as the mean of the squared difference between each prediction and each subject's average prediction. The MSE was calculated as the mean of the square of the difference between each prediction and the true value. The irreducible error was also calculated as MSE - biasˆ2 - variance. In formula terms:

$$E[(\hat{f}(x) - f(x))^2] = (E[\hat{f}(x)] - f(x))^2 + E[(\hat{f}(x) - E[\hat{f}(x)])^2] + \epsilon$$

Note how the variance has no relationship to the true predictions $\hat{f}(x)$. It simply gives the variance of the predictions themselves.

As a secondary analysis, 10-fold cross validation was performed to compare the CV error and the OOB error from bagging and random forests with the test error estimated via bootstrapping described above.

For classification, some modifications were needed. The data were again split with 80% going to training and 20% going to testing, and 5 models were again fit. There were a few differences with the models. First for the typical random forest model, the number of randomly chosen predictors was $\sqrt{p}$. Second, because $\sqrt{p}$ was already quite small for the credit data, random non-informative predictors were added to the 5th random forest model (instead of using 3 randomly chosen predictors). Third, for the boostrapping procedure 101 replications were used instead of 100, to ensure no ties when calculating the modal prediction for a given individual. For the bagging and random forest models, probability averaging was used instead of consensus voting. The two methods generally give very similar results, but probability averaging has the advantage of allowing us to calculate the Brier score for all models.

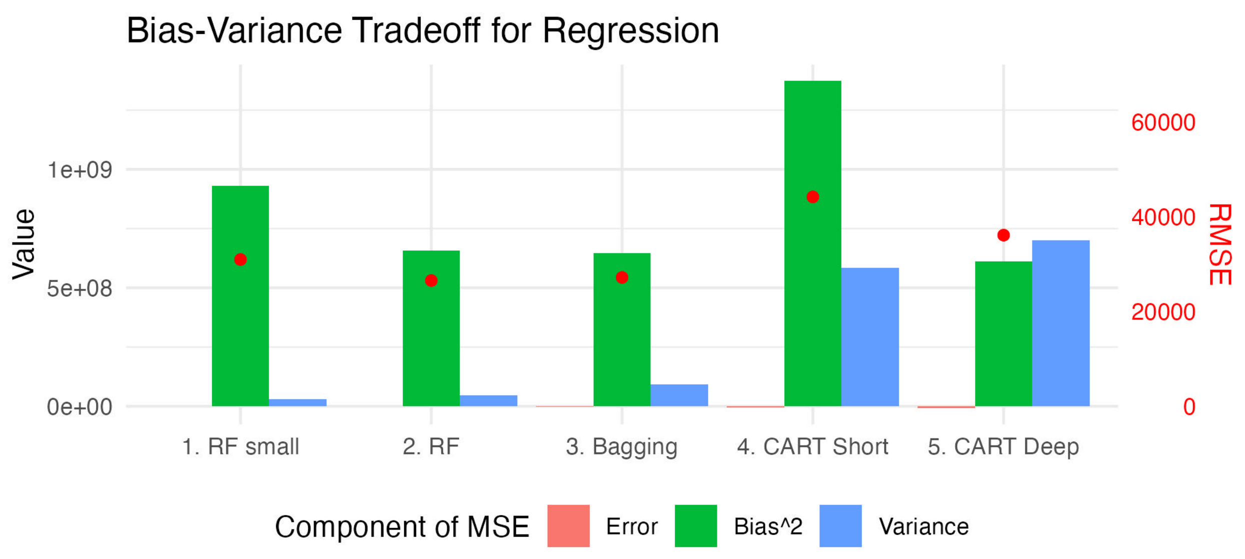
The Brier score has the same form as the MSE in regression, treating the probabilities as predictions and the outcomes as numeric 0 and 1. This further allows for decomposition of the MSE into "bias" and "variance" terms. Technically, these terms represent reliability and resolution, but the math works out to be identical to the regression case. As a sensitivity analysis, the 0-1 loss (misclassification error), bias, and variance can be calculated as follows. The loss for a given subject is the average number of times the predicted class does not equal the true class. The bias is a 1 if the modal class is not the true class, and 0 otherwise. The variance is the average number of times the predicted class does not equal the modal class. The results for this loss setup are also presented, but the values do not decompose as nicely–the bias and variance do not add up to the loss.

As a secondary analysis, 10-fold cross validation was performed to compare the CV error and the OOB error from bagging and random forests with the test error estimated via bootstrapping described above.

3 Results

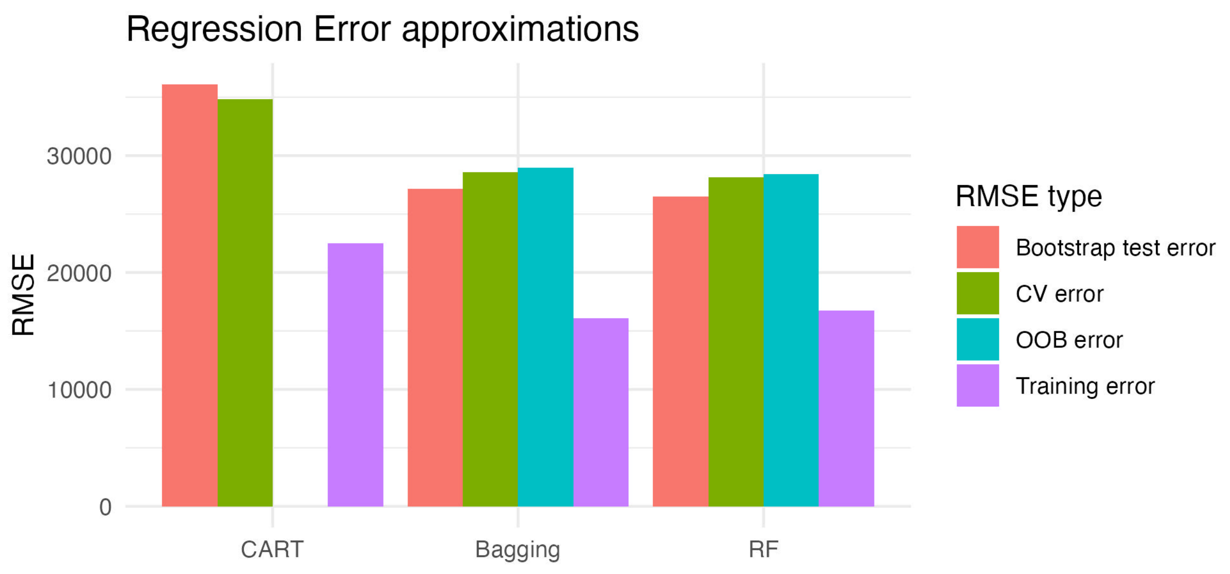
3.1 Regression - Bias Variance Tradeoff

Results for the regression case are shown in the below figure, sorted from left to right by increasing variance. The green bars show the biasˆ2, the blue bars show the variance, and the red data points show the RMSE (the red bars representing the error terms are very small). The two CART trees show an example fo the bias-variance tradeoff The short regression tree (CART Short) has very high bias. The deep regression tree (CART deep) has much lower bias, and just slightly higher variance, resulting in a significant reduction in MSE. The deep CART tree is much more "flexible" than the short one. Bagging, which takes the average predictions of 100 deep trees, has similar bias to the deep regression tree, but much lower variance as expected from theory. Random forests modestly improves on this reduction in variance too, albeit with a slight increase in bias. Still, the MSE is lower than for bagging. Decreasing the number of randomly chosen predictors further, as in the small random forest (RF small) leads to even further reduction in variance–but now a substantial jump in bias leading to higher MSE. Note that the component trees in bagging use all predictors, and are therefore more flexible, than those in random forests.



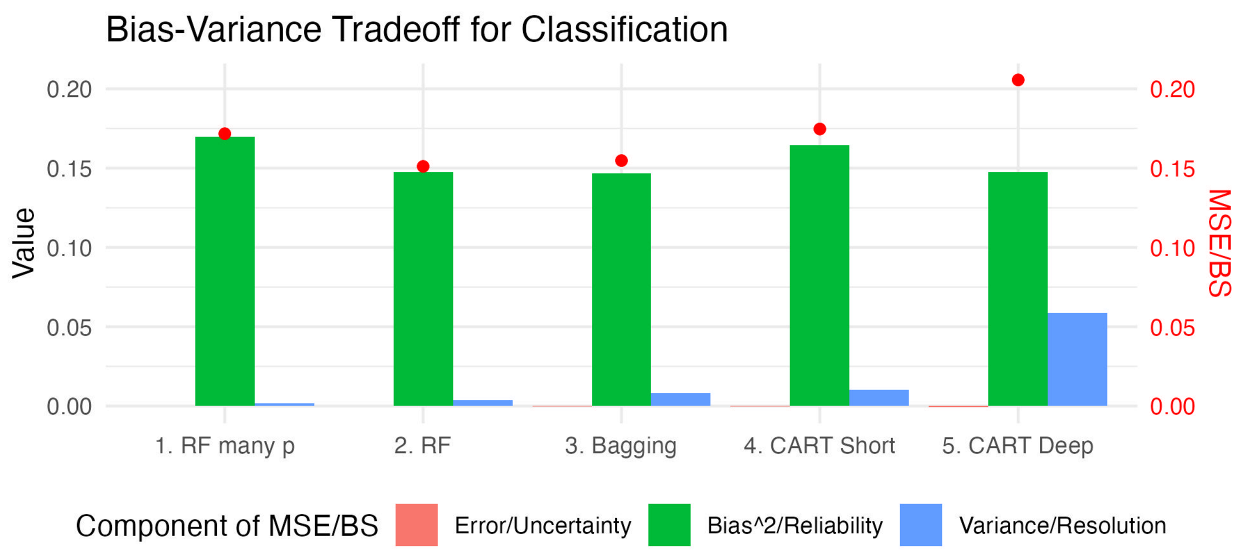
3.2 Regression - Training approximations (CV and OOB error)

The out-of-bag error (in blue; for bagging and RF) and cross validation error (in green; for all models) is shown below for the CART, bagging, and RF models fit earlier, along with the training error (in purple). Clearly, the training error does a poor job approximating the test error. Here, you can see that both cross validation and OOB error come fairly close to the error estimated from bootstrapping. The CV error is slightly lower than the bootstrap test error for CART, while the CV and OOB error is higher for bagging and RF. CV error also appears to be closer to the test error than OOB error.

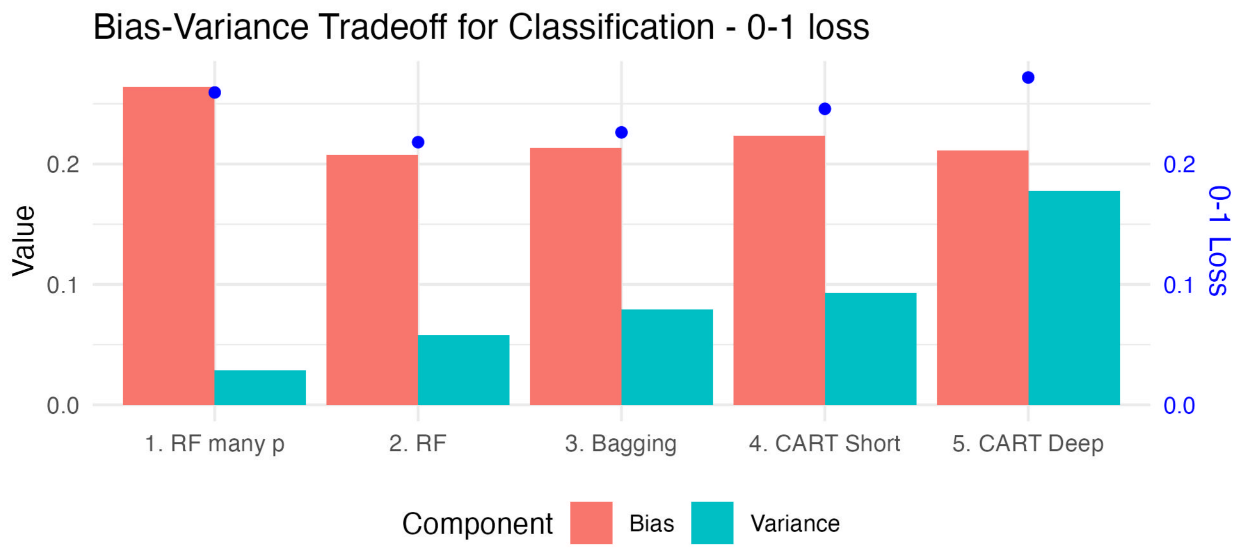


3.3 Classification - Bias Variance Tradeoff

Results for the classification case are shown in the below figure. The green bars show the bias^2, the blue bars show the variance, and the red data points show the MSE/Brier Score (the red bars representing the error terms are very small). The deep regression tree (CART deep) has high variance. The short regression tree (CART short) has much lower variance, and just slightly higher bias, resulting in a significant reduction in MSE. Bagging, which takes the average predictions of 100 deep trees, has similar bias to the deep regression tree, but much lower variance as expected from theory. Random forests modestly improved on this reduction in variance too, with a slight increase in bias. Adding many noninformative predictors to the data, as in the RF with many p, leads to even further reduction in variance, but with a substantial jump in bias and therefore MSE. Again note the tradeoffs in flexibility. Bagging uses all (informative) predictors, and is therefore more flexible than random forests which only relies on a subset of predictors.

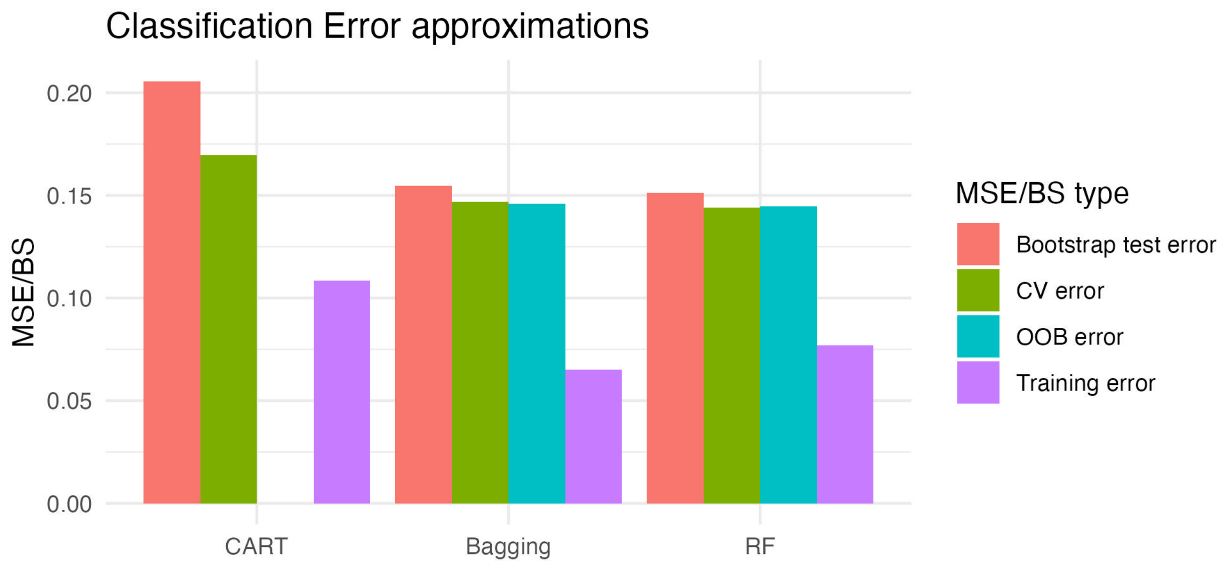


Qualitatively similar results for the 0-1 loss (misclassification error) are shown in the below figure. You can see that variance goes up from left to right, while bias goes down, resulting in the characteristic "U" shape in the loss function (blue points).



3.4 Classification - Training approximations (CV and OOB error)

The out-of-bag MSE/Brier Score (for bagging and RF) and cross validation error (for all) is shown below for the CART, bagging, and RF models fit earlier, along with the training error. Again, the training error does a poor job approximating the test error. Both cross validation and OOB error come fairly close to the error estimated from bootstrapping, with not much difference between CV and OOB error.



4 Discussion

Classification and regression trees are easy to understand, but their simplicity means they have high variance. Averaging many such trees together as in bagging leads to a substantial reduction in variance. Random forests increases the complexity further by randomly choosing a fraction of the predictions for each bootstrap split, to decorrelate the trees. Taking this too far, however, by making this number very small or increasing the number of uninformative predictors can end up increasing bias. Hence a middle ground of complexity is more optimal. This is how you end up with the "U" shaped MSE curve in most textbook examples, with bias decreasing and variance increasing with increasing model complexity.

The traditional bias-variance-MSE curve depicted in most places has "model complexity" as a theoretical construct on the x-axis. A major takeaway of this analysis is that, in practice, the more "complex" models here like random forests actually had lower, not higher variance. This is because "complex" does not quite capture what is happening; the notion of "complexity" should be taken in context of the overall modeling process at hand. A fully grown deep CART tree is more complex and flexible than a shallow CART tree. Similarly, despite the more complicated procedure, the component trees in random forests use fewer predictors than the full set of predictors, making them less flexible compared to the trees in bagging. Thus ensemble methods like bagging and random forests may not fit nicely on a scale of increasing "complexity", since they are just averaging less complex predictors. While we did not address gradient boosting, such algorithms take advantage of the U-shape to descend the error gradient and reach the point with lowest error. Their constituent learners are typically very simple, with high bias and low variance. Yet the models as a whole can perform better than more complex trees alone. Likewise with random forests: the individual trees are even simpler than the fully grown CART trees since they only use a subset of predictors, and then averaging reduces the variance, leading to lower MSE. Depending on how you choose to think about it, this is either more complex (bootstrapping, random sampling of predictors, averaging) or less complex (simpler trees with fewer predictors) than a single fully grown CART tree. Hence some nuance is needed, and considering the base learner may be a more intuitive way of placing the given model on the bias-variance tradeoff curve.

Cross validation is a good, easily applicable way to evaluate models, precisely because it does a good job approximating the test error. If you just looked at training error, you would constantly underestimate the test error. This is because you could get training error super low by increasing complexity of your model. Most modeling pipelines take this into account, and you always optimize the model parameters to minimize the MSE; that's the whole point of model tuning and training. Understanding the reasons for this are important.

Overall, there are some limitations of this analysis. Ultimately, we are still relying on simulation via bootstrapping. So even though we are trying to simulate the tradeoff using real datasets, there is an element of approximation. Additionally, this analysis was limited to relatively simple tree-based models. The ensemble methods used are mainly effective at reducing variance, but don't much impact bias. The traditional bias-variance tradeoff curve depicted in most textbooks, with variance increasing and bias decreasing with increasing model complexity was not exactly reproduced here, because the ensemble methods don't reduce bias much. Still you can see nicely how the bias and variance decomposes with various tree based models, and the impact of bagging and random forests on reducing that variance. You can easily see, within each class of models (CART vs. bagging + RF), the bias-variance tradeoff taking place. In conclusion, understanding the bias-variance tradeoff is crucial to the machine learning modeling process. Ensemble tree-based methods are some of the most popular tools for machine learning, hence understanding how they fit in to the bias-variance tradeoff is valuable for any statistician.

Ensemble Trees

Ravi Brenner

```r
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 error
```

```r
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()
```

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

```r
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) |>
```

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

Bagging

```r
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

```r
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))
```

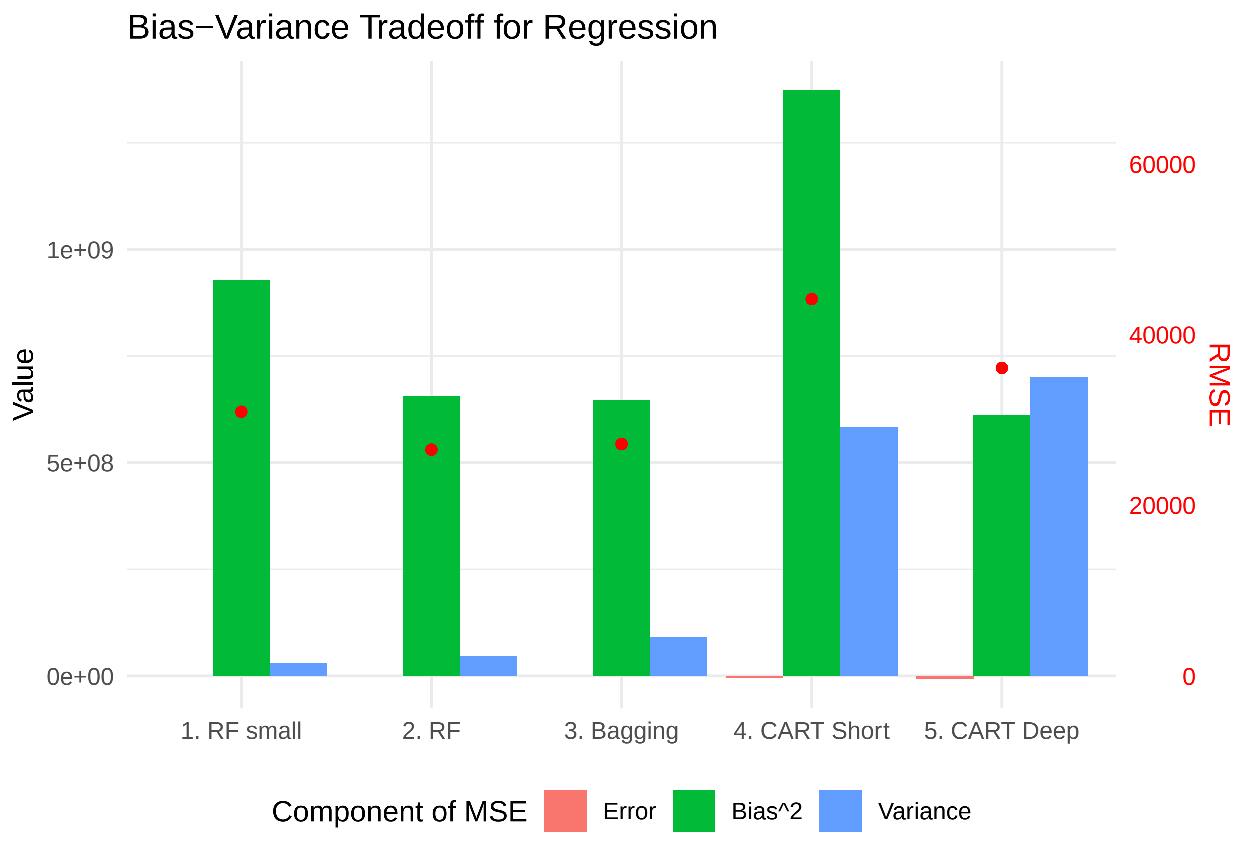
Too complex s.t. bias starts to increase more and mse starts to increase tooo

i think rf with a smaller mtry would work

```r
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

```r
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")
```

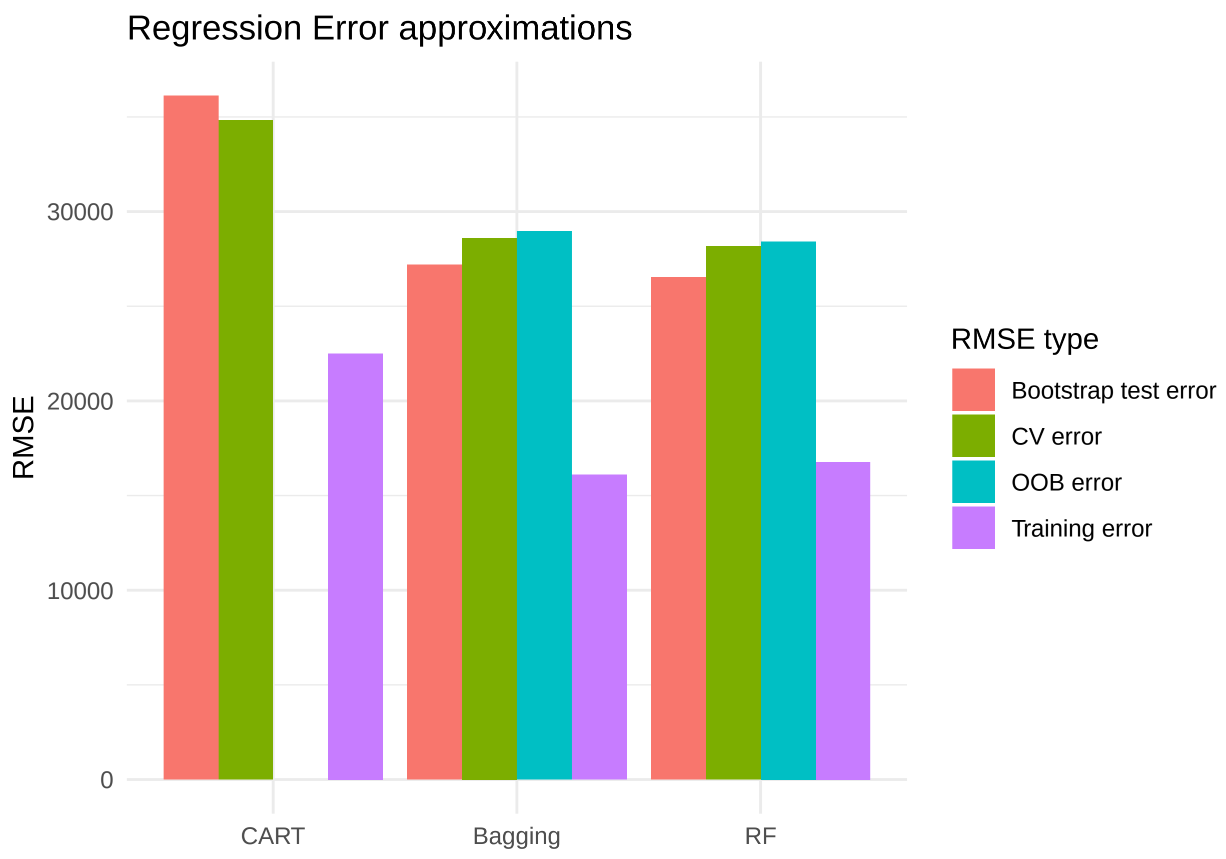


```r
ggsave("output/reg_bv_results.jpg",
       width = 6.5, height = 3)
```

how well does CV and OOB approximate the error?

```r
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()`).
```



```r
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

```r
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

```r
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

```r
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

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

```r
  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

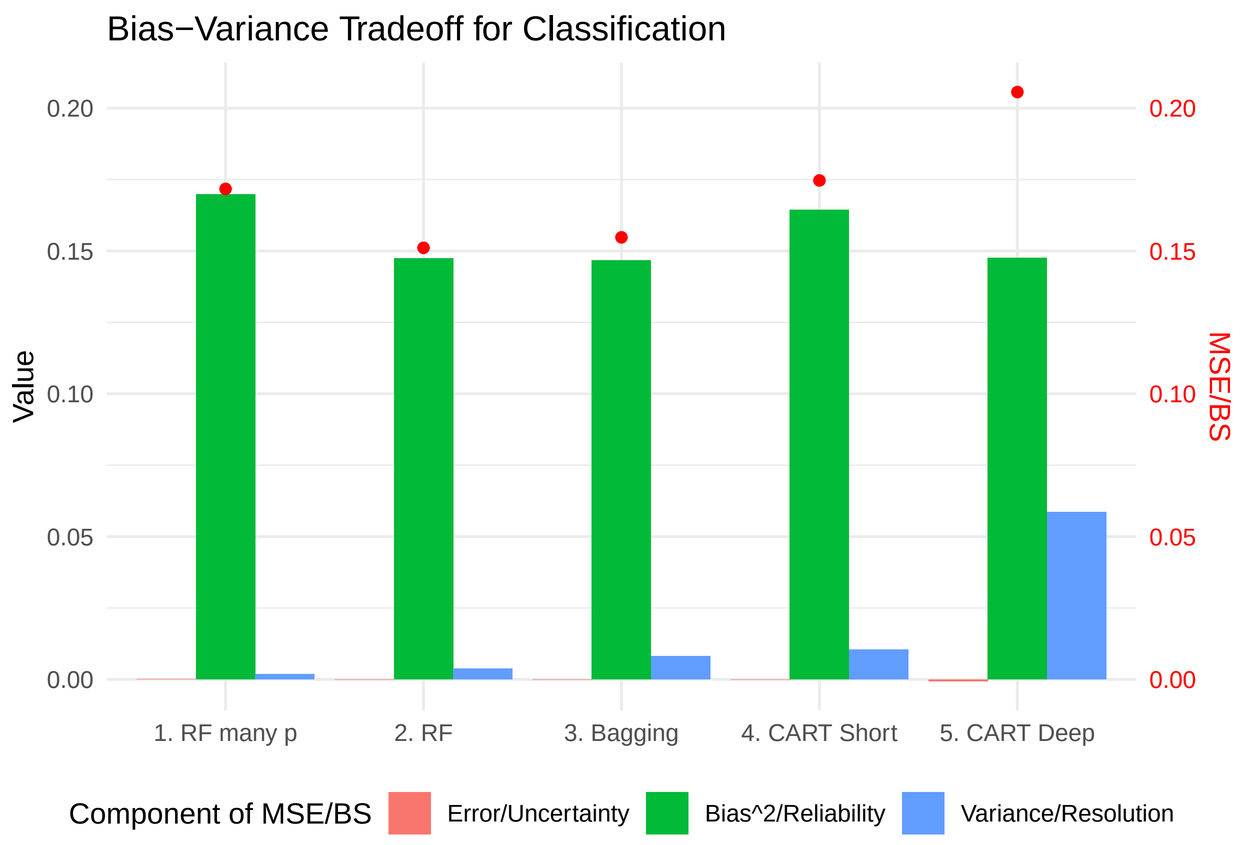
```r
set.seed(2025)
rf_mod <-
  rand_forest(mode = "classification",
              engine = "ranger",
              mtry = floor(sqrt(ncol(credit_data) - 1)),
              trees = 100,
              min_n = 20)  |>
  set_engine("ranger",
             probability = TRUE)

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

```r
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

```r
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")
```

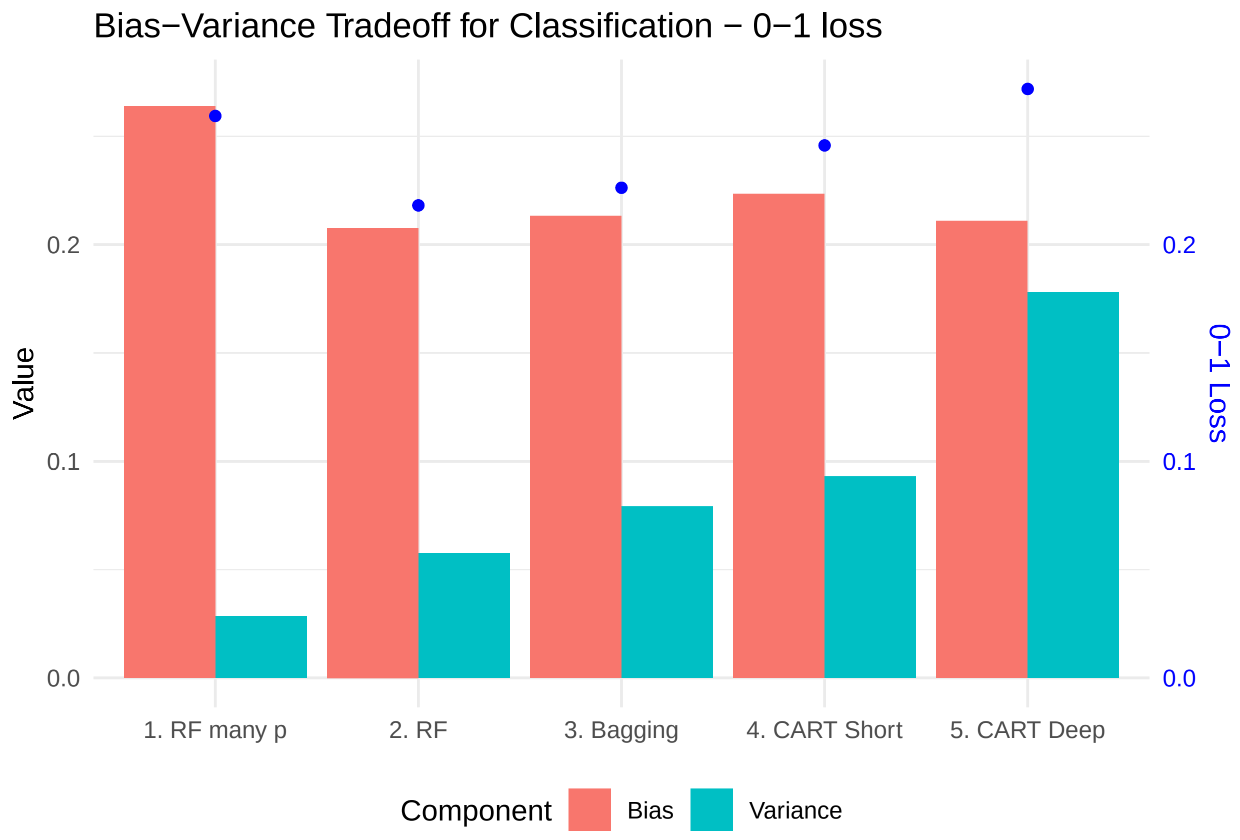


```r
ggsave("output/class_bv_results.jpg",
       width = 6.5, height = 3)
```

with 0-1 loss

```r
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")
```



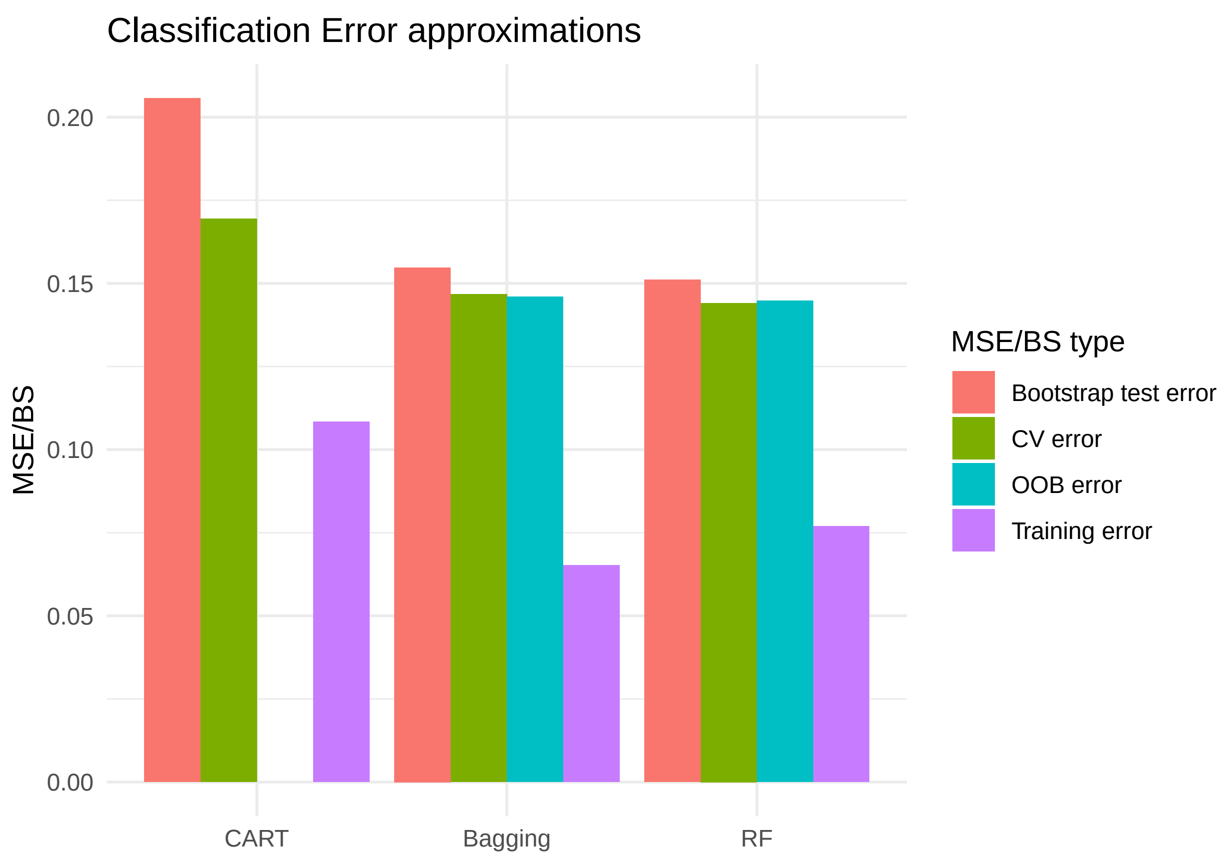
```r
ggsave("output/class_bv01_results.jpg",
       width = 6.5, height = 3)
```

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

how well does CV and OOB approximate the error?

```r
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()`).
```



```r
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

```r
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