Tree-based methods

Saniya Bekova

2024-12-01

1.1. Variable choice

From previous assignment bonus task, Lasso left these variables: hh_income, acres_plots, bicycles, basic_cell_phones,yrs_in_mkt, profit, customers_pr_day

I will add some categorical variables with few levels:

female with 2 levels, married with 2 levels, pay_even_disagree with 2 levels

Importing libraries and loading data

```
# Loading necessary libraries
library(dplyr)
library(tidymodels)

# Loading dataset
load("data/vendor_data.RData")

vendor_data <- vendor_data |>
  mutate(
         recent_receipt_7 = as.factor(recent_receipt_7),
         female = as.factor(female),
         married = as.factor(married),
         pay_even_disagree = as.factor(pay_even_disagree))
str(vendor_data)
```

```
tibble [2,531 x 34] (S3: tbl_df/tbl/data.frame)
 $ market           : num [1:2531] 1 1 1 1 1 1 1 1 1 1 ...
 $ district         : num [1:2531] 1 1 1 1 1 1 1 1 1 1 ...
```

```
$ language         : Factor w/ 4 levels "Chichewa","English",..: 1 1 1 1 1 1 1 1 1 1 ...
 ..- attr(*, "label")= chr "What is the primary language this survey will be conducted in?"
$ female           : Factor w/ 2 levels "0","1": 2 1 1 1 2 1 2 1 1 2 ...
$ age              : num [1:2531] 32 64 31 50 24 26 27 33 46 33 ...
 ..- attr(*, "label")= chr "How old are you?"
$ tribe            : Factor w/ 7 levels "Chewa","Lomwe",..: 2 4 2 4 3 1 4 4 4 2 ...
 ..- attr(*, "label")= chr "What is your tribe?"
$ married          : Factor w/ 2 levels "0","1": 2 2 1 2 1 2 2 2 2 2 ...
$ education        : Factor w/ 18 levels "None","Nursery School",..: 14 12 10 10 14 14 9 11
$ literacy         : Factor w/ 4 levels "Could not read",..: 4 4 4 4 4 4 4 4 4 4 ...
$ reading_language : Factor w/ 3 levels "Chichewa","English",..: 2 2 1 2 1 1 1 1 2 1 ...
 ..- attr(*, "label")= chr "Can you read this card for me?"
$ hh_income        : num [1:2531] 100000 70000 80000 150000 50000 47000 50000 70000 60000 1(
 ..- attr(*, "label")= chr "What is your estimated total household monthly income? In other
$ houses           : num [1:2531] 0 1 0 0 3 0 1 1 1 3 ...
 ..- attr(*, "label")= chr "Houses"
$ acres_farmland   : num [1:2531] 0.75 1 1.5 5 3 0 0 2 3 0 ...
 ..- attr(*, "label")= chr "Acres of Farmland"
$ acres_plots      : num [1:2531] 0 0 0 0.5 0 0 0 1 0 0 ...
 ..- attr(*, "label")= chr "Acres of Undeveloped Plots"
$ bicycles         : num [1:2531] 1 1 1 2 1 0 3 1 1 2 ...
 ..- attr(*, "label")= chr "Bicycles"
$ chickens         : num [1:2531] 0 0 0 20 9 0 4 0 0 0 ...
 ..- attr(*, "label")= chr "Chickens"
$ goats            : num [1:2531] 0 0 0 10 2 0 0 7 0 0 ...
 ..- attr(*, "label")= chr "Goats"
$ basic_cell_phones: num [1:2531] 0 0 0 2 2 1 2 2 2 0 ...
 ..- attr(*, "label")= chr "Basic Cell Phones"
$ smart_phones     : num [1:2531] 1 1 1 0 0 1 0 0 0 0 ...
 ..- attr(*, "label")= chr "Smart Phones"
$ days_pr_week     : num [1:2531] 6 5 6 7 6 5 6 7 7 6 ...
$ service          : num [1:2531] 0 0 0 0 0 1 0 0 0 0 ...
$ yrs_in_mkt       : num [1:2531] 8 5 2 22 5 2 2 15 18 5 ...
$ profit           : num [1:2531] 333 1000 3500 7143 200 ...
$ customers_pr_day : num [1:2531] 5 5 50 50 2 1 10 15 35 10 ...
 ..- attr(*, "label")= chr "On an average day as a vendor at this market, how many customers
$ stall_type       : Factor w/ 5 levels "Tarp, blanket or baskets on the ground",..: 5 5 3 1
 ..- attr(*, "label")= chr "Do NOT read: What type of stall is this?"
$ any_constr       : num [1:2531] 0 0 0 0 0 0 0 0 0 0 ...
$ trust_dist_gov   : Factor w/ 4 levels "Not at all trustworthy",..: 3 1 3 2 3 1 2 2 4 3 ..
$ trust_ward_clr   : Factor w/ 4 levels "Not at all trustworthy",..: 4 1 2 2 3 1 2 2 2 2 ..
$ satisfaction_dev : Factor w/ 4 levels "Very Dissatisfied",..: 4 2 3 3 1 3 3 1 4 3 ...
$ pay_even_disagree: Factor w/ 2 levels "0","1": 2 2 2 2 2 1 1 2 2 2 ...
```

```
$ tax_duty         : Factor w/ 4 levels "Strongly Disagree",..: 2 4 4 4 4 4 4 4 4 4 ...
$ vote_intend      : num [1:2531] 1 1 1 1 0 1 1 1 1 1 ...
$ recent_receipt_7 : Factor w/ 2 levels "0","1": 2 2 2 2 2 1 1 1 2 1 ...
$ test             : num [1:2531] 0 0 0 0 1 0 0 1 0 0 ...
```

```
summary(vendor_data)
```

```
    market          district         language    female        age
Min.   :  1.00   Min.   :1.000   Chichewa:2431   0:1749   Min.   :18.00
1st Qu.: 33.00   1st Qu.:3.000   English :   0   1: 782   1st Qu.:27.00
Median : 65.00   Median :5.000   Tumbuka : 100           Median :34.00
Mean   : 64.73   Mean   :4.714   Yawo    :   0           Mean   :34.82
3rd Qu.: 97.00   3rd Qu.:7.000                           3rd Qu.:41.00
Max.   :128.00   Max.   :8.000                           Max.   :85.00
                                                         NA's   :1
    tribe     married        education
Lomwe  :731   0: 489   MSCE/Form 4:403
Chewa  :652   1:2042   Standard 8 :388
Yao    :486            Standard 7 :306
Tumbuka:295            JCE/Form 2 :237
Ngoni  :230            Standard 6 :194
(Other): 54            (Other)    :999
NA's   : 83            NA's       :  4
                                    literacy          reading_language
Could not read                          : 268   Chichewa      :1475
Could read some of the card             : 121   English       : 800
Could read the whole card with difficulty: 305   Could not read: 256
Could read the whole card with ease     :1822
NA's                                    :  15
```

```
  hh_income          houses         acres_farmland    acres_plots
Min.   :      2   Min.   :    0.00   Min.   : 0.000   Min.   : 0.0000
1st Qu.:  30000   1st Qu.:    1.00   1st Qu.: 0.000   1st Qu.: 0.0000
Median :  50000   Median :    1.00   Median : 1.000   Median : 0.0000
Mean   :  94384   Mean   :   32.85   Mean   : 1.736   Mean   : 0.3151
3rd Qu.: 100000   3rd Qu.:    1.00   3rd Qu.: 2.000   3rd Qu.: 0.0000
Max.   :2700000   Max.   :80000.00   Max.   :80.000   Max.   :40.0000
NA's   :26        NA's   :1          NA's   :1        NA's   :3
   bicycles          chickens          goats          basic_cell_phones
Min.   : 0.0000   Min.   :  0.000   Min.   :  0.000   Min.   :0.000
1st Qu.: 0.0000   1st Qu.:  0.000   1st Qu.:  0.000   1st Qu.:1.000
```

```
Median : 1.0000   Median :  0.000   Median :  0.000   Median :1.000
Mean   : 0.9593   Mean   :  3.928   Mean   :  1.254   Mean   :1.194
3rd Qu.: 1.0000   3rd Qu.:  5.000   3rd Qu.:  1.000   3rd Qu.:2.000
Max.   :20.0000   Max.   :100.000   Max.   :108.000   Max.   :8.000
NA's   :1         NA's   :10        NA's   :4         NA's   :1
 smart_phones     days_pr_week      service         yrs_in_mkt
Min.   :0.0000   Min.   :1.000   Min.   :0.0000   Min.   : 0.000
1st Qu.:0.0000   1st Qu.:1.000   1st Qu.:0.0000   1st Qu.: 2.000
Median :0.0000   Median :2.000   Median :0.0000   Median : 4.000
Mean   :0.3611   Mean   :3.665   Mean   :0.1019   Mean   : 6.581
3rd Qu.:1.0000   3rd Qu.:7.000   3rd Qu.:0.0000   3rd Qu.: 9.000
Max.   :5.0000   Max.   :7.000   Max.   :1.0000   Max.   :50.000
NA's   :8        NA's   :36                       NA's   :6
    profit         customers_pr_day
Min.   :     0.0   Min.   :     0.0
1st Qu.:   833.3   1st Qu.:    12.0
Median :  2000.0   Median :    20.0
Mean   :  4172.4   Mean   :   113.1
3rd Qu.:  5000.0   3rd Qu.:    30.0
Max.   :250000.0   Max.   :100000.0
NA's   :98         NA's   :166
                                                                  stall_type
Tarp, blanket or baskets on the ground                                :1125
Tempoarary stall or tablet that gets put up and taken down every day: 248
Uncovered permanent stall                                             : 256
Covered permanent stall WITHOUT lock                                  : 344
Covered permanent stall with lock                                     : 553
NA's                                                                  :   5

  any_constr                    trust_dist_gov                trust_ward_clr
Min.   :0.0000   Not at all trustworthy: 399   Not at all trustworthy:455
1st Qu.:0.0000   Not very trustworthy  : 508   Not very trustworthy  :557
Median :0.0000   Somewhat trustworthy  :1068   Somewhat trustworthy  :946
Mean   :0.2347   Very trustworthy      : 534   Very trustworthy      :489
3rd Qu.:0.0000   NA's                  :  22   NA's                  : 84
Max.   :1.0000
NA's   :26
             satisfaction_dev pay_even_disagree             tax_duty
Very Dissatisfied    :917     0   : 969         Strongly Disagree:  61
Somewhat Dissatisfied:532     1   :1559         Somewhat Disagree:  90
Somewhat Satisfied   :768     NA's:   3         Somewhat Agree   : 447
Very Satisfied       :313                       Strongly Agree   :1933
NA's                 :  1
```

```
  vote_intend    recent_receipt_7      test
 Min.   :0.000   0   :1863        Min.   :0.0000
 1st Qu.:1.000   1   : 666        1st Qu.:0.0000
 Median :1.000   NA's:   2        Median :0.0000
 Mean   :0.867                    Mean   :0.1999
 3rd Qu.:1.000                    3rd Qu.:0.0000
 Max.   :1.000                    Max.   :1.0000
 NA's   :4
```

```
selected_vendor_data <- vendor_data |>
  select(hh_income, acres_plots, bicycles,
         basic_cell_phones,yrs_in_mkt,  profit,
         customers_pr_day, female, married,
         pay_even_disagree, test, recent_receipt_7)
```

Splitting train/test data

```
train_data <- selected_vendor_data |>
  filter(test == 0) |>
  select(-test)

test_data <- selected_vendor_data |>
  filter(test == 1) |>
  select(-test)
```

1.2.1. Gradient Boosting

step_naomit(recent_receipt_7) - drops `NA` values from our outcome, `recent_receipt_7`

step_upsample(recent_receipt_7) - to balance the outcome `recent_receipt_7`

step_impute_mean - to replace NA's from all numeric features with their mean value

step_impute_mode - to replace NA's from all categorical features with their mode value

```
library(tidymodels)
library(bonsai)
library(themis)
```

```r
vendor_data_rec <- recipe(recent_receipt_7 ~ ., data = train_data) |>
  step_naomit(recent_receipt_7) |>
  step_upsample(recent_receipt_7) |>
  step_impute_mean(all_numeric(), -all_outcomes()) |>
  step_impute_mode(all_nominal(), -all_outcomes())



boost_vendor <- boost_tree(mode = "classification",
                           engine = "lightgbm",
                           # B
                           trees = tune(),
                           # d
                           tree_depth = tune(),
                           # lambda
                           learn_rate = tune())


boost_wf <- workflow() |>
  add_recipe(vendor_data_rec) |>
  add_model(boost_vendor)
```

Grid-Search Cross-Validation

Used trees from 500 to 3000 by 500. This range lets the model try smaller numbers of trees for quicker training and larger numbers for better accuracy, covering a good middle ground

tree_depth - depth controls how detailed each tree can get; shallow trees are simple and fast, while deeper ones can capture more complexity without going overboard

learn_rate (0.01, 0.05, 0.1).Learning rate affects how quickly the model adjusts; smaller values are careful but slow, and larger ones are faster but risk missing details, so these options balance it out.

```r
```{r cv-r}
#| eval: false

boost_grid <- crossing(
 trees =  seq(500, 3000, by = 500),
 tree_depth = 1:5,
 learn_rate = c(0.01, 0.05, 0.1)
)
```
```

```
folds <- vfold_cv(train_data,
               v = 6)

f_meas_sec_level <- metric_tweak("f_meas_sec_level", f_meas,
                                 event_level = "second")

boost_cv_vendor <- tune_grid(boost_wf,
                      resamples = folds,
                      grid = boost_grid,
                      metrics = metric_set(f_meas_sec_level)
                      )
save(boost_cv_vendor, file = "data/vendor_boost_cv_out.RData")
```

```{r load-cv-r}
#| eval: true

load(file = "data/vendor_boost_cv_out.RData")
```

```
collect_metrics(boost_cv_vendor) |>
  arrange(desc(mean))
```

```
# A tibble: 90 x 9
   trees tree_depth learn_rate .metric    .estimator  mean     n std_err .config
   <dbl>      <int>      <dbl> <chr>      <chr>      <dbl> <int>   <dbl> <chr>
 1  1500          1       0.05 f_meas_se~ binary     0.387     6  0.0128 Prepro~
 2  1000          1       0.1  f_meas_se~ binary     0.386     6  0.0116 Prepro~
 3  1500          1       0.01 f_meas_se~ binary     0.386     6  0.0166 Prepro~
 4  2000          1       0.05 f_meas_se~ binary     0.385     6  0.0121 Prepro~
 5   500          1       0.05 f_meas_se~ binary     0.384     6  0.0134 Prepro~
 6  3000          1       0.01 f_meas_se~ binary     0.383     6  0.0129 Prepro~
 7  1000          1       0.01 f_meas_se~ binary     0.383     6  0.0178 Prepro~
 8  2000          1       0.01 f_meas_se~ binary     0.382     6  0.0152 Prepro~
 9  2500          1       0.01 f_meas_se~ binary     0.381     6  0.0139 Prepro~
10  2500          1       0.05 f_meas_se~ binary     0.380     6  0.0144 Prepro~
# i 80 more rows
```

```{r refit-r}
#| eval: false
boost_wf_best <- boost_wf |>
```

```
  finalize_workflow(select_best(boost_cv_vendor, metric = "f_meas_sec_level")) |>
  fit(train_data)
```

```{r eval-mod-r}
#| eval: false
vendor_test_aug <- boost_wf_best |>
  augment(new_data = test_data)

vendor_test_aug |>
  f_meas(recent_receipt_7,
         .pred_class,
         event_level = "second")

vendor_test_aug |>
  conf_mat(recent_receipt_7,
           .pred_class)
```

The F1 score of the model is 0.385. Gradient Boosting did not perform well.

Iterative Search Cross-Validation(For Bonus Point)

`trees(range = c(1000, 3000))`. The model tries between 1000 and 3000 trees to balance accuracy and training time. Fewer trees train faster, and more trees can capture complex patterns.

`iter = 100`. The model tests 100 different combinations of parameters.

```{r cv-bayes-r}
#| eval: false

boost_params <- extract_parameter_set_dials(boost_wf)

boost_params <- boost_params |>
  update(trees = trees(range = c(1000, 3000)))

set.seed(756)
boost_cv_bayes_vendor <- boost_wf |>
  tune_bayes(
    resamples = folds,
```

```
    param_info = boost_params,
    initial = boost_cv_vendor,
    iter = 100,
    metrics = metric_set(f_meas_sec_level),
    control = control_bayes(no_improve = 15)
  )

save(boost_cv_bayes_vendor, file = "data/vendor_boost_cv_bayes_out.RData")
```

```
load(file = "data/vendor_boost_cv_bayes_out.RData")
```

```
collect_metrics(boost_cv_bayes_vendor) |>
  arrange(desc(mean))
```

```
# A tibble: 108 x 10
   trees tree_depth learn_rate .metric    .estimator  mean     n std_err .config
   <dbl>      <int>      <dbl> <chr>      <chr>      <dbl> <int>   <dbl> <chr>
 1  1488          4   1.05e-10 f_meas_se~ binary     0.387     6  0.0218 Iter3
 2  1500          1   5   e- 2 f_meas_se~ binary     0.387     6  0.0128 Prepro~
 3  1000          1   1   e- 1 f_meas_se~ binary     0.386     6  0.0116 Prepro~
 4  1500          1   1   e- 2 f_meas_se~ binary     0.386     6  0.0166 Prepro~
 5  2000          1   5   e- 2 f_meas_se~ binary     0.385     6  0.0121 Prepro~
 6   500          1   5   e- 2 f_meas_se~ binary     0.384     6  0.0134 Prepro~
 7  3000          1   1   e- 2 f_meas_se~ binary     0.383     6  0.0129 Prepro~
 8  1000          1   1   e- 2 f_meas_se~ binary     0.383     6  0.0178 Prepro~
 9  2000          1   1   e- 2 f_meas_se~ binary     0.382     6  0.0152 Prepro~
10  1064         14   3.75e- 6 f_meas_se~ binary     0.382     6  0.0178 Iter13
# i 98 more rows
# i 1 more variable: .iter <int>
```

```
```{r eval-bayes-r}
#| eval: true

boost_wf_best_bayes <- boost_wf |>
  finalize_workflow(select_best(boost_cv_bayes_vendor,
                                metric = "f_meas_sec_level")) |>
  fit(train_data)

vendor_aug_bayes <- boost_wf_best_bayes |>
  augment(new_data = test_data)
```
```

```
vendor_aug_bayes |>
  f_meas(recent_receipt_7,
         .pred_class,
         event_level = "second")

vendor_aug_bayes |>
  conf_mat(recent_receipt_7,
           .pred_class)
vendor_aug_bayes
```

```
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 f_meas  binary         0.386
          Truth
Prediction   0   1
         0  79  28
         1 297 102
# A tibble: 506 x 14
   .pred_class .pred_0 .pred_1 hh_income acres_plots bicycles basic_cell_phones
   <fct>         <dbl>   <dbl>     <dbl>       <dbl>    <dbl>              <dbl>
 1 0             0.500   0.500     50000        0           1                  2
 2 0             0.500   0.500     70000        1           1                  2
 3 0             0.500   0.500     70000        0           1                  1
 4 0             0.500   0.500     25000        0.25        1                  1
 5 0             0.500   0.500     15000        5           0                  0
 6 1             0.500   0.500     80000        0           1                  2
 7 0             0.500   0.500     40000        0.25        1                  0
 8 1             0.500   0.500    800000        0           0                  3
 9 1             0.500   0.500     50000        0           0                  0
10 1             0.500   0.500     70000        0           1                  1
# i 496 more rows
# i 7 more variables: yrs_in_mkt <dbl>, profit <dbl>, customers_pr_day <dbl>,
#   female <fct>, married <fct>, pay_even_disagree <fct>,
#   recent_receipt_7 <fct>
```

Iterative Search Cross-Validation slightly improved the model's performance compared to Grid-Search Cross-Validation.

The F1 score of the model is 0.43

1.2.2 Random Forest

```
rf_model <- rand_forest(
  mode = "classification",
  trees = tune(),
  mtry = tune()
) |>
  set_engine("ranger")
```

We will use the same recipe as Gradient Boosting

```
rf_recipe_vendor <- recipe(recent_receipt_7 ~ ., data = train_data) |>
  step_naomit(recent_receipt_7) |>
  step_upsample(recent_receipt_7) |>
  step_impute_mean(all_numeric(), -all_outcomes()) |>
  step_impute_mode(all_nominal(), -all_outcomes())

rf_workflow <- workflow() |>
  add_recipe(rf_recipe_vendor) |>
  add_model(rf_model)
```

Grid-Search Cross-Validation

trees(range = c(100, 3000)) to balance between accurate predictions (more trees) and faster training (fewer trees).

mtry(c(2, ncol(train_data) - 1)) tries different numbers of predictors for each split, from a small amount (2) to almost all predictors, to see what works best, uses all predictors except the target value

```
rf_grid <- grid_regular(
  trees(range = c(100, 3000)),
  mtry(range = c(2, ncol(train_data) - 1)),
  levels = 8
)
```

```
set.seed(123)
rf_folds <- vfold_cv(train_data, v = 6)
f_meas_sec_level <- metric_tweak("f_meas_sec_level", f_meas,
                                 event_level = "second")
```

```
rf_results_vendor <- tune_grid(
  rf_workflow,
  resamples = rf_folds,
  grid = rf_grid,
  metrics = metric_set(f_meas_sec_level),
  control = control_grid(save_pred = TRUE)
)
save(rf_results_vendor, file = "data/vendor_rf_cv_out.RData")
```

```
load(file = "data/vendor_rf_cv_out.RData")
```

```
collect_metrics(rf_results_vendor) |>
  arrange(desc(mean))
```

```
# A tibble: 64 x 8
    mtry trees .metric          .estimator  mean     n std_err .config
   <int> <int> <chr>            <chr>      <dbl> <int>   <dbl> <chr>
 1     8   100 f_meas_sec_level binary     0.250     6 0.00457 Preprocessor1_Mo~
 2    10  2585 f_meas_sec_level binary     0.236     6 0.00910 Preprocessor1_Mo~
 3     8   928 f_meas_sec_level binary     0.235     6 0.0101  Preprocessor1_Mo~
 4    10  1757 f_meas_sec_level binary     0.235     6 0.00924 Preprocessor1_Mo~
 5    10  2171 f_meas_sec_level binary     0.235     6 0.0103  Preprocessor1_Mo~
 6     5   100 f_meas_sec_level binary     0.234     6 0.0146  Preprocessor1_Mo~
 7     8   514 f_meas_sec_level binary     0.234     6 0.0143  Preprocessor1_Mo~
 8    10   514 f_meas_sec_level binary     0.233     6 0.00961 Preprocessor1_Mo~
 9     8  3000 f_meas_sec_level binary     0.232     6 0.0122  Preprocessor1_Mo~
10     8  1342 f_meas_sec_level binary     0.231     6 0.0127  Preprocessor1_Mo~
# i 54 more rows
```

```
best_rf <- rf_results_vendor |>
  select_best(metric = "f_meas_sec_level")
```

```
rf_workflow_best <- rf_workflow |>
  finalize_workflow(best_rf)

rf_final_model <- rf_workflow_best |>
  fit(train_data)
```

```
rf_test_results_vendor <- rf_final_model |>
  augment(new_data = test_data)
```

```
rf_test_results_vendor |>
  f_meas(recent_receipt_7, .pred_class, event_level = "second")
```

```
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 f_meas  binary          0.24
```

```
rf_test_results_vendor |>
  conf_mat(recent_receipt_7, .pred_class)
```

```
          Truth
Prediction   0   1
         0 330 106
         1  46  24
```

Iterative Search Cross-Validation for Random Forest(For Bonus Point)

```
rf_params <- extract_parameter_set_dials(rf_workflow) |>
  update(
    trees = trees(range = c(1000, 3000)),
    mtry = mtry(range = c(2, ncol(train_data) - 1))
  )

set.seed(456)
rf_iterative_results <- rf_workflow |>
  tune_bayes(
    resamples = rf_folds,
    param_info = rf_params,
    iter = 100,
    metrics = metric_set(f_meas_sec_level),
    control = control_bayes(no_improve = 15, save_pred = TRUE)
  )
```

```
! No improvement for 15 iterations; returning current results.
```

```
save(rf_iterative_results, file = "data/vendor_rf_iterative_bayes_out.RData")
```

Load file

```
load(file = "data/vendor_rf_iterative_bayes_out.RData")
```

Finalize the workflow with the best parameters

```
collect_metrics(rf_iterative_results) |>
  arrange(desc(mean))
```

```
# A tibble: 35 x 9
    mtry trees .metric          .estimator  mean     n std_err .config     .iter
   <int> <int> <chr>            <chr>      <dbl> <int>   <dbl> <chr>       <int>
 1     9  1705 f_meas_sec_level binary     0.240     6 0.00973 Iter15         15
 2     9  1726 f_meas_sec_level binary     0.236     6 0.00809 Iter6           6
 3     9  1304 f_meas_sec_level binary     0.235     6 0.00957 Iter21         21
 4     9  1876 f_meas_sec_level binary     0.235     6 0.0104  Preprocess~     0
 5     9  1794 f_meas_sec_level binary     0.232     6 0.0118  Iter8           8
 6    10  1311 f_meas_sec_level binary     0.232     6 0.0133  Iter18         18
 7     9  2997 f_meas_sec_level binary     0.232     6 0.00865 Iter17         17
 8     9  2269 f_meas_sec_level binary     0.232     6 0.0103  Iter26         26
 9     8  1786 f_meas_sec_level binary     0.231     6 0.0114  Iter7           7
10     8  1694 f_meas_sec_level binary     0.231     6 0.00932 Iter11         11
# i 25 more rows
```

```
best_rf_iterative <- rf_iterative_results |>
  select_best(metric = "f_meas_sec_level")

rf_workflow_best_iterative <- rf_workflow |>
  finalize_workflow(best_rf_iterative)

rf_final_model_iterative <- rf_workflow_best_iterative |>
  fit(train_data)
```

Testing and Evaluation

```
rf_test_results_iterative <- rf_final_model_iterative |>
  augment(new_data = test_data)

rf_test_results_iterative |>
  f_meas(recent_receipt_7, .pred_class, event_level = "second")
```

```
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 f_meas  binary         0.266
```

```
rf_test_results_iterative |>
  conf_mat(recent_receipt_7, .pred_class)
```

```
          Truth
Prediction   0   1
         0 330 103
         1  46  27
```

1.3. Gradient Boosting with Iterative Search Cross-Validation performed better than others

Gradient Boosting Grid-Search Cross-Validation: 0.38

Gradient Boosting Iterative Search Cross-Validation:0.43

Random Forest Grid-Search Cross-Validation:0.23

Random Forest Iterative Search Cross-Validation:0.25

1.4. I can't compare with Assignment 2, because I used different features and f_meas with default level(first)

2. This semester, I found Gradient Boosting to be one of the most interesting techniques we studied. It improves predictions step by step by correcting errors from earlier stages, making it a practical and efficient method. It works well with complex tasks like imbalanced or noisy data, and its settings, such as the number of trees and learning rate, can be easily adjusted for different needs.