VPS Customer churn prediction - Part 2

JG Pardyak

02-9-2021

Introduction

Motivation

This is the continuation of the presentations:

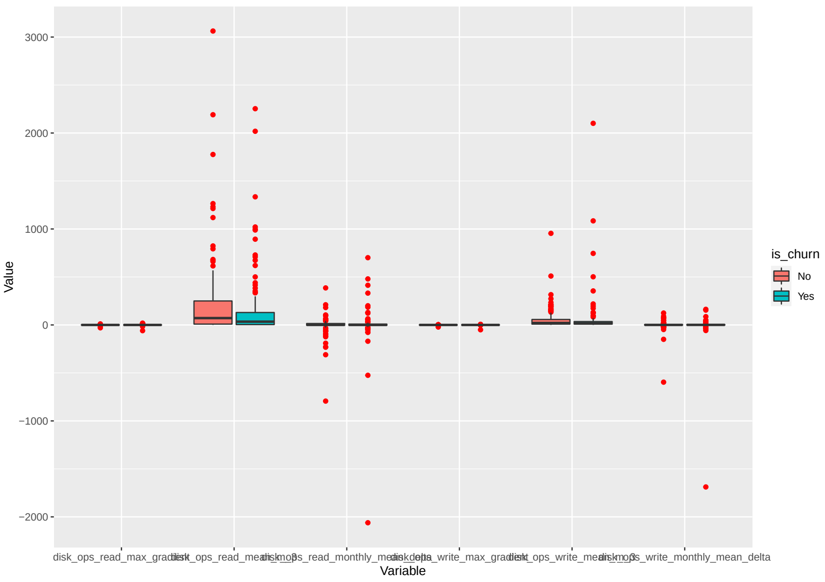
- https://github.com/JacekPardyak/vps/blob/master/vps-part-1.pdf ,
- https://github.com/JacekPardyak/vps/blob/master/vps-part-2.pdf .

In this presentation we check:

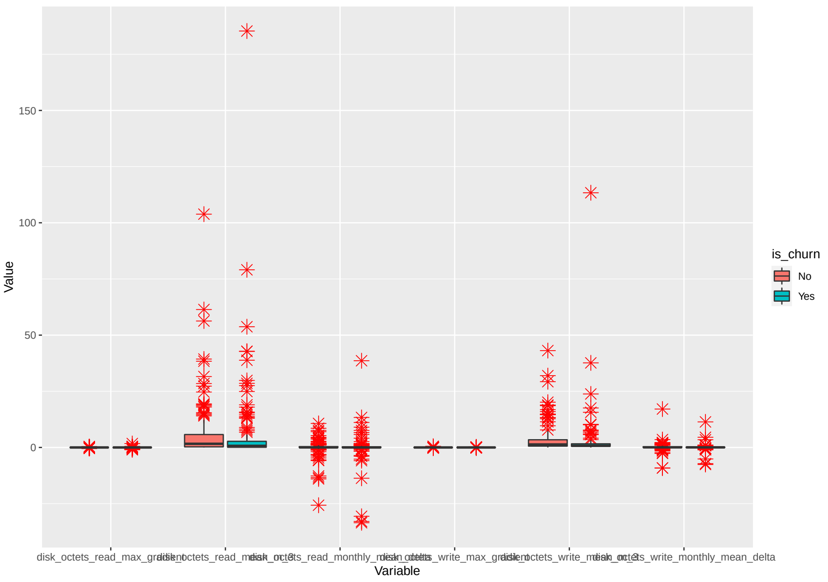
- outliers - data points that differs significantly from other observations,
- Feature engineering - combining and transforming further existing variables,
- Model tuning of `decision_tree` and `rand_forest`

Outlier detection

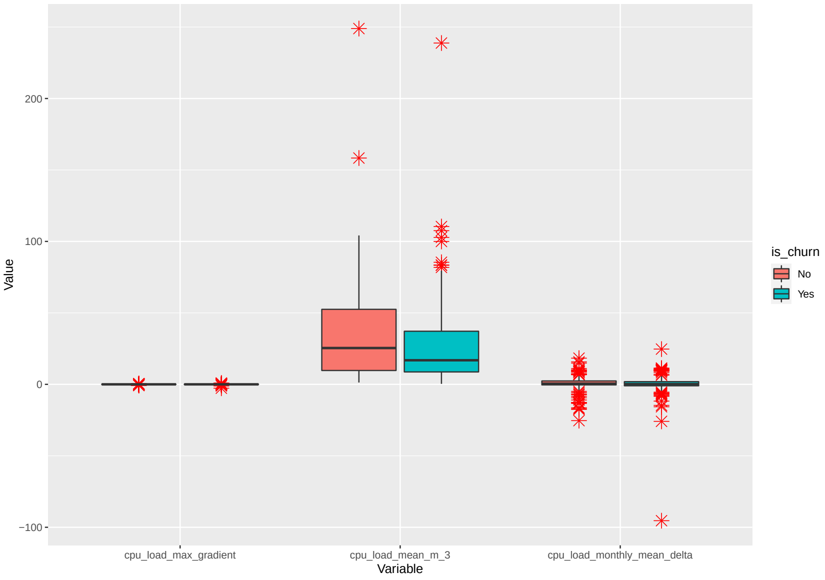
disk_ops variables



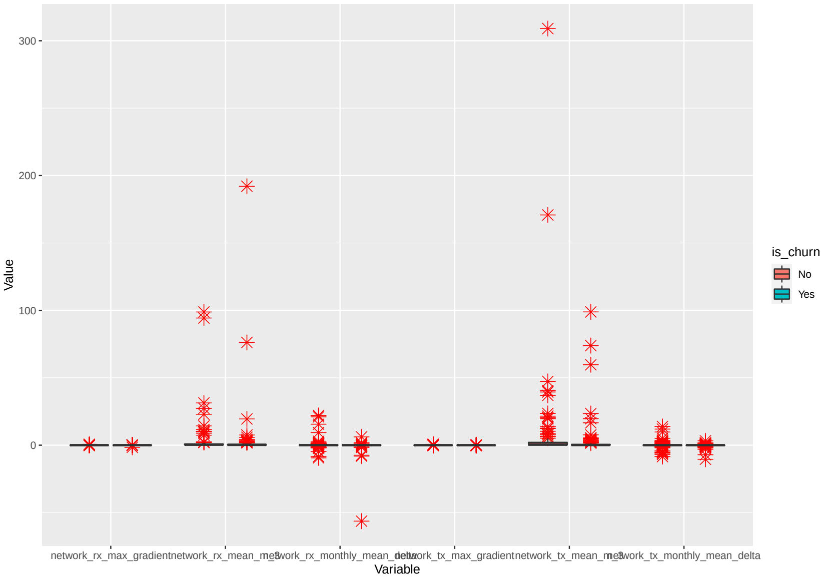
disk_octets variables



cpu variables



network variables



Tune `decision_tree` hyperparameters

Split and prepare data

```
set.seed(123)
vps_split <- initial_split(vps, strata = is_churn)
vps_train <- training(vps_split)
vps_test  <- testing(vps_split)
vps_recipe <- vps_train %>%
  recipe(is_churn ~ .) %>%
  step_rm(id) %>%
  step_corr(all_predictors()) #%>%
  # make vars to be of mean zero
#  step_center(all_predictors(), -all_outcomes()) %>%
  # make vars to be standard dev of 1
  #step_scale(all_predictors(), -all_outcomes())
```

Create model specification for tuning

```r
tune_spec <-
  decision_tree(
    cost_complexity = tune(),
    tree_depth = tune()
  ) %>%
  set_engine("rpart") %>%
  set_mode("classification")

tune_spec
```

```
## Decision Tree Model Specification (classification)
##
## Main Arguments:
##   cost_complexity = tune()
##   tree_depth = tune()
##
## Computational engine: rpart
```

Create grid of hyperparameters values

```
tree_grid <- grid_regular(cost_complexity(),
                          tree_depth(),
                          levels = 5)
tree_grid
```

```
## # A tibble: 25 x 2
##    cost_complexity tree_depth
##              <dbl>      <int>
##  1    0.0000000001          1
##  2    0.0000000178          1
##  3    0.00000316            1
##  4    0.000562              1
##  5    0.1                   1
##  6    0.0000000001          4
##  7    0.0000000178          4
##  8    0.00000316            4
##  9    0.000562              4
## 10    0.1                   4
## # ... with 15 more rows
```

Create cross-validation folds for tuning

```
set.seed(234)
vps_folds <- vfold_cv(vps_train)
vps_folds
```

```
## #  10-fold cross-validation
## # A tibble: 10 x 2
##    splits           id
##    <list>           <chr>
##  1 <split [190/22]> Fold01
##  2 <split [190/22]> Fold02
##  3 <split [191/21]> Fold03
##  4 <split [191/21]> Fold04
##  5 <split [191/21]> Fold05
##  6 <split [191/21]> Fold06
##  7 <split [191/21]> Fold07
##  8 <split [191/21]> Fold08
##  9 <split [191/21]> Fold09
## 10 <split [191/21]> Fold10
```

Fit models at all the different values

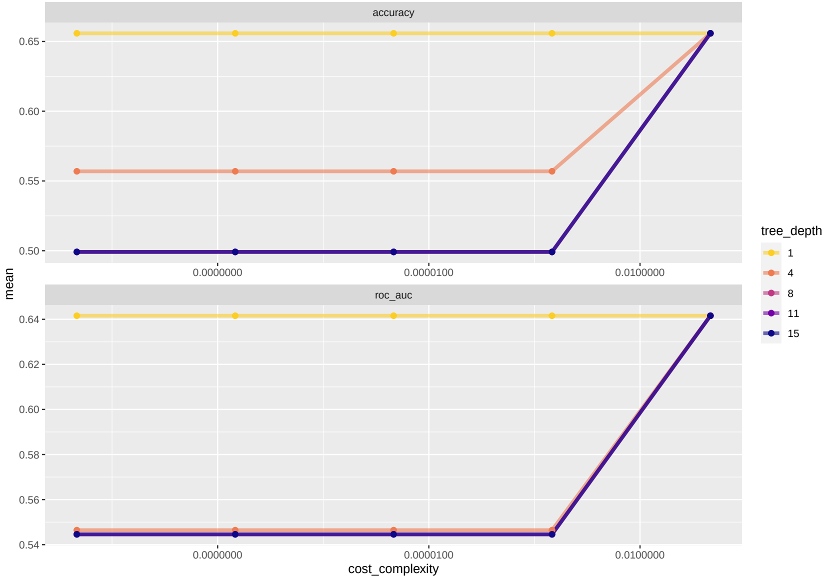
```
set.seed(345)
tune_wf <- workflow() %>%
  add_model(tune_spec) %>%
  add_recipe(vps_recipe)
tree_res <-
  tune_wf %>%
  tune_grid(
    resamples = vps_folds,
    grid = tree_grid
    )

tree_res
```

```
## # Tuning results
## # 10-fold cross-validation
## # A tibble: 10 x 4
##    splits           id     .metrics            .notes
##    <list>           <chr>  <list>              <list>
##  1 <split [190/22]> Fold01 <tibble [50 x 6]> <tibble [0
```

Get metrics of different models

```
## # A tibble: 50 x 8
##    cost_complexity tree_depth .metric  .estimator  mean
##              <dbl>      <int> <chr>    <chr>      <dbl>
##  1    0.0000000001          1 accuracy binary     0.656
##  2    0.0000000001          1 roc_auc  binary     0.642
##  3    0.0000000178          1 accuracy binary     0.656
##  4    0.0000000178          1 roc_auc  binary     0.642
##  5    0.00000316            1 accuracy binary     0.656
##  6    0.00000316            1 roc_auc  binary     0.642
##  7    0.000562              1 accuracy binary     0.656
##  8    0.000562              1 roc_auc  binary     0.642
##  9    0.1                   1 accuracy binary     0.656
## 10    0.1                   1 roc_auc  binary     0.642
## # ... with 40 more rows
```

Plot metrics of different models



Show models with best metrics

```
## # A tibble: 5 x 8
##   cost_complexity tree_depth .metric  .estimator  mean
##             <dbl>      <int> <chr>    <chr>      <dbl> <
## 1    0.0000000001          1 accuracy binary     0.656
## 2    0.0000000178          1 accuracy binary     0.656
## 3    0.00000316            1 accuracy binary     0.656
## 4    0.000562              1 accuracy binary     0.656
## 5    0.1                   1 accuracy binary     0.656
```

Pick one model with the best metrics

```
best_tree <- tree_res %>%
  select_best("accuracy")
best_tree
```

```
## # A tibble: 1 x 3
##   cost_complexity tree_depth .config
##             <dbl>      <int> <chr>
## 1    0.0000000001          1 Preprocessor1_Model01
```

Finalizing best model

```r
final_wf <-
  tune_wf %>%
  finalize_workflow(best_tree)

final_wf
```

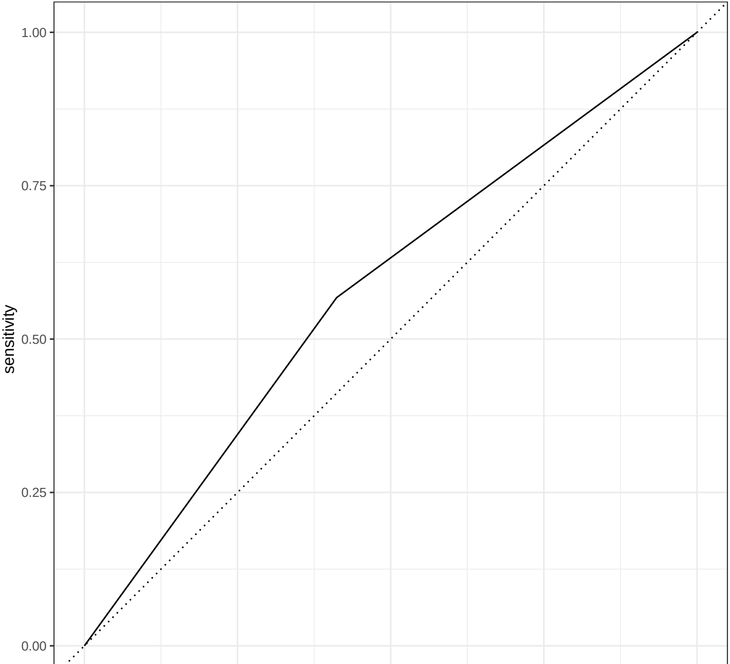
```
## == Workflow ==================================
## Preprocessor: Recipe
## Model: decision_tree()
##
## -- Preprocessor ------------------------------
## 2 Recipe Steps
##
## * step_rm()
## * step_corr()
##
## -- Model -------------------------------------
## Decision Tree Model Specification (classification)
##
```

The final fit

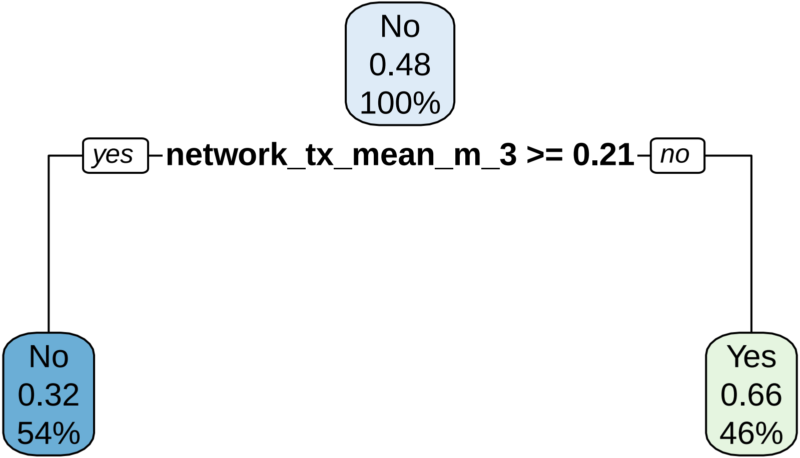
```r
final_fit <-
  final_wf %>%
  last_fit(vps_split)

final_fit %>%
  collect_metrics()
```

```
## # A tibble: 2 x 4
##   .metric  .estimator .estimate .config
##   <chr>    <chr>          <dbl> <chr>
## 1 accuracy binary         0.577 Preprocessor1_Model1
## 2 roc_auc  binary         0.578 Preprocessor1_Model1
```

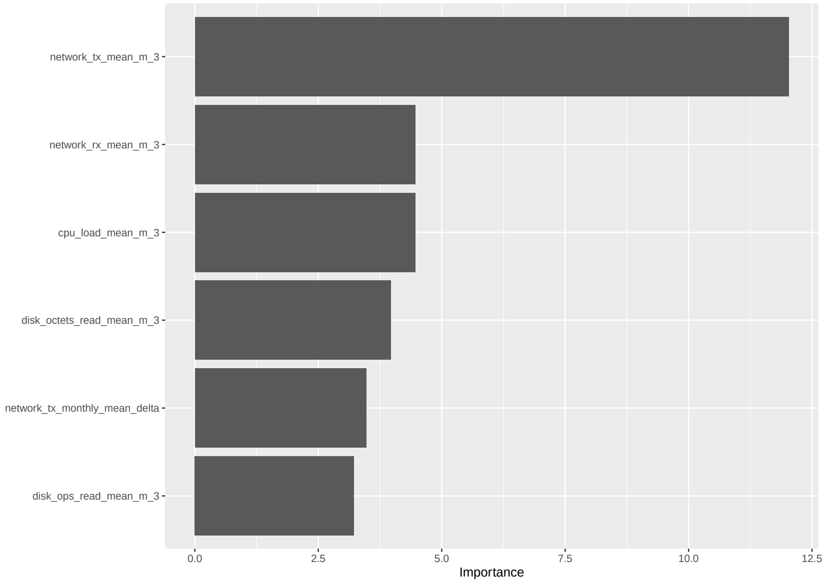
The final fit ROC curve



Visualizing decision tree from the workflow



Estimate variable importance



Tune `rand_forest` hyperparameters

Create model specification for tuning

```
tune_spec <- rand_forest(
  mtry = tune(),
  trees = 1000,
  min_n = tune()
) %>%
  set_engine("ranger") %>%
  set_mode("classification")

tune_spec
```

```
## Random Forest Model Specification (classification)
##
## Main Arguments:
##   mtry = tune()
##   trees = 1000
##   min_n = tune()
##
## Computational engine: ranger
```

Create cross-validation folds for tuning

```r
set.seed(234)
vps_folds <- vfold_cv(vps_train)
vps_folds
```

```
## #  10-fold cross-validation
## # A tibble: 10 x 2
##    splits           id
##    <list>           <chr>
##  1 <split [190/22]> Fold01
##  2 <split [190/22]> Fold02
##  3 <split [191/21]> Fold03
##  4 <split [191/21]> Fold04
##  5 <split [191/21]> Fold05
##  6 <split [191/21]> Fold06
##  7 <split [191/21]> Fold07
##  8 <split [191/21]> Fold08
##  9 <split [191/21]> Fold09
## 10 <split [191/21]> Fold10
```

Initially fit models at all the different values

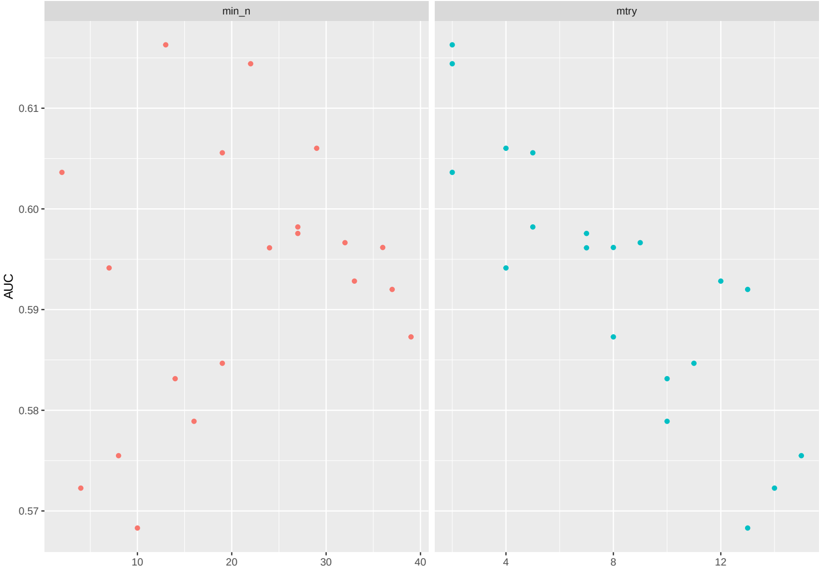
```
set.seed(234)
doParallel::registerDoParallel()
tune_wf <- workflow() %>%
  add_recipe(vps_recipe) %>%
  add_model(tune_spec)

tune_res <- tune_grid(
  tune_wf,
  resamples = vps_folds,
  grid = 20
)
```

```
## i Creating pre-processing data to finalize unknown paran
```

Get metrics of different models

```
## # A tibble: 40 x 8
##     mtry min_n .metric  .estimator  mean     n std_err .
##    <int> <int> <chr>    <chr>      <dbl> <int>   <dbl> <
##  1     2     2 accuracy binary     0.571    10  0.0342 P
##  2     2     2 roc_auc  binary     0.604    10  0.0399 P
##  3    10    16 accuracy binary     0.562    10  0.0389 P
##  4    10    16 roc_auc  binary     0.579    10  0.0435 P
##  5     4    29 accuracy binary     0.599    10  0.0358 P
##  6     4    29 roc_auc  binary     0.606    10  0.0426 P
##  7     8    36 accuracy binary     0.585    10  0.0384 P
##  8     8    36 roc_auc  binary     0.596    10  0.0404 P
##  9     7    24 accuracy binary     0.566    10  0.0381 P
## 10     7    24 roc_auc  binary     0.596    10  0.0417 P
## # ... with 30 more rows
```

Plot initial tuning results



Create detailed grid of hyperparameters values

```
rf_grid <- grid_regular(
  mtry(range = c(1, 3)),
  min_n(range = c(1, 5)),
  levels = 5
)

rf_grid
```

```
## # A tibble: 15 x 2
##      mtry min_n
##     <int> <int>
##  1      1     1
##  2      2     1
##  3      3     1
##  4      1     2
##  5      2     2
##  6      3     2
##  7      1     3
```

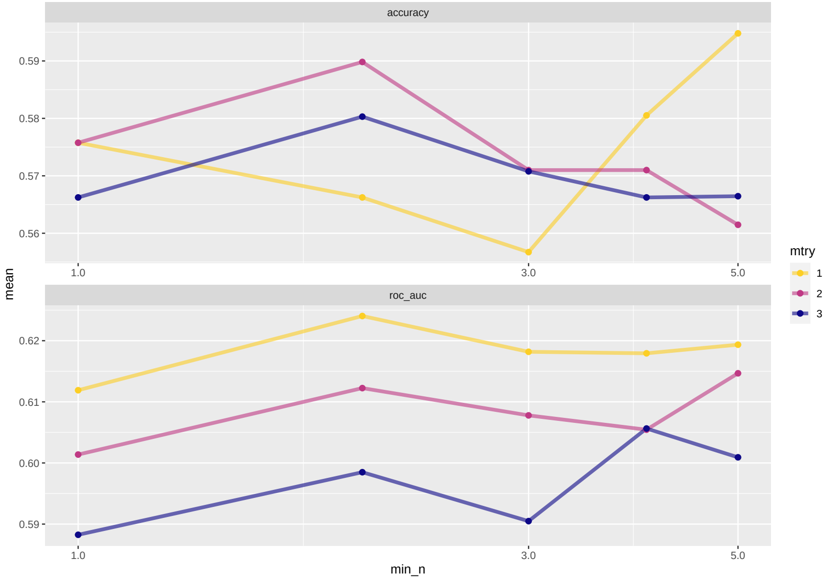
Fit models at all the different values

```
set.seed(456)
forest_res <- tune_wf %>%
  tune_grid(
  resamples = vps_folds,
  grid = rf_grid
)
```

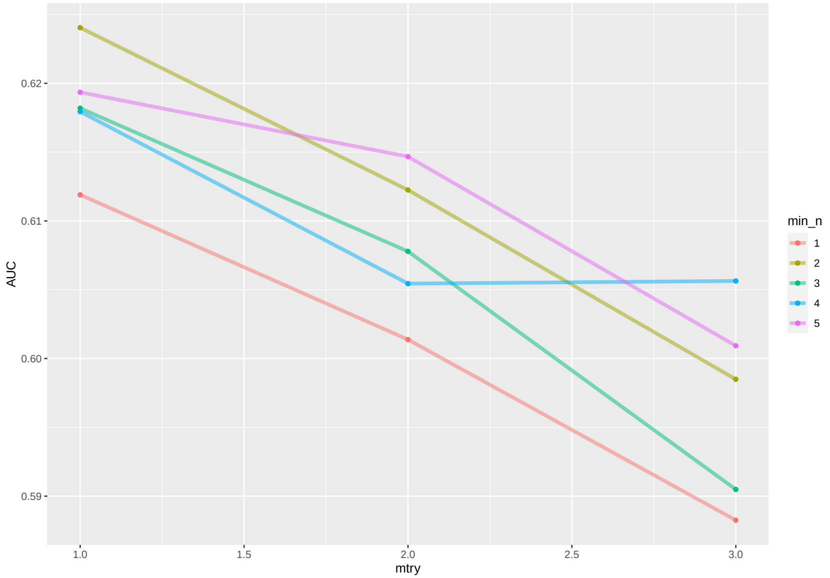
Get metrics of different models

```
## # A tibble: 30 x 8
##     mtry min_n .metric  .estimator  mean     n std_err .
##    <int> <int> <chr>    <chr>      <dbl> <int>   <dbl> <
##  1     1     1 accuracy binary     0.576    10  0.0329 P
##  2     1     1 roc_auc  binary     0.612    10  0.0388 P
##  3     2     1 accuracy binary     0.576    10  0.0297 P
##  4     2     1 roc_auc  binary     0.601    10  0.0426 P
##  5     3     1 accuracy binary     0.566    10  0.0313 P
##  6     3     1 roc_auc  binary     0.588    10  0.0438 P
##  7     1     2 accuracy binary     0.566    10  0.0344 P
##  8     1     2 roc_auc  binary     0.624    10  0.0414 P
##  9     2     2 accuracy binary     0.590    10  0.0305 P
## 10     2     2 roc_auc  binary     0.612    10  0.0437 P
## # ... with 20 more rows
```

Plot metrics of different models



Plot metrics of different models



Show models with best metrics

```
## # A tibble: 5 x 8
##    mtry min_n .metric  .estimator  mean     n std_err .c
##   <int> <int> <chr>    <chr>      <dbl> <int>   <dbl> <c
## 1     1     5 accuracy binary     0.595    10  0.0345 Pr
## 2     2     2 accuracy binary     0.590    10  0.0305 Pr
## 3     1     4 accuracy binary     0.581    10  0.0354 Pr
## 4     3     2 accuracy binary     0.580    10  0.0333 Pr
## 5     1     1 accuracy binary     0.576    10  0.0329 Pr
```

Pick one model with the best metrics

```
best_forest <- forest_res %>%
  select_best("accuracy")
best_forest
```

```
## # A tibble: 1 x 3
##    mtry min_n .config
##   <int> <int> <chr>
## 1     1     5 Preprocessor1_Model13
```

Finalizing best model

```
final_wf <-
  tune_wf %>%
  finalize_workflow(best_forest)

final_wf
```

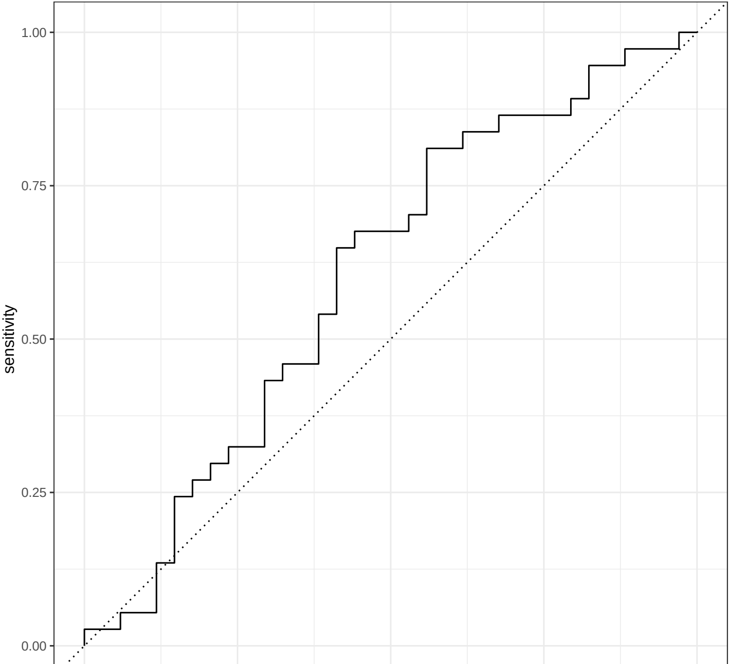
```
## == Workflow ================================
## Preprocessor: Recipe
## Model: rand_forest()
##
## -- Preprocessor ----------------------------
## 2 Recipe Steps
##
## * step_rm()
## * step_corr()
##
## -- Model -----------------------------------
## Random Forest Model Specification (classification)
##
```

The final fit

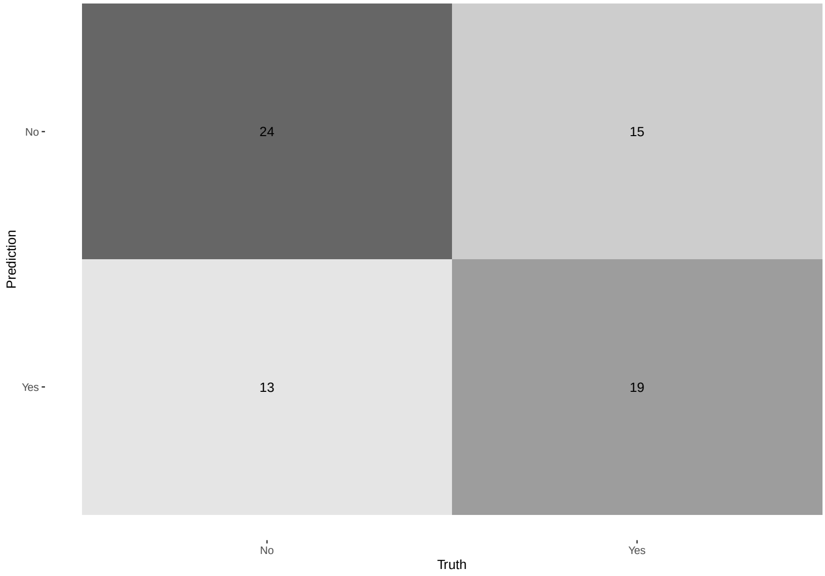
```r
final_fit <-
  final_wf %>%
  last_fit(vps_split)

final_fit %>%
  collect_metrics()
```

```
## # A tibble: 2 x 4
##   .metric  .estimator .estimate .config
##   <chr>    <chr>          <dbl> <chr>
## 1 accuracy binary         0.606 Preprocessor1_Model1
## 2 roc_auc  binary         0.603 Preprocessor1_Model1
```

The final fit ROC curve



The final fit Confusion Matrix



Model deployment

Model deployment

```
production <- read_csv("./data/vps_test_data.txt")

tmp <- final_wf %>% fit(vps_train)  %>% predict(production)
  rename(is_churn = .pred_class) %>%
  mutate(is_churn = ifelse(is_churn == "Yes", 1, 0 ))

production %>% select(! one_of('is_churn')) %>%
  bind_cols(tmp) %>%
  write_csv("./data/vps_test_data_pred_part_3.txt")
```

Further steps

Further steps

- demonstrate how to use **SparkR** (R on Spark)