VPS Customer churn prediction - Part 2

JG Pardyak

02-9-2021

Introduction

Data explorations & preparation

Train models

Evaluate models

The best model

Model deployment

Introduction

Motivation

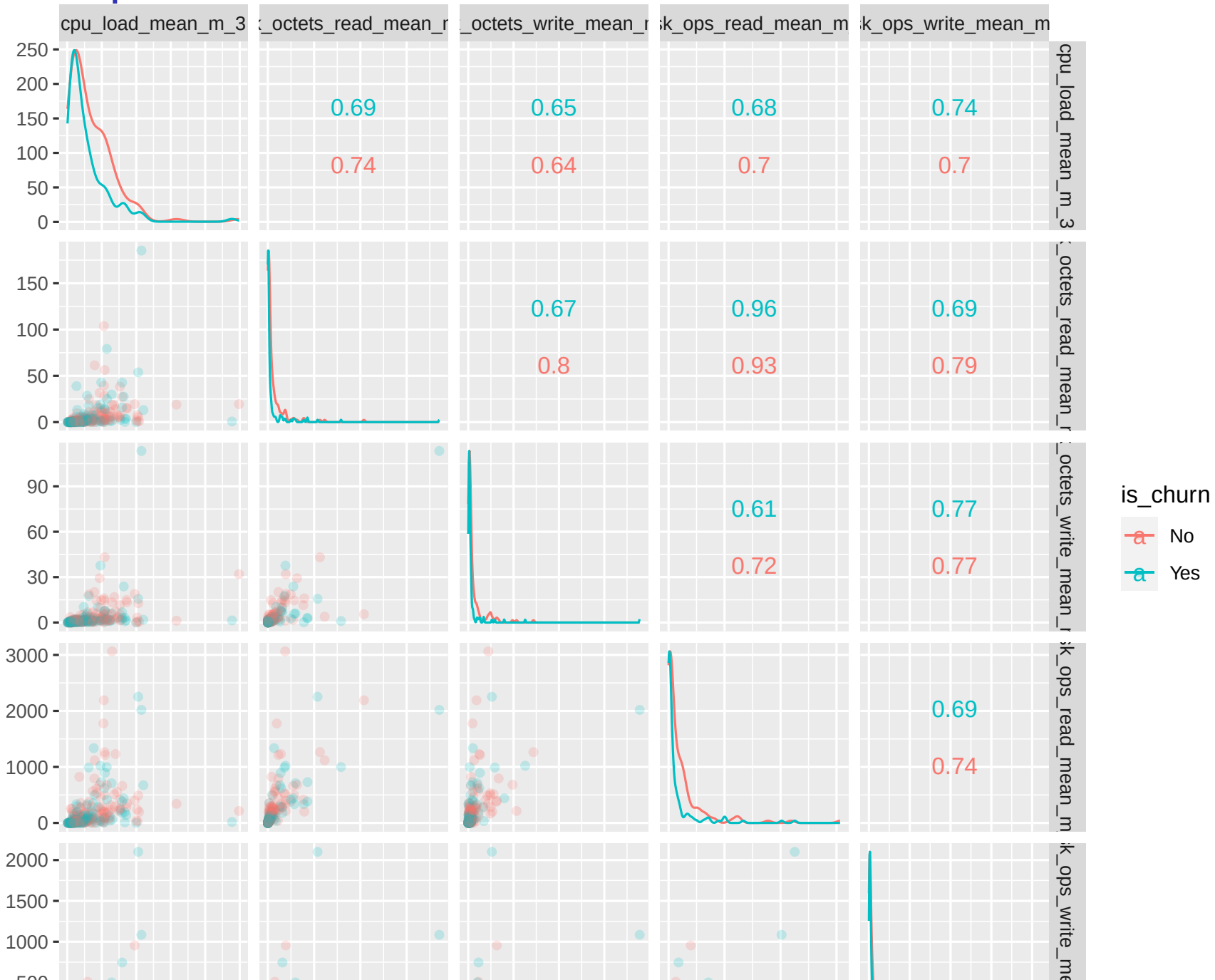
This is the continuation of the presentation
https://github.com/JacekPardyak/vps/blob/master/vps.pdf.

Data explorations & preparation

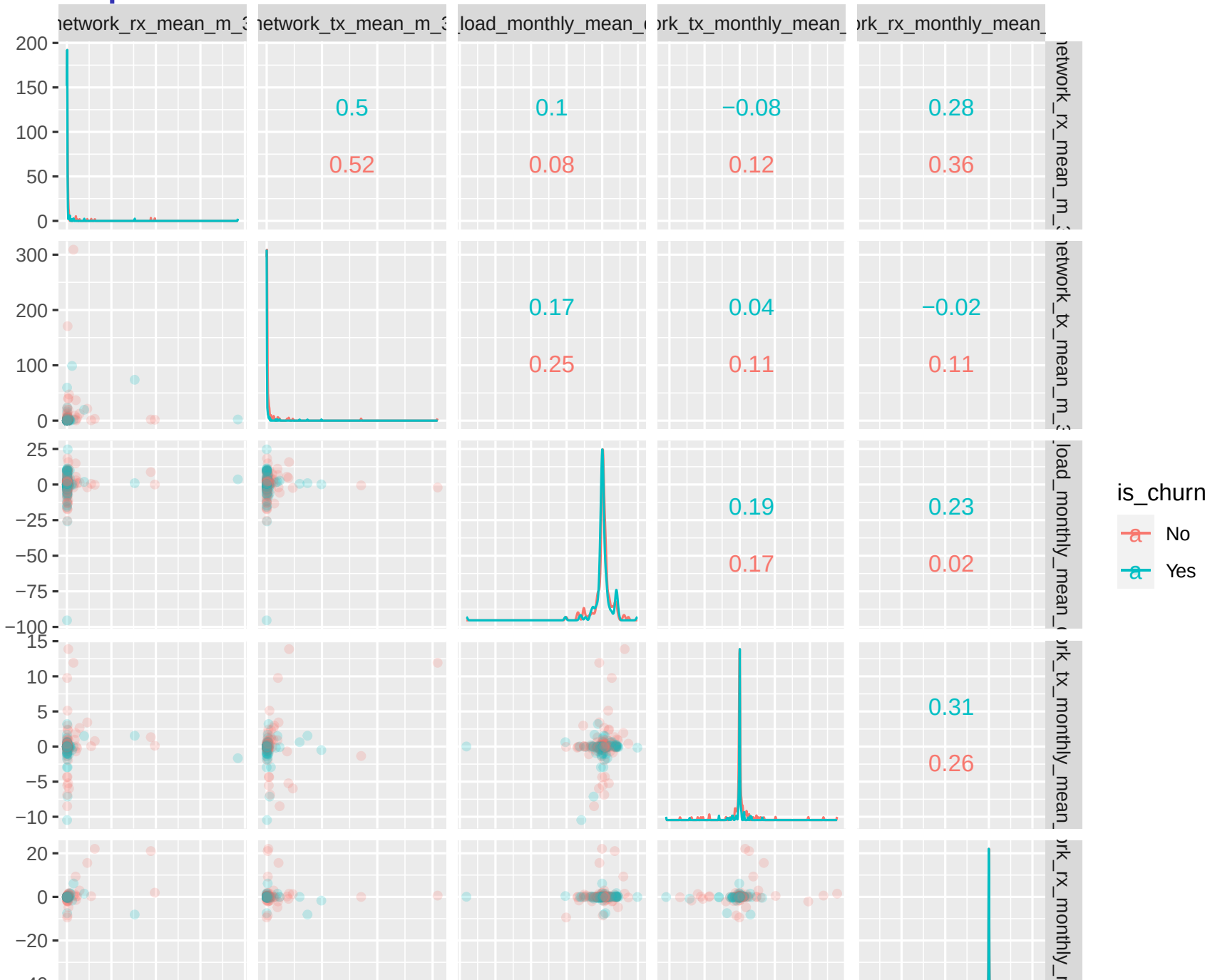
Loading data and libraries

```
library(tidyverse)
library(tidymodels)
library(GGally)
vps <- read_csv("./data/vps_churn_data.txt") %>%
  mutate(is_churn = factor(ifelse(is_churn == 0,
                                  "No", "Yes")))
# this chunk is used to generate subsequent charts
# vps %>%
#  ggscatmat(columns = c(2:7),
#            color = 'is_churn',
#            corMethod = "spearman",
#            alpha=0.2)
```

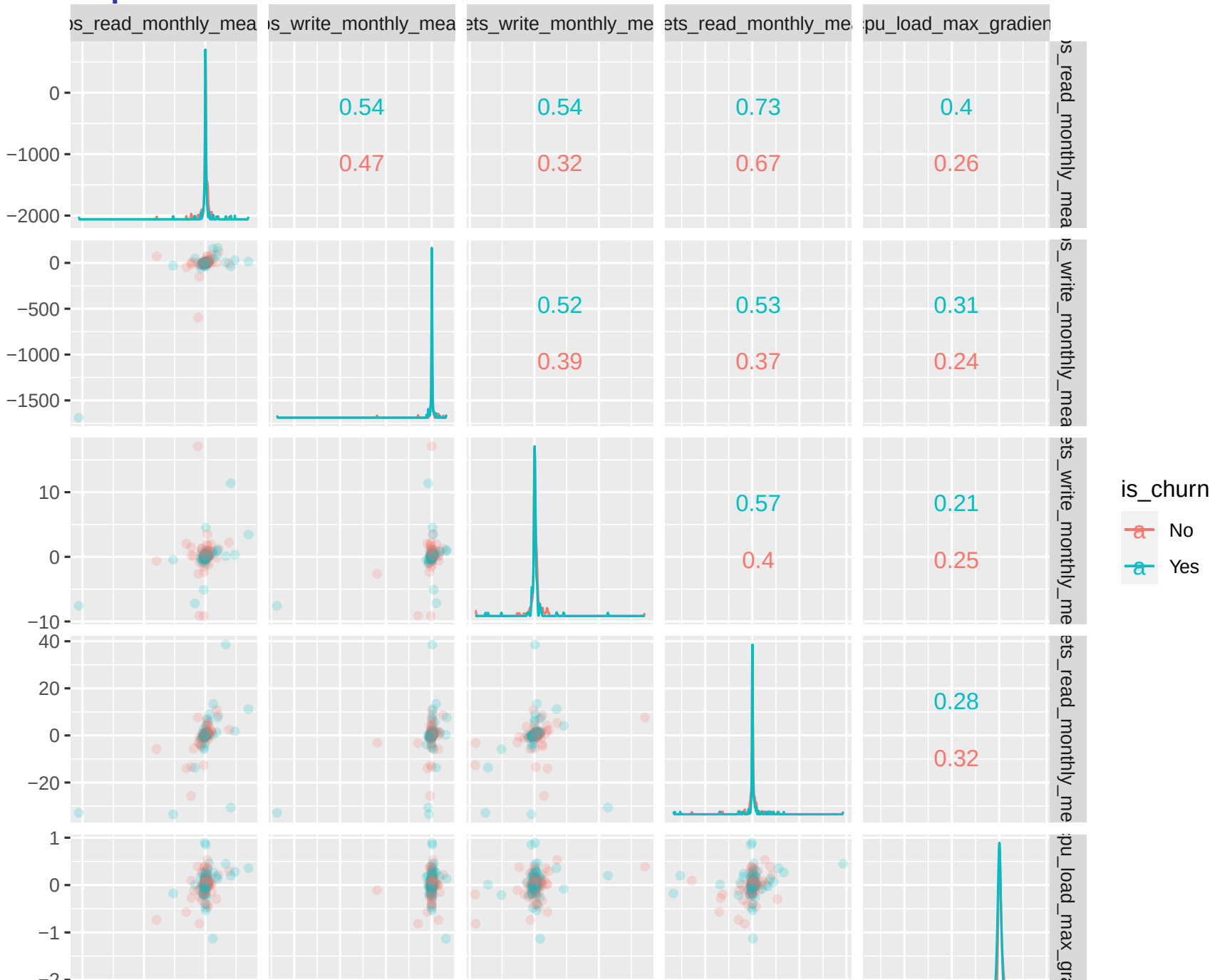
Data explorations



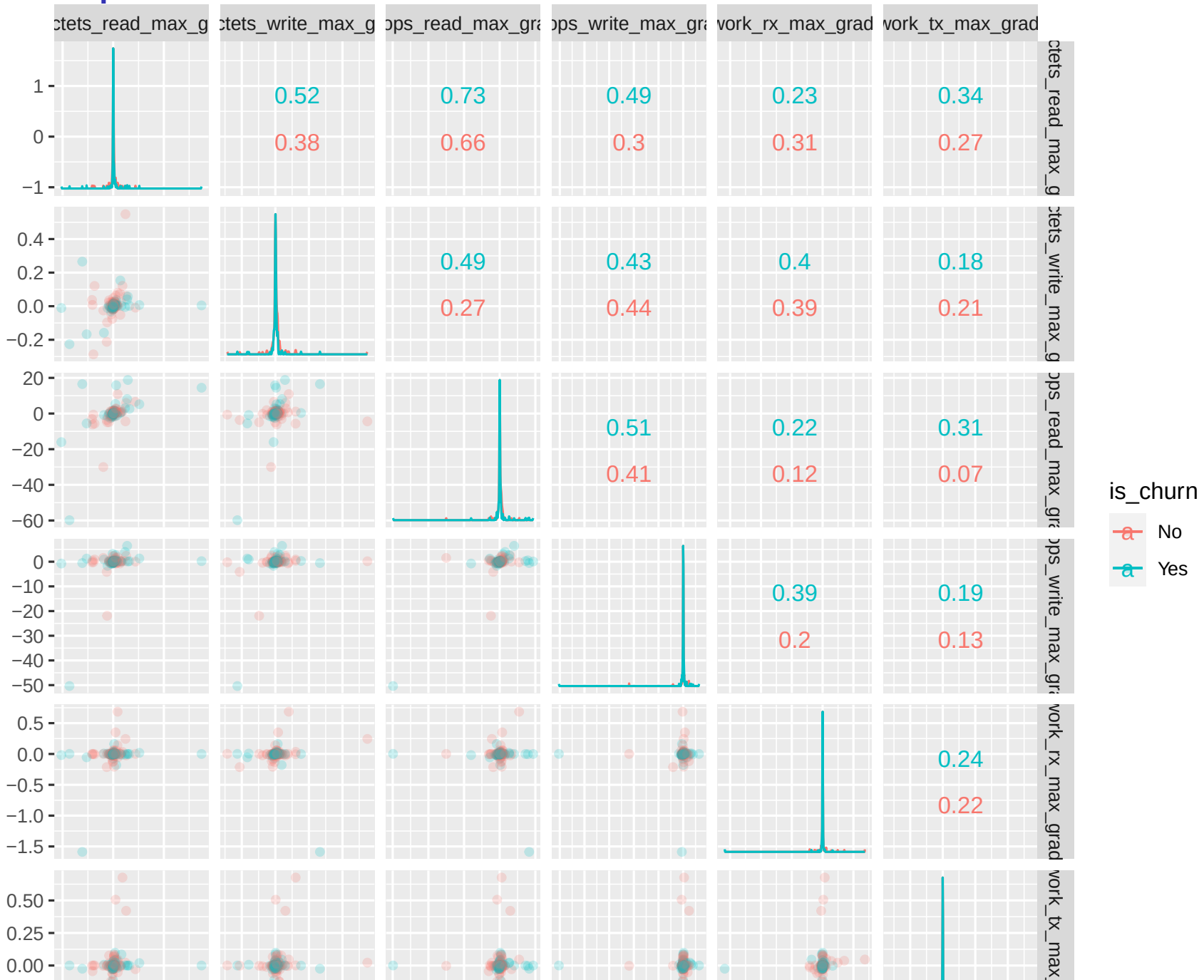
Data explorations cont.



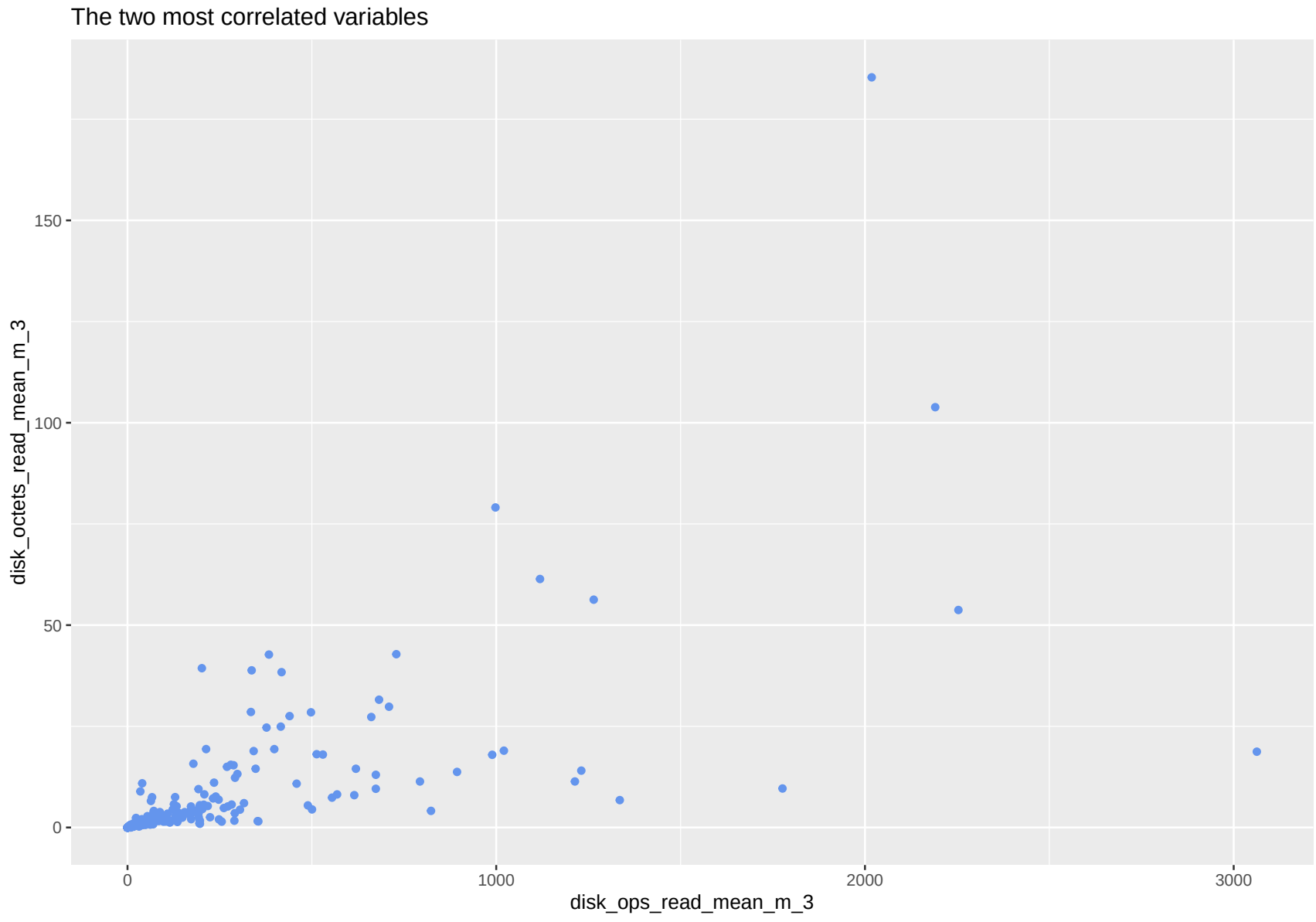
Data explorations cont.



Data explorations cont.



Data explorations cont.



Data preparation

```
# split data and write data preparation recipe
set.seed(1234)
vps_split  <- vps %>% initial_split(prop = 3/4, strata = is
train_data <- vps_split %>% training()
test_data  <- vps_split %>% testing()

# for model evaluation we will use k-fold cross validation
cv_folds <-
 vfold_cv(vps,
          v = 5,
          strata = is_churn)
```

Data preparation cont.

```
vps_recipe <-
  recipe(is_churn ~.,
         data = train_data) %>%
  step_rm(id) %>% # step remove id column
  step_log(all_nominal(), -all_outcomes()) %>%
  step_naomit(everything(), skip = TRUE) %>%
  step_novel(all_nominal(), -all_outcomes()) %>%
  step_normalize(all_numeric(), -all_outcomes()) %>%
  step_dummy(all_nominal(), -all_outcomes()) %>%
  step_zv(all_numeric(), -all_outcomes()) %>%
  step_corr(all_predictors(), threshold = 0.7, method = "sp
```

Other recipe

```
vps_recipe <- train_data %>%
  recipe(is_churn ~.) %>% # training formula
  step_rm(id) %>% # step remove id column
  # remove variables highly correlated with other vars
  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())
```

Data preparation cont.

```
prepped_data <-
  vps_recipe %>% # use the recipe object
  prep() %>% # perform the recipe on training data
  juice() # extract only the preprocessed dataframe

glimpse(prepped_data)
```

```
## Rows: 212
## Columns: 17
## $ cpu_load_mean_m_3                 <dbl> 8.47705465, 2.
## $ disk_octets_read_mean_m_3         <dbl> 2.137385e+00,
## $ disk_octets_write_mean_m_3        <dbl> 5.326178e+00,
## $ disk_ops_read_mean_m_3            <dbl> 6.209808e-01,
## $ disk_ops_write_mean_m_3           <dbl> 1.2264244271,
## $ network_rx_mean_m_3               <dbl> 0.1577688887,
## $ network_tx_mean_m_3               <dbl> 8.550132e-02,
## $ cpu_load_monthly_mean_delta       <dbl> 1.716186991, -
## $ network_tx_monthly_mean_delta     <dbl> -1.130669e-01,
## $ network_rx_monthly_mean_delta     <dbl> -0.1725863854
```

Train models

Null model

```
null_spec <- null_model() %>%
  set_engine("parsnip") %>%
  set_mode("classification")
null_wflow <-
  workflow() %>%
  add_recipe(vps_recipe) %>%
  add_model(null_spec)
null_res <-
  null_wflow %>%
  fit_resamples(
    resamples = cv_folds,
    metrics = metric_set(recall, precision, f_meas,
                         accuracy, kap, roc_auc, sens),
    control = control_resamples(save_pred = TRUE))
```

Null model - cont.

```
null_res %>%  collect_metrics(summarize = TRUE)
```

```
## # A tibble: 7 x 6
##   .metric   .estimator  mean     n std_err .config
##   <chr>     <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy  binary     0.523     5 0.00207 Preprocessor1
## 2 f_meas    binary     0.687     5 0.00179 Preprocessor1
## 3 kap       binary     0         5 0       Preprocessor1
## 4 precision binary     0.523     5 0.00207 Preprocessor1
## 5 recall    binary     1         5 0       Preprocessor1
## 6 roc_auc   binary     0.5       5 0       Preprocessor1
## 7 sens      binary     1         5 0       Preprocessor1
```

Logistic regression

```
log_spec <- # your model specification
  logistic_reg() %>%  # model type
  set_engine(engine = "glm") %>%  # model engine
  set_mode("classification") # model mode
log_wflow <- # new workflow object
  workflow() %>% # use workflow function
  add_recipe(vps_recipe) %>%   # use the new recipe
  add_model(log_spec)   # add your model spec
log_res <-
  log_wflow %>%
  fit_resamples(
    resamples = cv_folds,
    metrics = metric_set(recall, precision, f_meas,
      accuracy, kap, roc_auc, sens),
    control = control_resamples(
      save_pred = TRUE)
  )
```

Logistic regression - cont.

```
log_res %>%  collect_metrics(summarize = TRUE)
```

```
## # A tibble: 7 x 6
##   .metric   .estimator    mean     n std_err .config
##   <chr>     <chr>        <dbl> <int>   <dbl> <chr>
## 1 accuracy  binary       0.488     5  0.0199 Preprocesso
## 2 f_meas    binary       0.458     5  0.0303 Preprocesso
## 3 kap       binary      -0.0172    5  0.0390 Preprocesso
## 4 precision binary       0.512     5  0.0184 Preprocesso
## 5 recall    binary       0.420     5  0.0442 Preprocesso
## 6 roc_auc   binary       0.462     5  0.0141 Preprocesso
## 7 sens      binary       0.420     5  0.0442 Preprocesso
```

Random forest

```
library(ranger)
rf_spec <-
  rand_forest() %>%
  set_engine("ranger", importance = "impurity") %>%
  set_mode("classification")
rf_wflow <-
  workflow() %>%
  add_recipe(vps_recipe) %>%
  add_model(rf_spec)
rf_res <-
  rf_wflow %>%
  fit_resamples(
    resamples = cv_folds,
    metrics = metric_set( recall, precision, f_meas,
      accuracy, kap, roc_auc, sens),
    control = control_resamples(save_pred = TRUE)
  )
```

Random forest cont.

```
rf_res %>%  collect_metrics(summarize = TRUE)
```

```
## # A tibble: 7 x 6
##   .metric   .estimator  mean     n std_err .config
##   <chr>     <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy  binary     0.604     5  0.0174 Preprocessor1
## 2 f_meas    binary     0.643     5  0.0155 Preprocessor1
## 3 kap       binary     0.202     5  0.0351 Preprocessor1
## 4 precision binary     0.609     5  0.0157 Preprocessor1
## 5 recall    binary     0.682     5  0.0175 Preprocessor1
## 6 roc_auc   binary     0.609     5  0.0116 Preprocessor1
## 7 sens      binary     0.682     5  0.0175 Preprocessor1
```

XGBoost

```
library(xgboost)
xgb_spec <-
  boost_tree() %>%
  set_engine("xgboost") %>%
  set_mode("classification")
xgb_wflow <-
  workflow() %>%
  add_recipe(vps_recipe) %>%
  add_model(xgb_spec)
xgb_res <-
  xgb_wflow %>%
  fit_resamples(
    resamples = cv_folds,
    metrics = metric_set(
      recall, precision, f_meas,
      accuracy, kap,
      roc_auc, sens),
    control = control_resamples(save_pred = TRUE)
```

XGBoost cont.

```
xgb_res %>% collect_metrics(summarize = TRUE)
```

```
## # A tibble: 7 x 6
##   .metric   .estimator   mean     n std_err .config
##   <chr>     <chr>       <dbl> <int>   <dbl> <chr>
## 1 accuracy  binary      0.548     5 0.0105  Preprocesso
## 2 f_meas    binary      0.571     5 0.0214  Preprocesso
## 3 kap       binary      0.0915    5 0.0190  Preprocesso
## 4 precision binary      0.564     5 0.00852 Preprocesso
## 5 recall    binary      0.58      5 0.0347  Preprocesso
## 6 roc_auc   binary      0.566     5 0.00898 Preprocesso
## 7 sens      binary      0.58      5 0.0347  Preprocesso
```

K-nearest neighbor

```
knn_spec <-
  nearest_neighbor(neighbors = 4) %>% # we can adjust the n
  set_engine("kknn") %>%
  set_mode("classification")
knn_wflow <-
  workflow() %>%
  add_recipe(vps_recipe) %>%
  add_model(knn_spec)
knn_res <-
  knn_wflow %>%
  fit_resamples(
    resamples = cv_folds,
    metrics = metric_set(recall, precision, f_meas,
      accuracy, kap, roc_auc, sens),
    control = control_resamples(save_pred = TRUE))
```

```
## Warning: package 'kknn' was built under R version 4.1.1
```

K-nearest neighbor cont.

```
knn_res %>% collect_metrics(summarize = TRUE)
```

```
## # A tibble: 7 x 6
##   .metric   .estimator  mean     n std_err .config
##   <chr>     <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy  binary     0.555     5 0.00947 Preprocessor1
## 2 f_meas    binary     0.561     5 0.00851 Preprocessor1
## 3 kap       binary     0.109     5 0.0218  Preprocessor1
## 4 precision binary     0.582     5 0.0135  Preprocessor1
## 5 recall    binary     0.547     5 0.0255  Preprocessor1
## 6 roc_auc   binary     0.574     5 0.0145  Preprocessor1
## 7 sens      binary     0.547     5 0.0255  Preprocessor1
```

Neural network

```
library(keras)
nnet_spec <-
  mlp() %>%
  set_mode("classification") %>%
  set_engine("keras", verbose = 0)
nnet_wflow <-
  workflow() %>%
  add_recipe(vps_recipe) %>%
  add_model(nnet_spec)
nnet_res <-
  nnet_wflow %>%
  fit_resamples(
    resamples = cv_folds,
    metrics = metric_set(recall, precision, f_meas,
      accuracy, kap,roc_auc, sens),
    control = control_resamples(save_pred = TRUE))
```

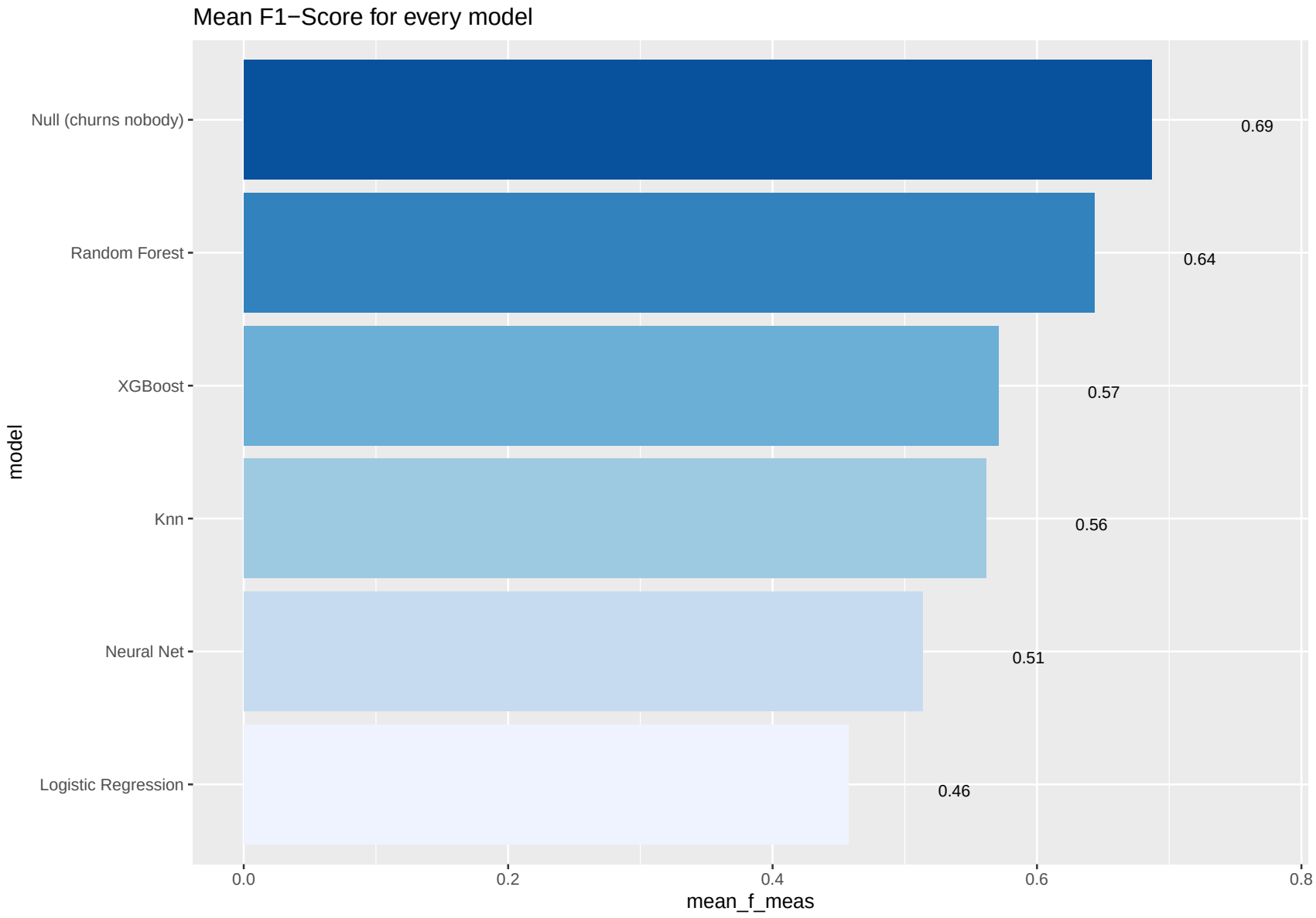
Neural network cont.

```
nnet_res %>% collect_metrics(summarize = TRUE)
```

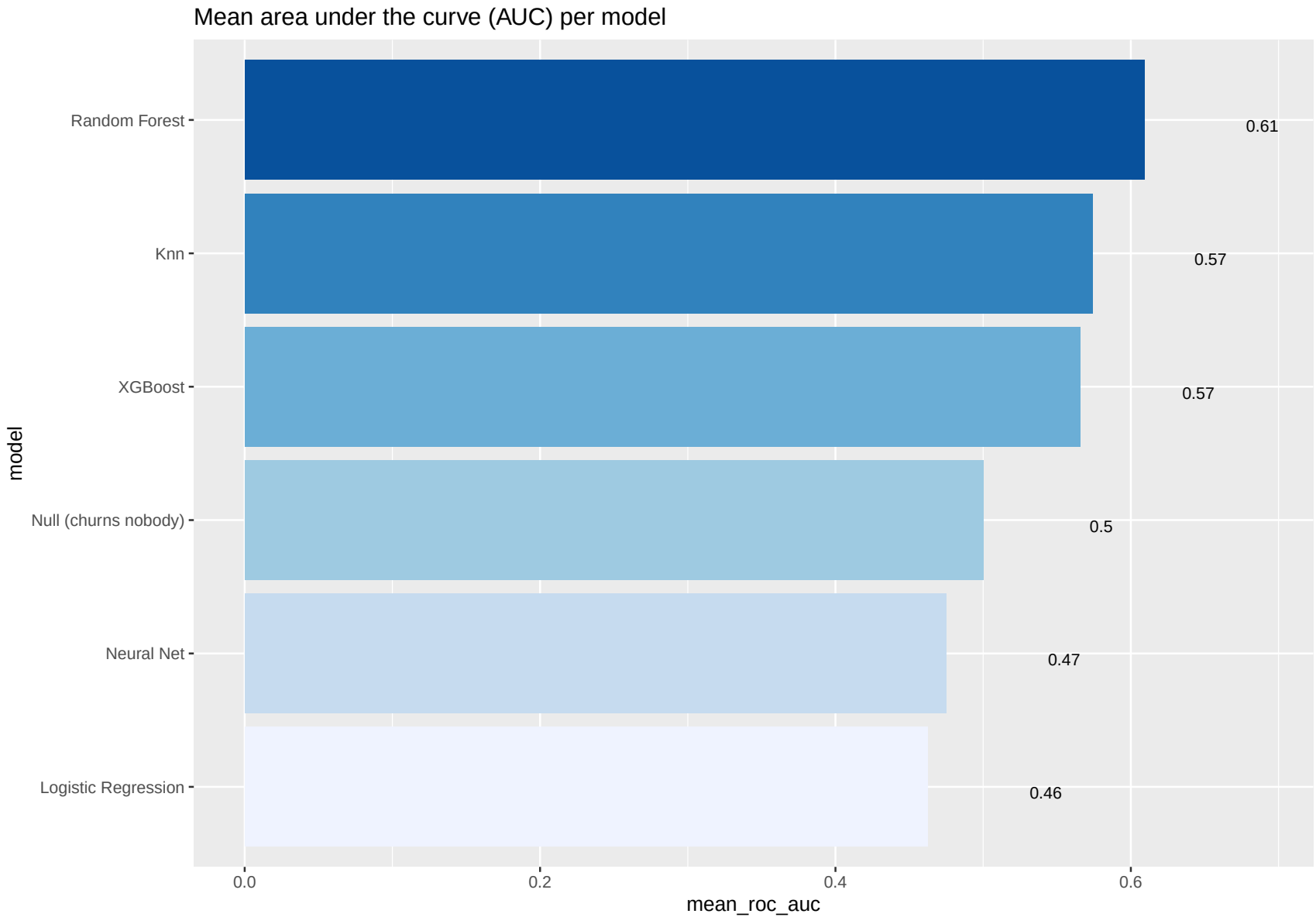
```
## # A tibble: 7 x 6
##   .metric   .estimator    mean     n std_err .config
##   <chr>     <chr>        <dbl> <int>   <dbl> <chr>
## 1 accuracy  binary       0.502     5 0.00653 Preprocesso
## 2 f_meas    binary       0.514     5 0.119   Preprocesso
## 3 kap       binary      -0.0143    5 0.0133  Preprocesso
## 4 precision binary       0.611     5 0.0974  Preprocesso
## 5 recall    binary       0.683     5 0.192   Preprocesso
## 6 roc_auc   binary       0.475     5 0.0417  Preprocesso
## 7 sens      binary       0.683     5 0.192   Preprocesso
```

Evaluate models

Compare models



Compare models - cont.



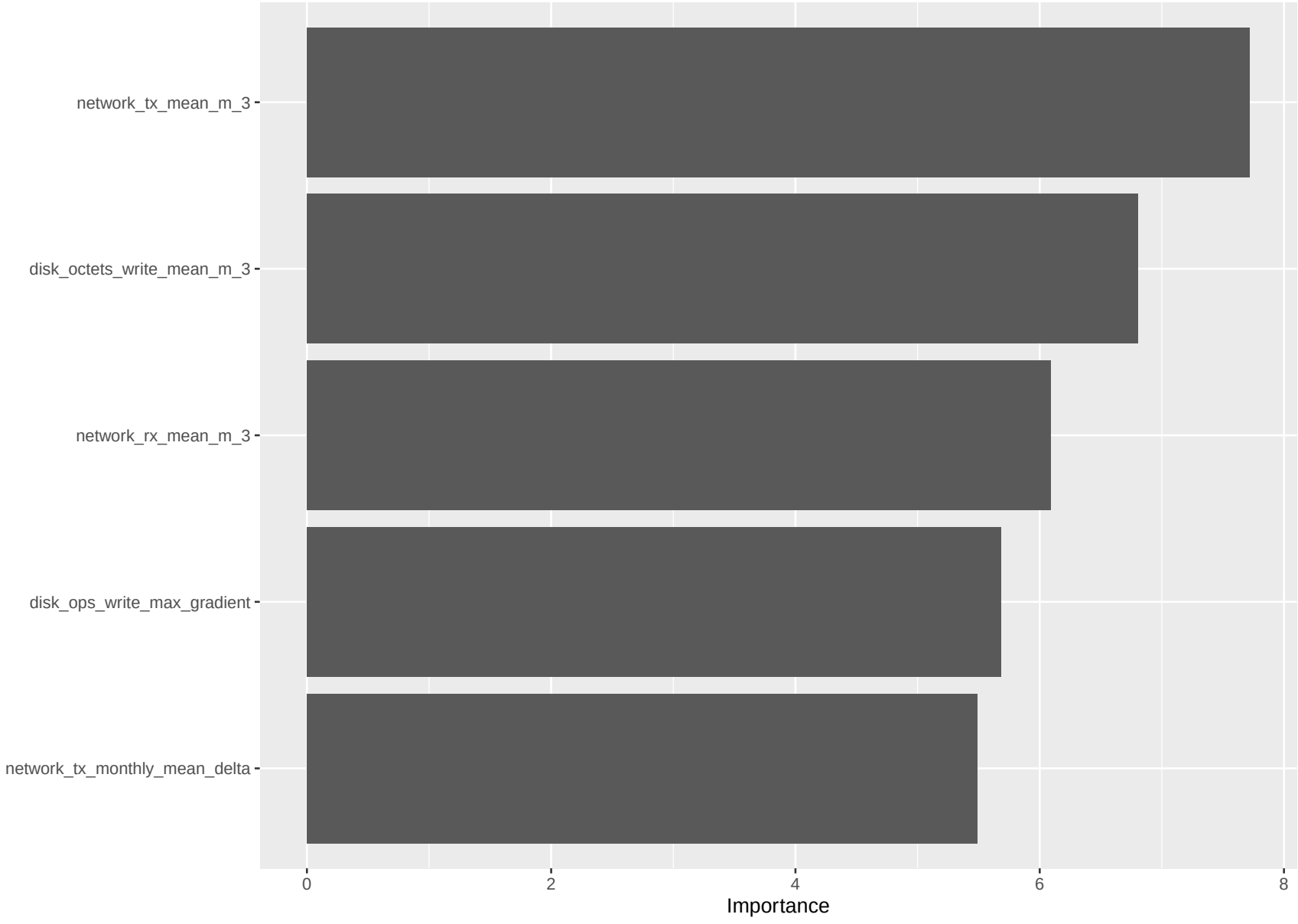
The best model

Metrics on test data

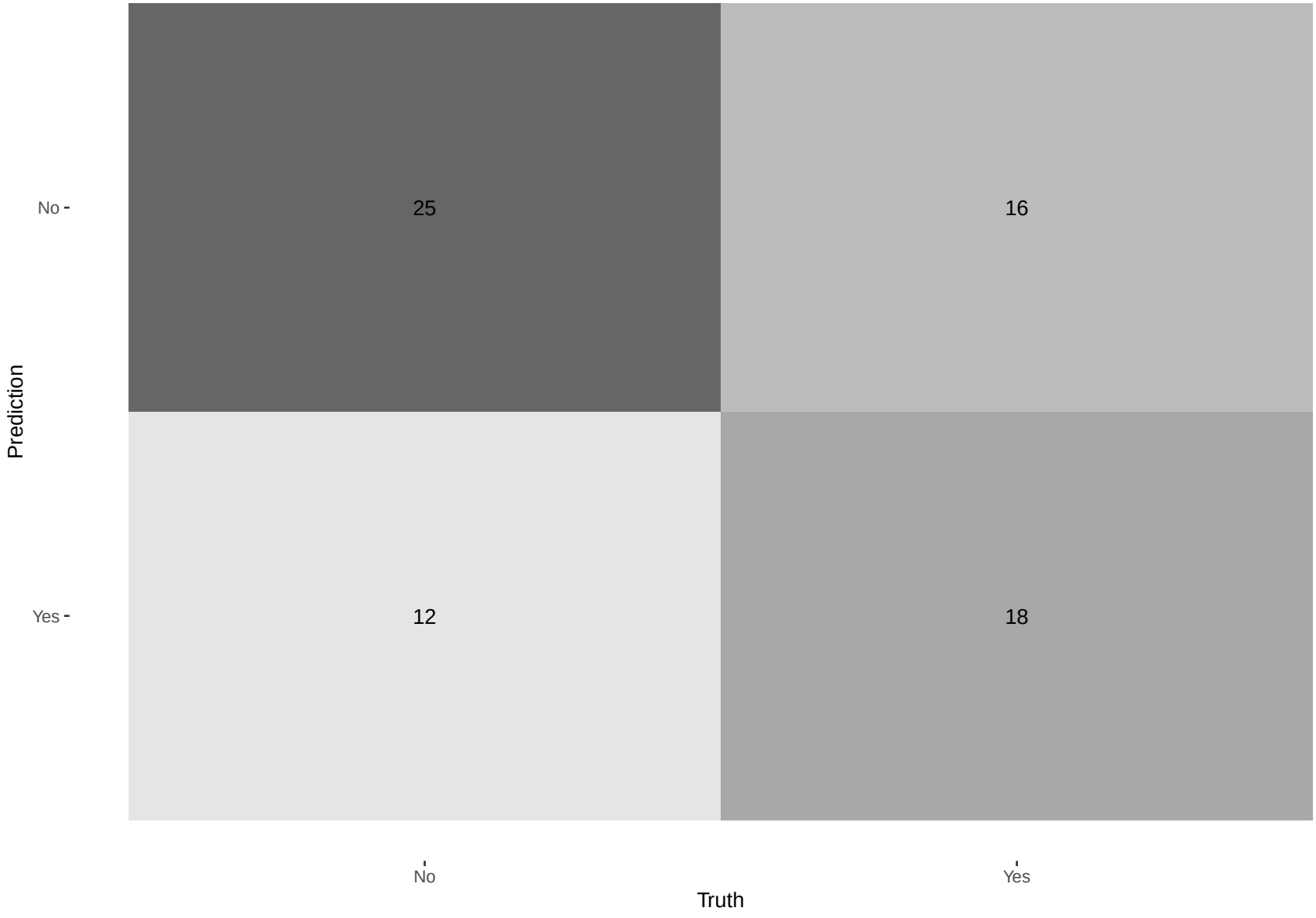
```
last_fit_rf <- last_fit(rf_wflow,
                        split = vps_split,
                        metrics = metric_set(recall, precis
                          accuracy, kap, roc_auc, sens))
last_fit_rf %>% collect_metrics(summarize = TRUE)
```

```
## # A tibble: 7 x 4
##   .metric   .estimator .estimate .config
##   <chr>     <chr>          <dbl> <chr>
## 1 recall    binary         0.676 Preprocessor1_Model1
## 2 precision binary         0.610 Preprocessor1_Model1
## 3 f_meas    binary         0.641 Preprocessor1_Model1
## 4 accuracy  binary         0.606 Preprocessor1_Model1
## 5 kap       binary         0.206 Preprocessor1_Model1
## 6 sens      binary         0.676 Preprocessor1_Model1
## 7 roc_auc   binary         0.692 Preprocessor1_Model1
```

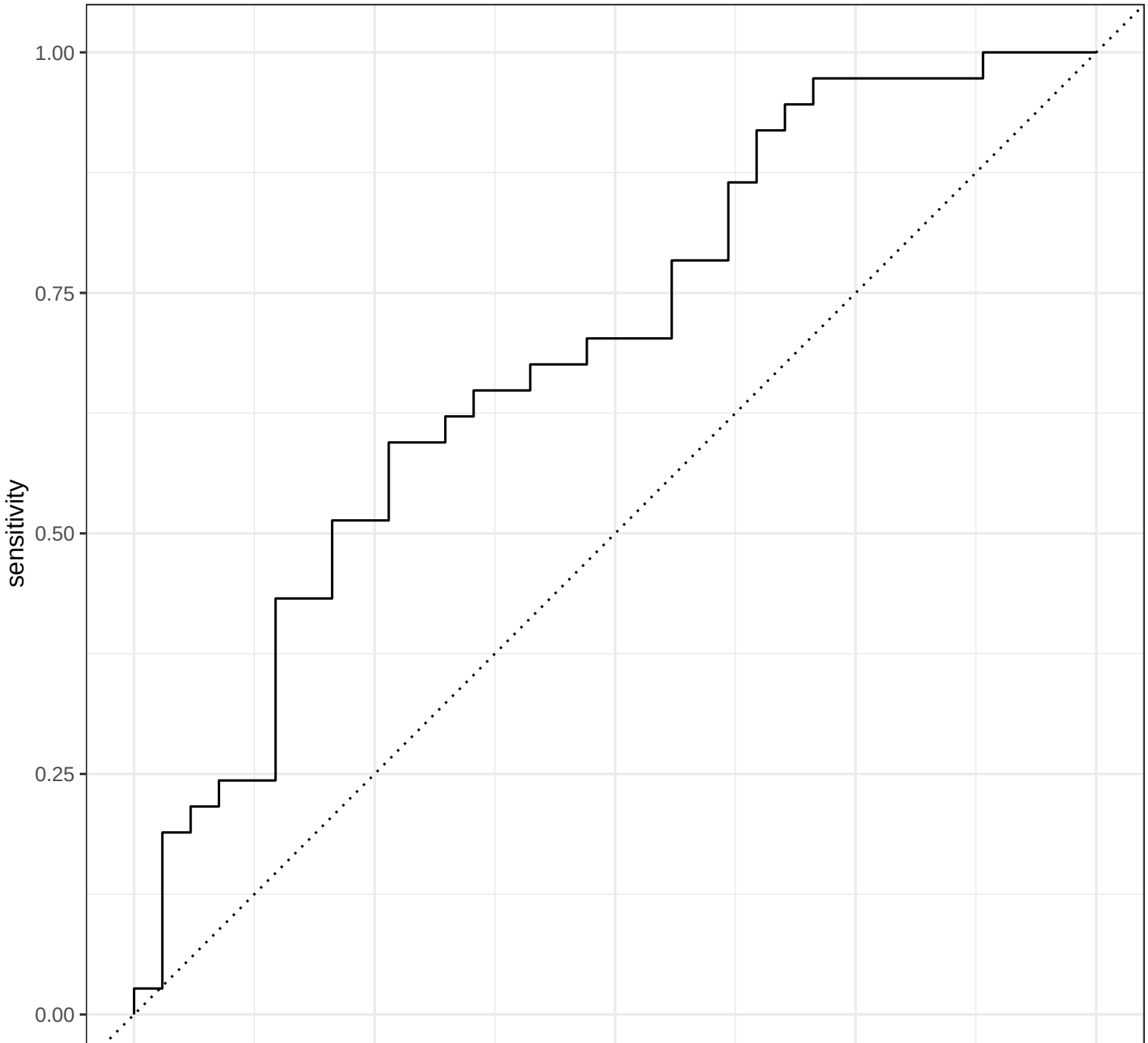
Variable importance



Confusion matrix



ROC curve



Model deployment

Model deployment

```
production <- read_csv("./data/vps_test_data.txt")
tmp <- rf_wflow %>% fit(train_data) %>% 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_2.txt")
```

Further steps

- feature engineering - combining and transforming further existing variables,
- tuning parameters of already tested algorithms used to train models,
- demonstrate how to use **SparkR** (R on Spark)