XGB

maycd

Contents


```
default_train <- read.csv("default_train.csv", stringsAsFactors = TRUE)
dim(default_train)  # dataset: default_train, response: bad_good
```

```
## [1] 171171     14
```

```
# number of features
n_features <- length(setdiff(names(default_train), "bad_good"))
```

XGBoost

```
xgb_prep <- recipe(bad_good ~ ., data = default_train) %>%
  step_integer(all_nominal()) %>%
  prep(training = default_train, retain = TRUE) %>%
  juice()

X <- as.matrix(xgb_prep[setdiff(names(xgb_prep), "bad_good")])
Y <- xgb_prep$bad_good
```

```
# before tuning
tic()
set.seed(123)
default_xgb1 <- xgb.cv(
  data = X,
  label = Y,
  nrounds = 2500,
  objective = "binary:logistic", # logistic regression binary classification, output probability
  early_stopping_rounds = 5,
  nfold = 10,
  params = list(
    eta = 0.01,
    max_depth = 7,
    min_child_weight = 2,
    subsample = 0.8,
    colsample_bytree = 0.9),
```

```
  eval_metric = "error",
  verbose = 0
)

# minimum test CV RMSE
min(default_xgb1$evaluation_log$test_error_mean)
```

```
## [1] 0
```

```
toc()
```

```
## 2.58 sec elapsed
```

```
# final model
tic()
set.seed(123)
default_xgb_final <- xgboost(
  data = X,
  label = Y,
  nrounds = 2500,
  objective = "binary:logistic",
  early_stopping_rounds = 5,
  verbose = 0,
  params = list(
    eta = 0.1,
    max_depth = 7,
    min_child_weight = 2,
    subsample = 0.8,
    colsample_bytree = 0.9,
    gamma = 0,
    lambda = 1,
    alpha = 0
  ),
  eval_metric = "error"
)
default_xgb_final
```

```
## ##### xgb.Booster
## raw: 6.8 Kb
## call:
##   xgb.train(params = params, data = dtrain, nrounds = nrounds,
##     watchlist = watchlist, verbose = verbose, print_every_n = print_every_n,
##     early_stopping_rounds = early_stopping_rounds, maximize = maximize,
##     save_period = save_period, save_name = save_name, xgb_model = xgb_model,
##     callbacks = callbacks, objective = "binary:logistic", eval_metric = "error")
## params (as set within xgb.train):
##   eta = "0.1", max_depth = "7", min_child_weight = "2", subsample = "0.8", colsample_bytree = "0.9",
## xgb.attributes:
##   best_iteration, best_msg, best_ntreelimit, best_score, niter
## callbacks:
##   cb.evaluation.log()
##   cb.early.stop(stopping_rounds = early_stopping_rounds, maximize = maximize,
##     verbose = verbose)
## # of features: 13
## niter: 6
```

```
## best_iteration : 1
## best_ntreelimit : 1
## best_score : 0
## best_msg : [1]   train-error:0.000000
## nfeatures : 13
## evaluation_log:
##     iter train_error
##        1           0
##        2           0
## ---
##        5           0
##        6           0
```

```
toc()
```

```
## 0.24 sec elapsed
```

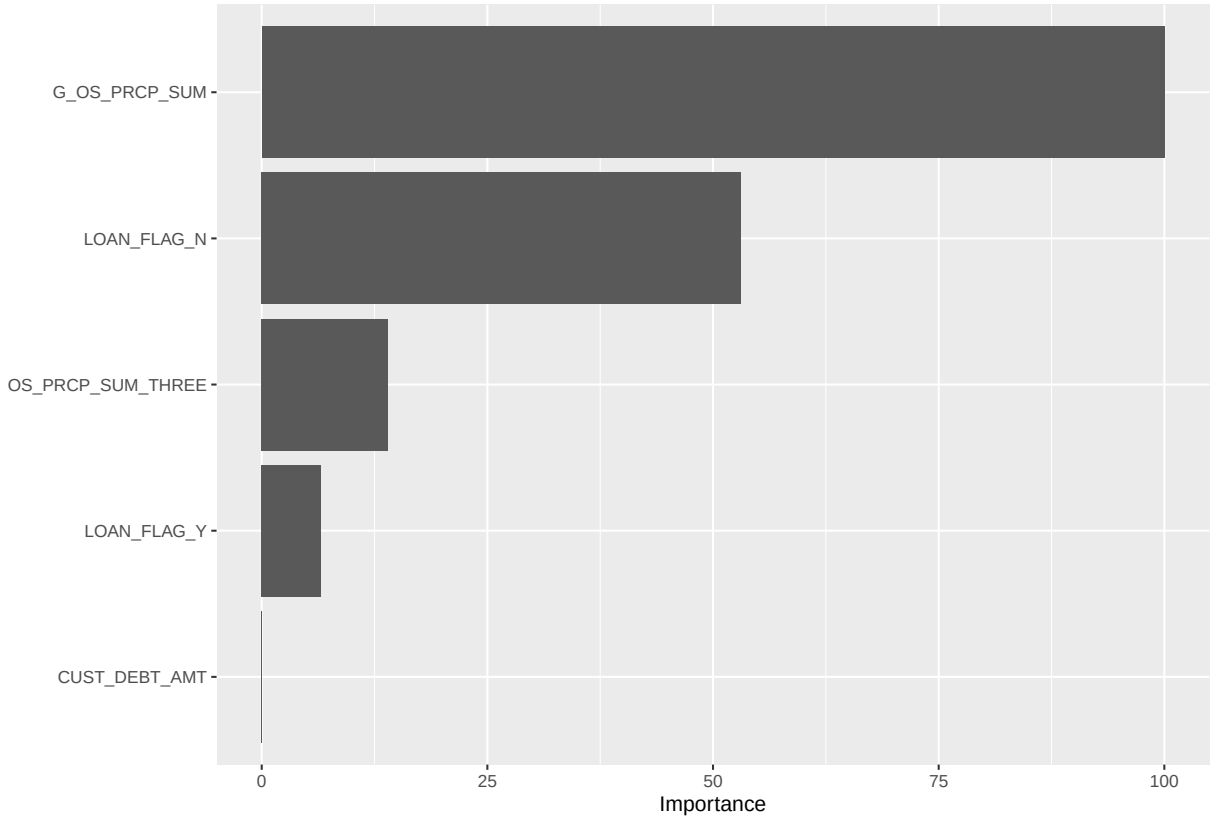
```
save(default_xgb_final, file = "default_xgb_final.rda")
```

Variable importance

```
vi_scores <- vi(default_xgb_final)
vi_scores
```

```
## # A tibble: 5 x 2
##   Variable          Importance
##   <chr>                  <dbl>
## 1 G_OS_PRCP_SUM       0.576
## 2 LOAN_FLAG_N         0.306
## 3 OS_PRCP_SUM_THREE   0.0805
## 4 LOAN_FLAG_Y         0.0379
## 5 CUST_DEBT_AMT       0.000332
```

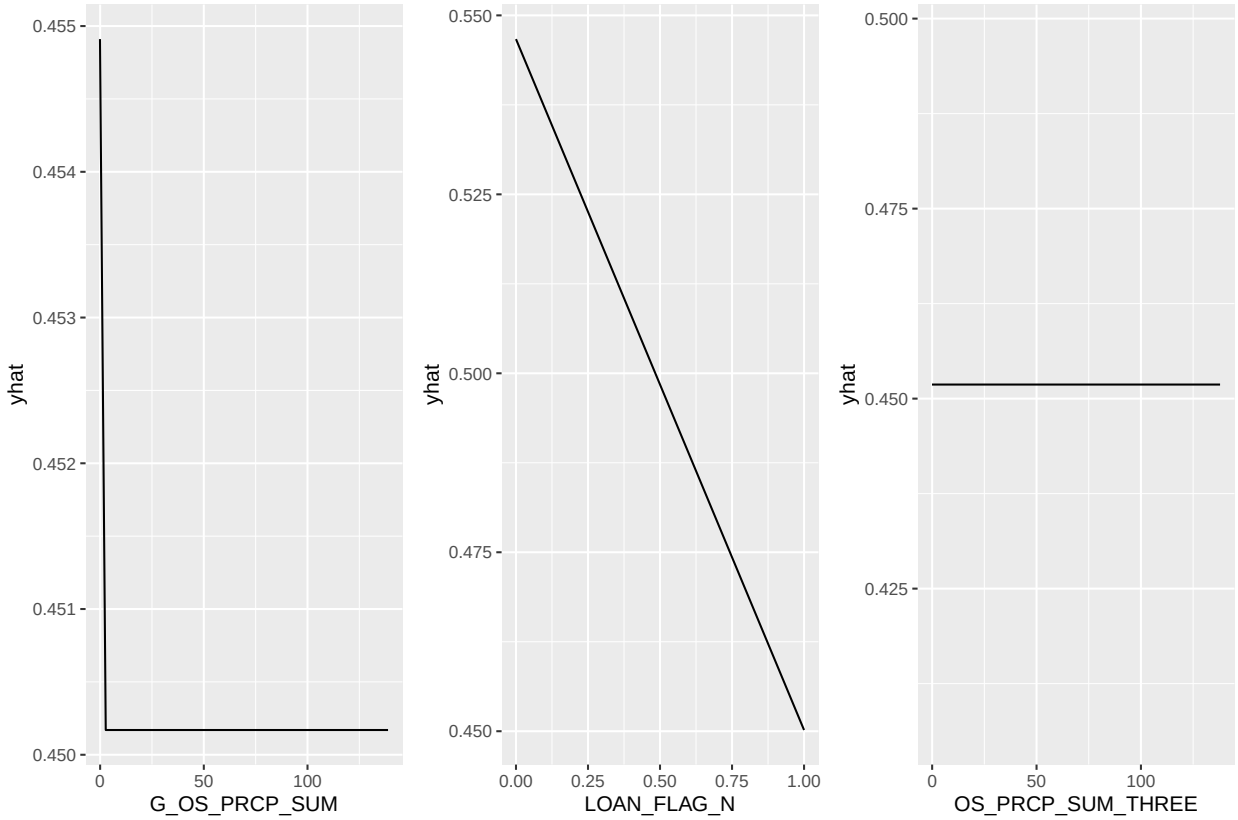
```
vip(default_xgb_final, num_features = 10, scale = TRUE)
```



PDP plots

```
tic()
p1 <- partial(default_xgb_final, pred.var = vi_scores[[1, 1]],
              train = X, type = "regression") %>% autoplot()
p2 <- partial(default_xgb_final, pred.var = vi_scores[[2, 1]],
              train = X, type = "regression") %>% autoplot()
p3 <- partial(default_xgb_final, pred.var = vi_scores[[3, 1]],
              train = X, type = "regression") %>% autoplot()
grid.arrange(p1, p2, p3, ncol = 3)
```

```
## Warning: Use of `object[[1L]]` is discouraged. Use `.data[[1L]]` instead.

## Warning: Use of `object[["yhat"]]` is discouraged. Use `.data[["yhat"]]`
## instead.

## Warning: Use of `object[[1L]]` is discouraged. Use `.data[[1L]]` instead.

## Warning: Use of `object[["yhat"]]` is discouraged. Use `.data[["yhat"]]`
## instead.

## Warning: Use of `object[[1L]]` is discouraged. Use `.data[[1L]]` instead.

## Warning: Use of `object[["yhat"]]` is discouraged. Use `.data[["yhat"]]`
## instead.
```




```
toc()
```

```
## 3.5 sec elapsed
```

Prediction

```
default_test <- read.csv("default_test.csv", stringsAsFactors = TRUE)
dim(default_test)  # dataset: default_test, response: bad_good
```

```
## [1] 114114     14
```

```
Y_pred <- predict(default_xgb_final, newdata = data.matrix(default_test[!names(default_test) %in% c("ba

table(Y_pred)
```

```
## Y_pred
## 0.450167536735535 0.450213372707367 0.549746870994568
##            108616              3504              1994
```

```
Y_pred_class <- ifelse(Y_pred < 0.5, 0, 1)

table(default_test$bad_good, Y_pred_class)
```

```
##    Y_pred_class
##          0      1
##   0 112120      1
##   1      0   1993
```

```r
mean(Y_pred_class - default_test$bad_good)
```

```
## [1] 8.763167e-06
```

```r
# precision = TP/(TP+FP)
pred_precision <- 1993/(1993+1)
pred_precision
```

```
## [1] 0.9994985
```

```r
# recall = TP/(TP+FN)
pred_recall <- 1993/(1993+0)
pred_recall
```

```
## [1] 1
```

```r
# F1 score
pred_f1 <- 2/(1/pred_precision + 1/pred_recall)
pred_f1
```

```
## [1] 0.9997492
```