Daatmin P6 - Latihan Hackathon.R

Ridson Alfarizal P

2022-09-30

```
library(tidyverse)
library(tidymodels)
library(themis)
tidymodels_prefer()

# Data ----------------------------------------------------------------
loc <- 'D:/data-mining-ta20222023'

df_train <- read.csv(file.path(loc, "training.csv"))
df_test <- read.csv(file.path(loc, "testing2.csv"), sep = ';')

glimpse(df_train)
## Rows: 18,890
## Columns: 18
## $ ID  <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 1
7, 18, 19,…
## $ X1  <int> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
 2, 2, 2, …
## $ X2  <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1, 1, …
## $ X3  <int> 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0,
 0, 1, 1, …
## $ X4  <int> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
 2, 2, 2, …
## $ X5  <int> 50, 60, 40, 30, 30, 45, 45, 45, 45, 48, 29, 29, 60, 45,
30, 43, 43…
## $ X6  <int> 22, 30, 19, 18, 18, 17, 17, 15, 15, 22, 12, 12, 27, 15,
13, 16, 14…
## $ X7  <int> 2, 2, 1, 1, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1, 1, …
## $ X8  <int> 23, 35, 20, 19, 19, 21, 21, 17, 17, 22, 14, 14, 28, 17,
13, 17, 15…
## $ X9  <int> 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
10, 10, 10…
## $ X10 <int> 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
10, 10, 10…
## $ X11 <int> 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1, 1, …
## $ X12 <int> 2, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 2, 1, 1, 1, 1,
 1, 1, 1, …
## $ X13 <int> 1, 1, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1, 2, 1, 2,
```

```
 2, 2, 1, …
## $ X14 <int> 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1, 0, …
## $ X15 <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2,
 2, 1, 1, …
## $ X16 <int> 1, 1, 1, 1, 1, 0, 0, 0, 0, 2, 1, 1, 1, 0, 1, 1, 0, 1, 1,
 1, 0, 0, …
## $ Y   <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1,
 2, 1, 1, …

glimpse(df_test)

## Rows: 3,799
## Columns: 17
## $ X.ID <int> 1, 5, 9, 10, 11, 12, 14, 20, 21, 29, 31, 32, 33, 34, 43,
 50, 60, …
## $ X1   <int> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
 2, 2, 2,…
## $ X2   <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1, 1,…
## $ X3   <int> 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0,…
## $ X4   <int> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
 2, 2, 1,…
## $ X5   <int> 50, 30, 45, 48, 29, 29, 45, 35, 35, 48, 48, 49, 40, 40,
 40, 16, 7…
## $ X6   <int> 22, 18, 15, 22, 12, 12, 15, 14, 17, 24, 17, 14, 19, 19,
 14, 14, 1…
## $ X7   <int> 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1,
 1, 1, 1,…
## $ X8   <int> 23, 19, 17, 22, 14, 14, 17, 16, 16, 24, 18, 15, 20, 20,
 14, 15, 1…
## $ X9   <int> 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 0, 0, 1
0, 10, 0, …
## $ X10  <int> 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
 10, 10, 1…
## $ X11  <int> 2, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 2, 2, 2, 2, 1, 1, 2, 1,
 1, 1, 1,…
## $ X12  <int> 2, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 2, 2,
 2, 1, 2,…
## $ X13  <int> 1, 1, 2, 2, 2, 2, 1, 2, 2, 1, 1, 1, 1, 2, 2, 2, 1, 2, 1,
 1, 1, 2,…
## $ X14  <int> 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1,
 1, 1, 1,…
## $ X15  <int> 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1, 1,…
## $ X16  <int> 1, 1, 0, 2, 1, 1, 0, 1, 0, 2, 2, 1, 1, 1, 0, 1, 1, 1, 1,
 0, 0, 1,…
```

```r
# Data Prepocessing ------------------------------------------------
unique(df_train$Y)
```

```
## [1] 1 2
```

```r
df_train <- df_train %>%
  mutate(Y = as.factor(Y)) %>%
  mutate_at(vars(X1:X4, X7, X9:X16), ~ as.factor(.x))

df_test <- df_test %>%
  mutate_at(vars(X1:X4, X7, X9:X16), ~ as.factor(.x)) %>%
  rename(ID = X.ID)

df_train %>%
  glimpse()
```

```
## Rows: 18,890
## Columns: 18
## $ ID  <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 1
7, 18, 19,…
## $ X1  <fct> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
 2, 2, 2, …
## $ X2  <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1, 1, …
## $ X3  <fct> 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0,
 0, 1, 1, …
## $ X4  <fct> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
 2, 2, 2, …
## $ X5  <int> 50, 60, 40, 30, 30, 45, 45, 45, 45, 48, 29, 29, 60, 45,
30, 43, 43…
## $ X6  <int> 22, 30, 19, 18, 18, 17, 17, 15, 15, 22, 12, 12, 27, 15,
13, 16, 14…
## $ X7  <fct> 2, 2, 1, 1, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1, 1, …
## $ X8  <int> 23, 35, 20, 19, 19, 21, 21, 17, 17, 22, 14, 14, 28, 17,
13, 17, 15…
## $ X9  <fct> 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
10, 10, 10…
## $ X10 <fct> 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
10, 10, 10…
## $ X11 <fct> 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1, 1, …
## $ X12 <fct> 2, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 2, 1, 1, 1, 1,
 1, 1, 1, …
## $ X13 <fct> 1, 1, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1, 2, 1, 2,
 2, 2, 1, …
## $ X14 <fct> 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1, 0, …
## $ X15 <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2,
 2, 1, 1, …
```

```
## $ X16 <fct> 1, 1, 1, 1, 1, 0, 0, 0, 0, 2, 1, 1, 1, 0, 1, 1, 0, 1, 1,
 1, 0, 0, …
## $ Y   <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1,
 2, 1, 1, …
```

```
# Metric ------------------------------------------------------------
----
my_metric <- metric_set(accuracy)


# train test split --------------------------------------------------
----
set.seed(1)
splits <- initial_split(df_train, prop = 0.7, strata = Y)
splits
```

```
## <Training/Testing/Total>
## <13222/5668/18890>
```

```
train_set <- training(splits)
test_set  <- testing(splits)

set.seed(1)
train_fold <- vfold_cv(train_set, v = 3, strata = Y)



# EDA ---------------------------------------------------------------
----
colSums(is.na(df_train))
```

```
##  ID  X1  X2  X3  X4  X5  X6  X7  X8  X9 X10 X11 X12 X13 X14 X15 X16
  Y
##   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0
```

```
# model -------------------------------------------------------------
----
logreg_spec <-
  logistic_reg(penalty = tune(), mixture = tune()) %>%
  set_engine('glmnet')

rf_spec <-
  rand_forest(mtry = tune(), min_n = tune(), trees = tune()) %>%
  set_engine('ranger') %>%
  set_mode('classification')

xgb_spec <-
  boost_tree(tree_depth = tune(), trees = tune(), learn_rate = tune(),
min_n = tune(), loss_reduction = tune(), sample_size = tune(), stop_ite
```

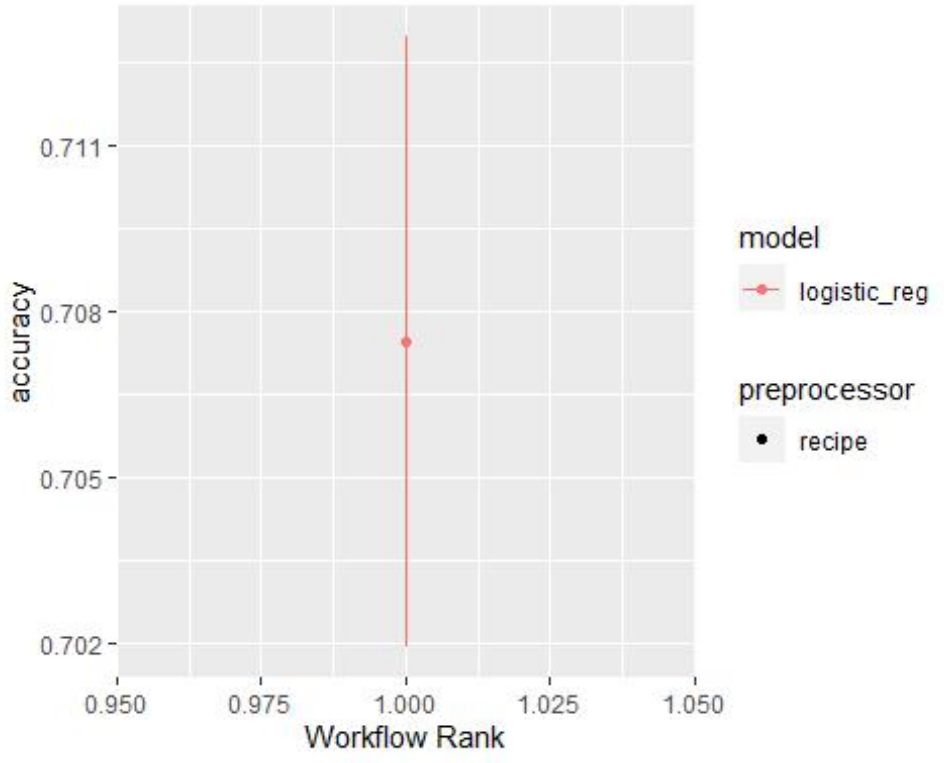
```
r = tune()) %>%
  set_engine('xgboost') %>%
  set_mode('classification')

mlp_spec <-
  mlp(hidden_units = tune(), penalty = tune(), epochs = tune()) %>%
  set_engine('nnet') %>%
  set_mode('classification')



svm_rbf_spec <-
  svm_rbf(cost = tune(), rbf_sigma = tune(), margin = tune()) %>%
  set_engine('kernlab') %>%
  set_mode('classification')

svm_poly_spec <-
  svm_poly(cost = tune(), degree = tune(), scale_factor = tune(), margi
n = tune()) %>%
  set_engine('kernlab') %>%
  set_mode('classification')

# recipe and workflow ----------------------------------------------
----
my_recipe <-
  recipe(Y ~ ., data = train_set) %>%
  update_role(ID, new_role = "id") %>%
  step_dummy(all_nominal_predictors()) %>%
  step_zv() %>%
  step_smote(Y, over_ratio = 0.9)


my_workflow <-
  workflow_set(
    preproc = list(my_recipe),
    models = list(
      logreg = logreg_spec
      # nnet = mlp_spec,
      # xgb = xgb_spec,
      # rf = rf_spec
      # svm_rbf = svm_rbf_spec
      # svm_poly = svm_poly_spec
    )
  )


# tuning -----------------------------------------------------------
----
grid_ctrl <-
```

```
  control_grid(
    # parallel_over = "resampling",
    save_pred = TRUE,
    verbose = T,
    allow_par = T
  )


library(doParallel)

## Loading required package: foreach

##
## Attaching package: 'foreach'

## The following objects are masked from 'package:purrr':
##
##     accumulate, when

## Loading required package: iterators

## Loading required package: parallel

all_cores <- parallel::detectCores(logical = FALSE)
cl <- makePSOCKcluster(all_cores)
registerDoParallel(cl)

grid_results <-
  my_workflow %>%
  workflow_map(
    seed = 1,
    resamples = train_fold,
    grid = 50,
    control = grid_ctrl,
    metrics = my_metric,
    verbose = TRUE
  )

## i 1 of 1 tuning:     recipe_logreg

## ✔ 1 of 1 tuning:     recipe_logreg (1m 44.8s)

# res1 <- grid_results
# res2 <- grid_results

# Hasil ------------------------------------------------------------------
----
metric_name <- 'accuracy'

grid_results %>%
  rank_results() %>%
```

```
  filter(.metric == metric_name) %>%
  select(model, .config, accuracy = mean, rank)
```

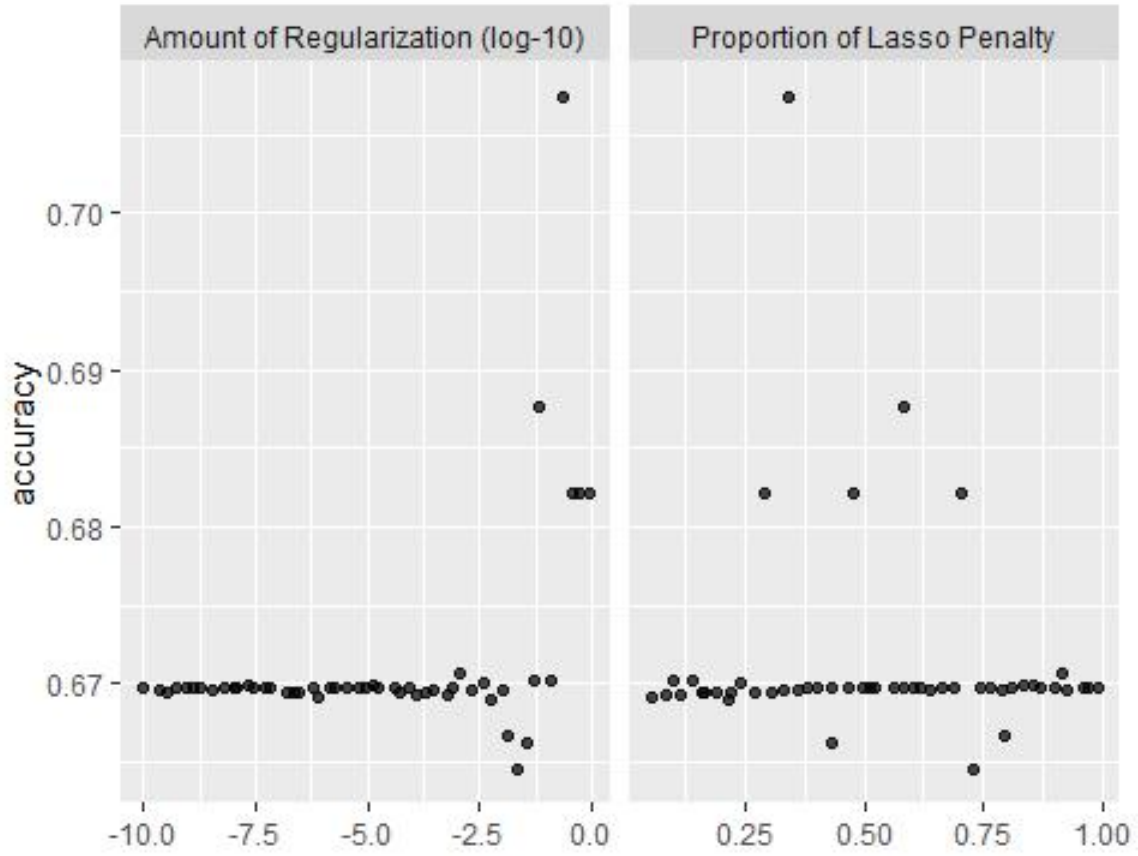
```
## # A tibble: 50 × 4
##    model        .config               accuracy  rank
##    <chr>        <chr>                    <dbl> <int>
##  1 logistic_reg Preprocessor1_Model16    0.707     1
##  2 logistic_reg Preprocessor1_Model28    0.688     2
##  3 logistic_reg Preprocessor1_Model23    0.682     3
##  4 logistic_reg Preprocessor1_Model13    0.682     4
##  5 logistic_reg Preprocessor1_Model35    0.682     5
##  6 logistic_reg Preprocessor1_Model46    0.671     6
##  7 logistic_reg Preprocessor1_Model05    0.670     7
##  8 logistic_reg Preprocessor1_Model03    0.670     8
##  9 logistic_reg Preprocessor1_Model11    0.670     9
## 10 logistic_reg Preprocessor1_Model42    0.670    10
## # … with 40 more rows
```

```
autoplot(
  grid_results,
  rank_metric = metric_name,  # <- how to order models
  metric = metric_name,       # <- which metric to visualize
  select_best = TRUE     # <- one point per workflow
)
```




```
autoplot(
  grid_results,
```

```
  id = "recipe_logreg",
  metric = metric_name
)
```




```
# finalizing model ----------------------------------------------------
----
best_model <- 'recipe_logreg'

best_results <-
  grid_results %>%
  extract_workflow_set_result(best_model) %>%
  select_best(metric = metric_name)

best_results

## # A tibble: 1 × 3
##   penalty mixture .config
##     <dbl>   <dbl> <chr>
## 1   0.226   0.343 Preprocessor1_Model16

final_wf <-
  grid_results %>%
  extract_workflow(best_model) %>%
```

```
  finalize_workflow(best_results)

final_wf

## == Workflow ============================================================
==========
## Preprocessor: Recipe
## Model: logistic_reg()
##
## -- Preprocessor ----------------------------------------------------------
----------
## 3 Recipe Steps
##
## • step_dummy()
## • step_zv()
## • step_smote()
##
## -- Model -----------------------------------------------------------------
----------
## Logistic Regression Model Specification (classification)
##
## Main Arguments:
##   penalty = 0.225892019025795
##   mixture = 0.343290190990083
##
## Computational engine: glmnet

final_fit <- final_wf %>% fit(df_train)
final_fit

## == Workflow [trained] ==================================================
==========
## Preprocessor: Recipe
## Model: logistic_reg()
##
## -- Preprocessor ----------------------------------------------------------
----------
## 3 Recipe Steps
##
## • step_dummy()
## • step_zv()
## • step_smote()
##
## -- Model -----------------------------------------------------------------
----------
##
## Call:  glmnet::glmnet(x = maybe_matrix(x), y = y, family = "binomial
",      alpha = ~0.343290190990083)
##
##    Df  %Dev  Lambda
```

```
## 1   0  0.00 0.32790
## 2   1  0.36 0.29880
## 3   3  0.94 0.27230
## 4   3  1.73 0.24810
## 5   3  2.46 0.22600
## 6   3  3.14 0.20600
## 7   3  3.75 0.18770
## 8   3  4.31 0.17100
## 9   4  4.85 0.15580
## 10  4  5.40 0.14200
## 11  5  5.90 0.12930
## 12  6  6.39 0.11790
## 13  7  6.86 0.10740
## 14  7  7.29 0.09785
## 15  8  7.71 0.08915
## 16  8  8.08 0.08123
## 17  8  8.41 0.07402
## 18  8  8.71 0.06744
## 19  8  8.96 0.06145
## 20  9  9.20 0.05599
## 21 10  9.41 0.05102
## 22 11  9.65 0.04649
## 23 11  9.88 0.04236
## 24 11 10.07 0.03859
## 25 11 10.24 0.03516
## 26 12 10.39 0.03204
## 27 13 10.52 0.02919
## 28 13 10.64 0.02660
## 29 13 10.74 0.02424
## 30 14 10.83 0.02208
## 31 16 10.93 0.02012
## 32 16 11.03 0.01833
## 33 17 11.12 0.01671
## 34 17 11.20 0.01522
## 35 17 11.27 0.01387
## 36 17 11.34 0.01264
## 37 18 11.39 0.01151
## 38 18 11.44 0.01049
## 39 18 11.48 0.00956
## 40 18 11.52 0.00871
## 41 18 11.55 0.00794
## 42 18 11.58 0.00723
## 43 19 11.60 0.00659
## 44 19 11.62 0.00600
## 45 19 11.64 0.00547
## 46 19 11.65 0.00498
##
## ...
## and 17 more lines.
```

```r
# Evaluasi ----------------------------------------------------------
----
# train
df_train %>%
  bind_cols(predict(final_fit, .)) %>%
  my_metric(truth = Y, estimate = .pred_class)
```

```
## # A tibble: 1 × 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.712
```

```r
df_train %>%
  bind_cols(predict(final_fit, .)) %>%
  conf_mat(truth = Y, estimate = .pred_class)
```

```
##           Truth
## Prediction     1     2
##          1 11725  4281
##          2  1160  1724
```

```r
# test
test_set %>%
  bind_cols(predict(final_fit, new_data = .)) %>%
  my_metric(truth = Y, estimate = .pred_class)
```

```
## # A tibble: 1 × 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.719
```

```r
test_set %>%
  bind_cols(predict(final_fit, .)) %>%
  conf_mat(truth = Y, estimate = .pred_class)
```

```
##           Truth
## Prediction    1    2
##          1 3548 1273
##          2  318  529
```

```r
# Prediction --------------------------------------------------------
----
preds <- predict(final_fit, new_data = df_test) %>% pull()

table(preds)
```

```
## preds
##    1    2
## 3213  586
```

```r
# Submission --------------------------------------------------------
----
```

```r
hasil <- df_test %>%
  select(ID) %>%
  mutate(Y = preds)

write.csv(hasil, 'E:/sub.csv', row.names = F, quote = F)

# system(
#   'kaggle competitions submit -c classification-data-challenge -f D:/
__Datasets/sub.csv -m "Message"'
# )
```