Homework4p2

2025-10-23

```
library(tidyverse)
library(lubridate)
library(randomForest)
library(pROC)

test <- read.csv("test_dataset.csv.gz")
train <- read.csv("train_dataset.csv.gz")
```

```
# Create features
featurize <- function(df) {
  df %>%
    mutate(
      # days between when appoint was made and when it's for
      days_between = as.numeric(
        as_datetime(appt_time) - as_datetime(appt_made), units = "days"),
      # hour of the day the appointment is at
      appt_hour = factor(hour(as_datetime(appt_time))),
      # What day of the week the appointment is on
      weekday = factor(wday(as_datetime(appt_time), label = TRUE, week_start = 1)),
      # If appt is on the weekend or not
      weekend = wday(as_datetime(appt_time), week_start = 1) %in% c(6,7)
    ) %>%
    mutate(
      address = factor(address),
      specialty = factor(specialty),
      provider_id = factor(provider_id),
      id = factor(id)
    )
}

train_x <- featurize(train)
test_x  <- featurize(test)

# Model matrix
model_vars <- c("provider_id","address","age","specialty",
                "days_between","appt_hour","weekday","weekend")
train_x <- train_x %>% select(all_of(model_vars), no_show)
test_x  <- test_x  %>% select(all_of(model_vars), no_show)
```

```
# model train and test set probabilities
set.seed(24)
rf <- randomForest(
  x = train_x %>% select(-no_show),
  y = factor(train_x$no_show),
```

```
  ntree = 500,
  mtry = floor(sqrt(ncol(train_x)-1)),
  nodesize = 5
)

# Probabilities on training
train_p <- as.numeric(predict(rf, newdata = train_x, type = "prob")[, "1"])
```

```
# Tries thresholds from 0.05 to 0.95 and picks the one with the
# lowest misclassification rate on the training set
search_thresh <- function(y_true, p_hat) {
  cand <- seq(0.05, 0.95, by = 0.01)
  errs <- sapply(cand, function(t) mean( (p_hat >= t) != as.logical(y_true) ))
  cand[which.min(errs)]
}
t_star <- search_thresh(train_x$no_show, train_p)
t_star
```

```
## [1] 0.55
```

```
# Check if overall error rate is less than 0.37
test_p <- as.numeric(predict(rf, newdata = test_x, type = "prob")[, "1"])
test_pred <- as.integer(test_p >= t_star)
overall_error <- mean(test_pred != test_x$no_show)
overall_error
```

```
## [1] 0.1135923
```

```
# Test confusion matrix + accuracy (1 - error)
table(Predicted = test_pred, Actual = test_x$no_show)
```

```
##           Actual
## Predicted     0     1
##         0 21475  2561
##         1  1600 10995
```

```
1 - overall_error
```

```
## [1] 0.8864077
```

```
# ROC AUC
pROC::auc(response = test_x$no_show, predictor = test_p)
```

```
## Area under the curve: 0.951
```

```
# Build a predictions table that keeps appt_time
test_feats <- featurize(test)

# Probabilities
```

```r
test_p <- as.numeric(
  predict(rf, newdata = test_feats %>% select(all_of(model_vars)), type = "prob")[, "1"])
test_pred <- as.integer(test_p >= t_star)

preds_out <- test_feats %>%
  mutate(prob_no_show = test_p,
         pred_no_show = test_pred) %>%
  # keep only what the Shiny app needs
  select(appt_time, provider_id, address, age, specialty, prob_no_show, pred_no_show)

# write the file the app will read
readr::write_csv(preds_out, "test_with_predictions.csv")
saveRDS(rf, "model_rf.rds")

saveRDS(list(model = rf, threshold = t_star), "model_rf_with_threshold.rds")
```