Homework4p2

2025-10-23

Model Development

The goal of the model is to predict whether a patient will not show up for their appointment.

The model outputs a probability between 0 and 1 for each appointment, indicating the likelihood of a no-show.

A binary prediction ("show" vs. "no-show") is then obtained by applying an optimal threshold chosen to minimize misclassification.

Data Preparation

Used the provided training dataset for model fitting and a separate test dataset for evaluation.

Engineered several time-based features to capture appointment scheduling patterns:

- days_between: number of days between booking and appointment.
- appt_hour: hour of the day the appointment occurs.
- weekday: day of the week.
- weekend: logical indicator if the appointment is on a Saturday or Sunday.

Converted categorical variables into factors so the model could interpret them properly.

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

# load in data
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 Choice and Justification

Selected a Random Forest classifier because it handles both numerical and categorical predictors, captures nonlinear relationships, and is robust to outliers.

I used a Random Forest with 500 trees, and at each split, the model randomly considered about the square root of the total number of features. This is a common, effective setup that helps the model generalize well instead of memorizing the training data.

Random Forest was chosen over logistic regression to capture complex interactions between features like day of week, provider, and patient demographics.

```
# 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"])
```

Threshold Selection

Instead of using a default 0.5 cutoff, the model searched for the optimal threshold between 0.05 and 0.95 that minimized the overall misclassification rate on the training data.

The selected threshold was approximately 0.55, meaning that appointments with predicted probability greater than or equal to the threshold are classified as no-shows.

```r
# 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 <- search_thresh(train_x$no_show, train_p)
t
```

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

Evaluation

Predictions were generated on the test dataset to measure generalization.

The overall error rate (misclassification rate) was 0.11, which meets the requirement of being below 0.37.

The ROC AUC was 0.951, indicating strong discrimination between no-shows and shows.

The confusion matrix shows the distribution of true vs. predicted outcomes, confirming that the model balances sensitivity and specificity reasonably well.

```r
# 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)
overall_error <- mean(test_pred != test_x$no_show)
overall_error
```

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

```r
# 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
```

```r
1 - overall_error
```

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

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

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

Discussion

Overall, the model demonstrates that a combination of time-based scheduling variables and provider/patient characteristics can reliably estimate no-show risk. This enables clinics to identify higher-risk appointments and plan overbooking strategies accordingly.

The predicted probabilities from this model were exported for use in the Shiny App, where they are aggregated by hour to compute the No-Show Chance (NSC) metric. This allows users to visualize periods of higher cancellation risk and make data-driven scheduling decisions.

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

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

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), "model_rf_with_threshold.rds")
```

Deviations From Proposed Design

I implemented all major elements from my original design. The only minor change was adjusting the hover text for each hour from "Consider overbooking +1" to simply "Consider overbooking." This change gives schedulers more flexibility to decide how many additional appointments to add based on their own judgment and contextual knowledge. For example, schedulers may know that certain times of year (such as December) tend to have higher cancellation rates and can adjust accordingly.