Patient No-show Prediction Model

```
set.seed(10-27-25)

library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.1     v stringr   1.5.2
## v ggplot2   4.0.0     v tibble    3.3.0
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.1.0
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(randomForest)
```

```
## randomForest 4.7-1.2
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:dplyr':
##
##     combine
##
## The following object is masked from 'package:ggplot2':
##
##     margin
```

Note: Model and Shiny app information is in the bottom sections underneath Model Creation

Create Model

Read in training and testing dataset

```
train_file <- "train_dataset.csv.gz"
test_file <- "test_dataset.csv.gz"

train_raw <- read_csv(train_file, guess_max = 10000)
```

```
## Rows: 36588 Columns: 8
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl  (6): id, provider_id, address, age, specialty, no_show
## dttm (1): appt_time
## date (1): appt_made
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
test_raw <- read_csv(test_file, guess_max = 10000)
```

```
## Rows: 36631 Columns: 8
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl  (6): id, provider_id, address, age, specialty, no_show
## dttm (1): appt_time
## date (1): appt_made
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Feature Engineering

```
# Parse time
train <- train_raw %>%
mutate(
appt_time = ymd_hms(appt_time, tz = "UTC"),
appt_made = as_date(appt_made)
)

test <- test_raw %>%
mutate(
appt_time = ymd_hms(appt_time, tz = "UTC"),
appt_made = as_date(appt_made)
)

# create lead_time_days, appt hour/day, weekend flag

train <- train %>%
mutate(
lead_time_days = as.numeric(difftime(appt_time, appt_made, units = "days")),
appt_hour = hour(appt_time),
appt_wday = wday(appt_time, label = TRUE, week_start = 1), # Monday = 1
is_weekend = if_else(appt_wday %in% c("Sat", "Sun"), 1, 0)
)

test <- test %>%
mutate(
lead_time_days = as.numeric(difftime(appt_time, appt_made, units = "days")),
appt_hour = hour(appt_time),
appt_wday = wday(appt_time, label = TRUE, week_start = 1),
is_weekend = if_else(appt_wday %in% c("Sat", "Sun"), 1, 0)
)

# make no show categorical

train$no_show <- as.factor(train$no_show)

test$no_show <- as.factor(test$no_show)
```

Train Model

```
features <- c("age", "address", "specialty", "provider_id",
"lead_time_days", "appt_hour", "is_weekend")

formula <- as.formula(paste("no_show ~", paste(features, collapse = " + ")))

rf_model <- randomForest(
  formula,
  data = train,
  ntree = 200,
  importance = TRUE
)

print(rf_model)
```

```
##
## Call:
##  randomForest(formula = formula, data = train, ntree = 200, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 200
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 11.54%
## Confusion matrix:
##       0     1 class.error
## 0 21102  1881   0.0818431
## 1  2341 11264   0.1720691
```

Model Results

```
# Class predictions
rf_pred_class <- predict(rf_model, newdata = test, type = "response")
head(rf_pred_class)
```

```
## 1 2 3 4 5 6
## 1 1 0 1 0 0
## Levels: 0 1
```

```
# Class probabilities
rf_pred_prob <- predict(rf_model, newdata = test, type = "prob")
head(rf_pred_prob)
```

```
##       0     1
## 1 0.060 0.940
## 2 0.345 0.655
## 3 0.995 0.005
## 4 0.345 0.655
## 5 0.960 0.040
## 6 0.990 0.010
```

```
# True values
actual <- test$no_show

# Error rate = misclassified / total
```

```r
error_rate <- mean(rf_pred_class != actual)

print(paste("Overall error rate:", round(error_rate, 4)))
```

```
## [1] "Overall error rate: 0.1123"
```

```r
print(paste("Overall error rate < .37:", error_rate < .37))
```

```
## [1] "Overall error rate < .37: TRUE"
```

Save model for Shiny app

```r
saveRDS(rf_model, "fitted_model.rds")
```

Model Information

Model Description

My patient no-show prediction model uses a Random Forest classifier to predict whether a patient will be a "no-show" to their appointment based on patient and appointment characteristics.

Feature engineering was done to create the following variables:

- lead_time_days: the number of days between when the appointment was made and when it occurs.
- appt_hour: the hour integer of the scheduled appointment, extracted from the appointment time.
- is_weekend: a binary flag to indicate if the appointment was on a Saturday or Sunday.

The predictors used in the model were: `age`, `address`, `specialty`, `provider_id`, `lead_time_days`, `appt_hour`, and `is_weekend`. The target variable `no_show` is binary, where 1 indicates a no-show and 0 indicates the patient attended the appointment.

The Random Forest classification model was trained on the training data with 200 trees using Gini impurity to determine the best splits.

Model Performance

The model achieved an Out-of-Bag (OOB) error rate of 11.54%.

Training confusion matrix (OOB):

- Class 0 (show): error rate = 8.2%
- Class 1 (no-show): error rate = 17.2%

When evaluated on the test dataset, the model produced an overall error rate of approximately 0.1123. This meets the requirement that the overall error rate should be less than 0.37. For producing binary predictions, the model uses a 0.5 probability cutoff, meaning appointments with a predicted no-show probability above 50% are classified as no-shows.

Deviations From Proposed Design

All major aspects of the tool remained consistent with the original design document. The minor changes that were made are summarized below:

- Empty time slots were left blank in the Shiny app instead of being filled with a gray bar, as originally proposed. This adjustment was made to reduce visual clutter in the graphs.
- The *Day of the Week* drop-down in the mock-up was replaced with the *Select Week Start Date* input in the actual Shiny app. This change allows clinic staff to view a rolling seven-day period starting from any chosen date, offering greater flexibility and practical use than limiting the view to a specific day of the week.
- Dates were added at the top of each column of bar graphs, in addition to the day-of-week labels. This provides extra clarity since the visualization can begin on any selected date.
- The daily expected utilization percentage is displayed above each column of bars rather than below, as initially proposed. This change simplified implementation of the utlization measure into the app while maintaining readability.
- In the hover tooltip, patient name was replaced with provider ID to align with the available data.