Classification Task 2 - Hepatitis

Tolga Tabanli

2025-10-16

Introduction

In this data set, we are trying to predict the survival class of patients based on a couple of medical predictors.

Since the data has missing values, we indicate if and which symbol is used for missing values with `na = "?"` while reading with `read_csv`.

```
library(tidyverse)
library(tidymodels)
library(naniar)

data <- read_csv("data/hepatitis.data", col_names = c(
  "class", "age", "sex", "steroid", "antivirals", "fatigue", "malaise",
  "anorexia", "liver_big", "liver_firm", "spleen_palpable", "spiders",
  "ascites", "varices", "bilirubin", "alk_phosphate", "sgot", "albumin",
  "protime", "histology"),
  na = "?")
```

Explore the data

This data set has missing values. View the missingness in the data set using `vis_miss` from the package naniar.

```
# TODO
```

Have a look at the distribution of numeric features. Use `pivot_longer` to bring the numeric features under a column named "predictors" with their values under "value". Draw histograms for each. **Hint**: Use `aes(x = value)` for data mapping and `facet_wrap` with `scales = "free"` to show all variables **facetted** in one plot grid design.

```
# TODO
```

Data split

Before modelling and data split, cast the outcome column as factor since it's a *class*:

```
# TODO
```

Then proceed with the data split, setting seed to 2025.

```
# TODO
```

Modelling

We will use random forests as our prediction model and tune each three hyperparameter.

We should account for the missingness in this data set. The procedure for " filling out" the missingness is called **imputation**. There are a couple of approaches to filling these missing cells. A simple one is to use column statistics, such as the mean of the column. On the other hand, we can use another ML algorithm to replace the missingness: For example, KNN identifies the closest data points, the neighbours, of the observation with the missing value, and fills it with the average of its neighbours' values if numerical or with the most frequent one if categorical. We'll compare the imputation approach with another model, where we drop the observations with missing values. We'll take on a transform-then-impute approach.

Your task is:

1. Create the two recipes for no-imputation vs imputation.
2. You may optionally transform the right-skewed variables with log-transform.
3. Specify the model, setting all three hyperparameters of random forest for tune with `tune()`. Remember to set the importance in `set_engine()` to later be able to investigate the variable importance.
4. Create the two workflows.
5. Set seed to 2025, create the parameter grid and CV-fold. Tune both workflows.
6. Plot the tuning results with `autoplot()`.

```
# recipe dropping missing observations
# TODO

# recipe for KNN imputation
# TODO



# model specification
# TODO

# === Workflows ===
# Model with dropping missing observations
# TODO

# Model with imputation
# TODO

# === Tuning ===
# TODO

# autoplot tuning results
# TODO
```

Next, we're going to finalize our workflow:

1. Get the best parameters for each workflow.
2. Finalize the workflows.
3. Fit them using the data split and the metrics of `accuracy` and `roc_auc`.
4. Collect the metrics.

```
# TODO
```

Optionally, investigate the variable importances by using `vip()`.
Hint: `vip()` takes a fitted pure model as an argument → You'll need to extract the parsnip model and get "fit" object for this.

```
library(vip)

# TODO
```