PU5558-PU5567 Assessment 1: Lab Report

Anonymous

2023-04-28

Load necessary packages

```
library(tidyverse)    # for general data science
library(tidymodels)   # for machine learning
library(patchwork)    # for plotting
library(vip)          # for variable importance plots
```

Load and inspect dataset

```
hep_data <- read_csv("hepatitisC_dataset.csv")
```

```
## New names:
## Rows: 615 Columns: 14
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (2): Category, Sex dbl (12): ...1, Age, ALB, ALP, ALT, AST, BIL, CHE, CHOL,
## CREA, GGT, PROT
## 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.
## * `` -> `...1`
```

```
# Use different functions to scan the dataset for variables and their values
glimpse(hep_data)
```

```
## Rows: 615
## Columns: 14
## $ ...1     <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18~
## $ Category <chr> "0=Blood Donor", "0=Blood Donor", "0=Blood Donor", "0=Blood D~
## $ Age      <dbl> 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 33, 33, 33, 33, 3~
## $ Sex      <chr> "m", "m", "m", "m", "m", "m", "m", "m", "m", "m", "m", "m", "~
## $ ALB      <dbl> 38.5, 38.5, 46.9, 43.2, 39.2, 41.6, 46.3, 42.2, 50.9, 42.4, 4~
## $ ALP      <dbl> 52.5, 70.3, 74.7, 52.0, 74.1, 43.3, 41.3, 41.9, 65.5, 86.3, 5~
## $ ALT      <dbl> 7.7, 18.0, 36.2, 30.6, 32.6, 18.5, 17.5, 35.8, 23.2, 20.3, 21~
## $ AST      <dbl> 22.1, 24.7, 52.6, 22.6, 24.8, 19.7, 17.8, 31.1, 21.2, 20.0, 2~
## $ BIL      <dbl> 7.5, 3.9, 6.1, 18.9, 9.6, 12.3, 8.5, 16.1, 6.9, 35.2, 17.2, 5~
## $ CHE      <dbl> 6.93, 11.17, 8.84, 7.33, 9.15, 9.92, 7.01, 5.82, 8.69, 5.46, ~
## $ CHOL     <dbl> 3.23, 4.80, 5.20, 4.74, 4.32, 6.05, 4.79, 4.60, 4.10, 4.45, 3~
## $ CREA     <dbl> 106, 74, 86, 80, 76, 111, 70, 109, 83, 81, 78, 79, 78, 65, 63~
## $ GGT      <dbl> 12.1, 15.6, 33.2, 33.8, 29.9, 91.0, 16.9, 21.5, 13.7, 15.9, 2~
## $ PROT     <dbl> 69.0, 76.5, 79.3, 75.7, 68.7, 74.0, 74.5, 67.1, 71.3, 69.9, 7~
```

```
summary(hep_data)
```

```
##       ...1         Category              Age             Sex
##  Min.   :  1.0   Length:615         Min.   :19.00   Length:615
##  1st Qu.:154.5   Class :character   1st Qu.:39.00   Class :character
##  Median :308.0   Mode  :character   Median :47.00   Mode  :character
##  Mean   :308.0                      Mean   :47.41
##  3rd Qu.:461.5                      3rd Qu.:54.00
##  Max.   :615.0                      Max.   :77.00
##
##       ALB             ALP              ALT              AST
##  Min.   :14.90   Min.   : 11.30   Min.   :  0.90   Min.   : 10.60
##  1st Qu.:38.80   1st Qu.: 52.50   1st Qu.: 16.40   1st Qu.: 21.60
##  Median :41.95   Median : 66.20   Median : 23.00   Median : 25.90
##  Mean   :41.62   Mean   : 68.28   Mean   : 28.45   Mean   : 34.79
##  3rd Qu.:45.20   3rd Qu.: 80.10   3rd Qu.: 33.08   3rd Qu.: 32.90
##  Max.   :82.20   Max.   :416.60   Max.   :325.30   Max.   :324.00
##  NA's   :1       NA's   :18       NA's   :1
##       BIL             CHE              CHOL            CREA
##  Min.   :  0.8   Min.   : 1.420   Min.   :1.430   Min.   :   8.00
##  1st Qu.:  5.3   1st Qu.: 6.935   1st Qu.:4.610   1st Qu.:  67.00
##  Median :  7.3   Median : 8.260   Median :5.300   Median :  77.00
##  Mean   : 11.4   Mean   : 8.197   Mean   :5.368   Mean   :  81.29
##  3rd Qu.: 11.2   3rd Qu.: 9.590   3rd Qu.:6.060   3rd Qu.:  88.00
##  Max.   :254.0   Max.   :16.410   Max.   :9.670   Max.   :1079.10
##                                   NA's   :10
##       GGT              PROT
##  Min.   :  4.50   Min.   :44.80
##  1st Qu.: 15.70   1st Qu.:69.30
##  Median : 23.30   Median :72.20
##  Mean   : 39.53   Mean   :72.04
##  3rd Qu.: 40.20   3rd Qu.:75.40
##  Max.   :650.90   Max.   :90.00
##                   NA's   :1
```

```r
head(hep_data)
```

```
## # A tibble: 6 x 14
##    ...1 Category       Age Sex     ALB   ALP   ALT   AST   BIL   CHE  CHOL  CREA
##   <dbl> <chr>        <dbl> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1     1 0=Blood Don~    32 m      38.5  52.5   7.7  22.1   7.5  6.93  3.23   106
## 2     2 0=Blood Don~    32 m      38.5  70.3  18    24.7   3.9 11.2   4.8     74
## 3     3 0=Blood Don~    32 m      46.9  74.7  36.2  52.6   6.1  8.84  5.2     86
## 4     4 0=Blood Don~    32 m      43.2  52    30.6  22.6  18.9  7.33  4.74    80
## 5     5 0=Blood Don~    32 m      39.2  74.1  32.6  24.8   9.6  9.15  4.32    76
## 6     6 0=Blood Don~    32 m      41.6  43.3  18.5  19.7  12.3  9.92  6.05   111
## # ... with 2 more variables: GGT <dbl>, PROT <dbl>
```

Suitable machine learning (ML) algorithm for three questions:

1. Calculate the percentage of men in the dataset who have cirrhosis

No ML algorithm needed to answer this question: Simply filter the data for men (Sex=="m"), count the number of rows (N), further filter for Category=="3=Cirrhosis", count the number of rows (n), and divide both numbers (n/N). Multiplied by 100, this will result in the percentage (n/n * 100) of men in the dataset (N) who have cirrhosis (n).

2. Separate the patients in the dataset into groups according to the diagnostic tests

This is an exploratory question, since the groups are not defined further, so an unsupervised ML algorithm would be the most suitable. One could for example use k-means, a clustering algorithm for numerical data like our diagnostic test values. One would start by removing all variables but the laboratory values and is left with numerical variables only. One could use the Elbow Method to determine the number of groups (clusters) in which the patients should be partitioned before finally applying k-means.

3. Predict whether a patient is a blood donor, or has Hepatitis C. Here we treat the three different stages ('just' Hepatitis C, Fibrosis, Cirrhosis) as a single category.

This would be best answered using supervised ML. With given groups of patients that are contained in the dataset, one can train a classification algorithm to predict whether a new patients is a blood donor or has Hepatitis C based on other variables such as age, sex or diagnostic test results. Random Forests, Logistic Regression and SVMs are examples for such classifiers. This question shall be answered in the following analysis.

Model building to answer chosen question (3.)

0. Data Prepping

Since we want to treat the three stages as a single category, we alter the dataset respectively. There is still more than one category for the blood donor group, so we assume suspected Blood Donors are in fact Blood Donors and adjust their category.

```r
# Remove potential rows with missing values
hep_data <- hep_data %>%
  na.omit()

# Adjust "1" category
hep_data <- hep_data %>%
  mutate(Category = factor(gsub(".*sis", "1=Hepatitis", Category)))

# Adjust "0" category
hep_data <- hep_data %>%
  mutate(Category = factor(gsub("0.*", "0=Donor", Category)))
```

1. Data splitting

In order to be able to properly assess the model's performance later, we set 20% of the dataset aside as a test set, while making sure Category has similar distributions in the training set and test set. We also assess the distribution of categories to predict in order to estimate class imbalance.

```r
hep_split <- hep_data %>%
    initial_split(prop = 0.8, strata = Category)

hep_train <- training(hep_split)
hep_test <- testing(hep_split)

# Assess class (im)balance
n_class0 <- hep_data %>%
  filter(Category == "0=Donor") %>%
  nrow()

print("Fraction of observations belonging to donors:")
```

```
## [1] "Fraction of observations belonging to donors:"
```

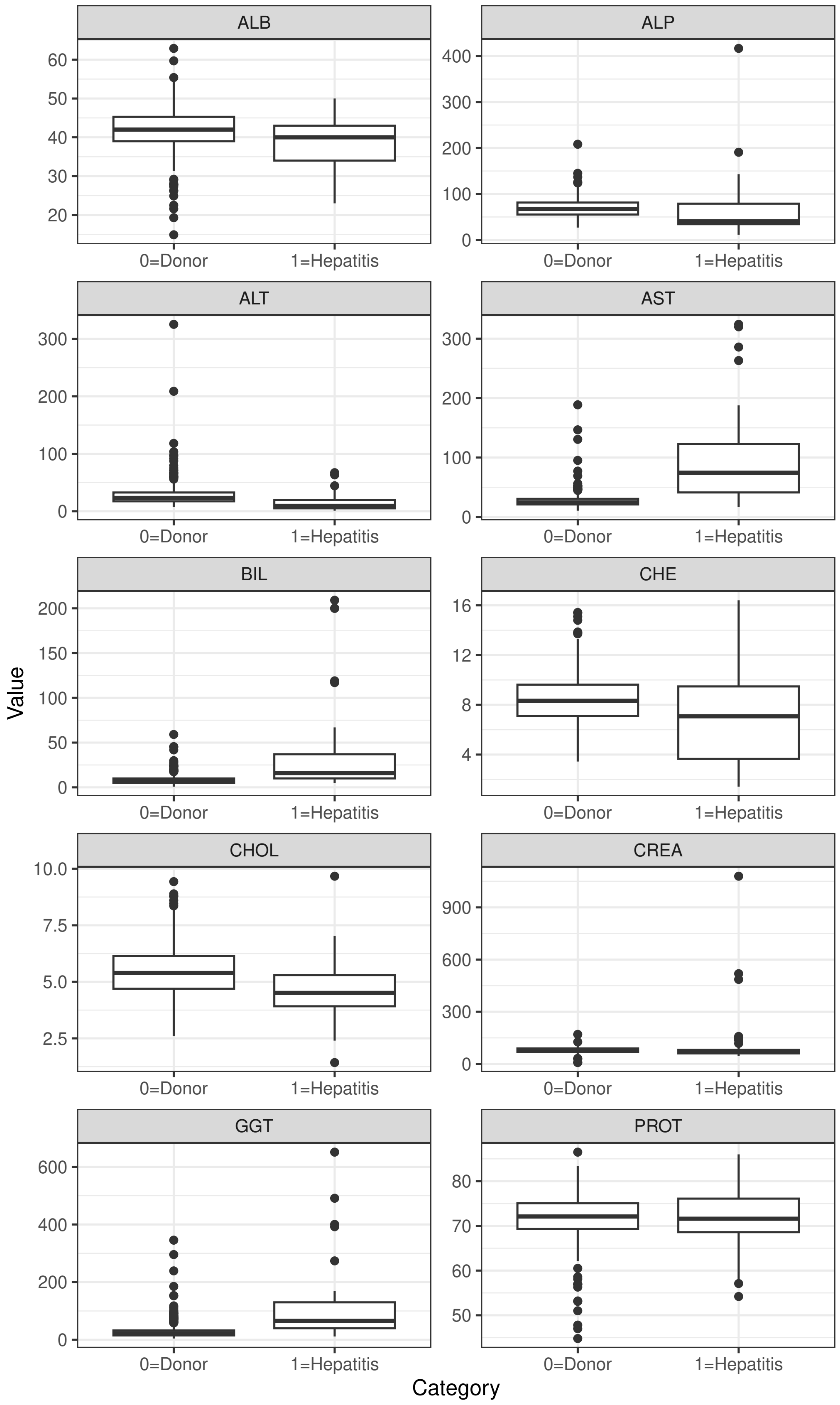
```r
n_class0/nrow(hep_data)
```

```
## [1] 0.9049236
```

2. Selection and preprocessing of predictors

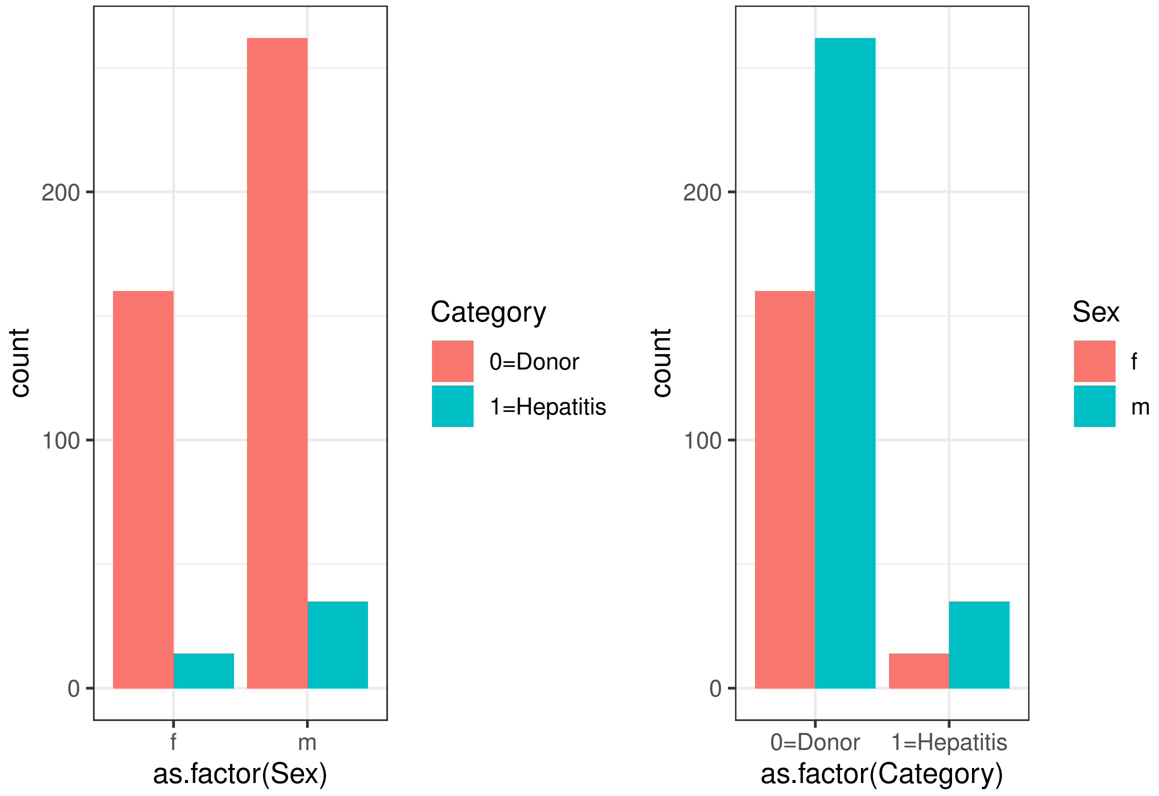
Now we can explore the relationships between our potential predictors and the patient's Category.

```r
# Boxplots for the continuous variables (e.g. Age and diagnostic test results)
plotting_data <- hep_train %>%
  pivot_longer(c("ALB","ALP","ALT","AST","BIL","CHE","CHOL","CREA","GGT","PROT"),
               names_to = "Test",
               values_to = "Value")

plotting_data %>%
  ggplot(aes(x = Category, y = Value)) +
  geom_boxplot() +
  theme_bw() +
  facet_wrap(~Test, ncol = 2, scales = "free")
```



ALB
ALP
ALT
AST
BIL
CHE
CHOL
CREA
GGT
PROT
0=Donor
1=Hepatitis
Value
Category

```r
# Barplots for categorical predictor Sex

bars1 <- hep_train %>%
  ggplot(aes(x = as.factor(Sex), fill = Category)) +
  geom_bar(position="dodge") +
  theme_bw()

bars2 <- hep_train %>%
  ggplot(aes(x = as.factor(Category), fill = Sex)) +
  geom_bar(position="dodge") +
  theme_bw()

bars1 + bars2 + plot_layout()
```



These plots tell us that GGT, BIL, ALT and AST might be helpful discriminatory predictors, while other variables such as Age or Sex did not seem to be differently distributed in the two outcome groups. In order to select the final set of predictors, we use the results from our exploratory analysis as well as expert opinions. Here, findings reported in given publication include the mentioning of ALB, ALT and CHE as expected to have high discriminatory power. The decision trees they calculated however also incorporated BIL and GGT. It is worth noting that these variables that discriminate between the different stages of Hepatitis do not necessarily contribute the most to our binary classification of separating cases from controls. This is why we relied more on our visual analysis than on the reported findings. Thus, we selected GGT, BIL, ALT and AST as predictors for patient category.

Next, we specify our recipe and include relevant pre-processing steps. For logistic regression, it is not required to normalise or transfrom predictors. Since we do not have qualitative or missing predictors, it is not required to encode them numerically or estimate them via imputation. Therefore, the only two steps we apply is removing potential columns with a single unique value as well as mitigating potential correlation between predictors.

```
rec <- hep_train %>%
  recipe(Category ~ GGT + BIL + ALT + AST) %>%
  step_zv(all_predictors()) %>%
  step_corr(all_predictors())
```

3. Model specification and training

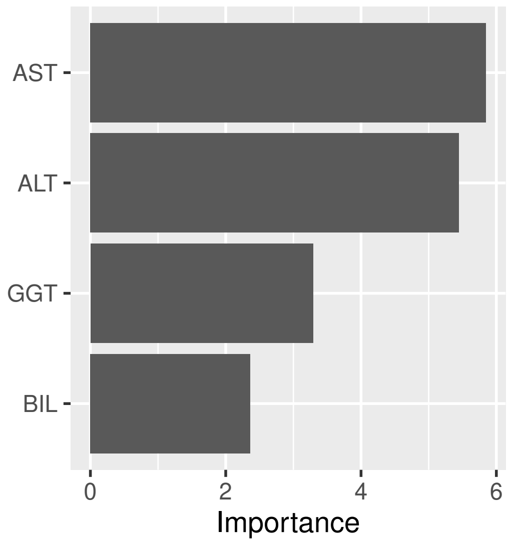
Logistic regression does classification by definition. As engine, we will use the default glm, and we set the model type to logistic_reg to specify the workflow. In order to be able to use a validation set to manually evaluate and tune hyperparameter settings while still using all of the training set for actual training, we employ cross-validation as a resampling method. Then we can actually apply the recipe and train the model. We also extract a ranking of variables by importance from the fitted model to get an idea of which laboratory tests help the most in the prediction process.

```
# Combine recipe and model in a workflow
wflow <- workflow() %>%
  add_recipe(rec) %>%
  add_model(logistic_reg(mixture = 0.5))

# Specify a 5-fold cross-validation
cv <- vfold_cv(hep_train, v = 5, strata = Category)

# Define a set of performance metrics to be used for evaluation
perf_metrics <- metric_set(accuracy, roc_auc, recall, spec, precision, mcc)

# Fit the model
wflow_fit <- wflow %>%
  fit_resamples(
    resamples = cv,
    control = control_resamples(save_pred = TRUE),
    metrics = perf_metrics
  )
```

```
## ! Fold1: preprocessor 1/1, model 1/1: glm.fit: fitted probabilities numerically 0 or 1 occurred

## ! Fold2: preprocessor 1/1, model 1/1: glm.fit: fitted probabilities numerically 0 or 1 occurred

## ! Fold3: preprocessor 1/1, model 1/1: glm.fit: fitted probabilities numerically 0 or 1 occurred

## ! Fold4: preprocessor 1/1, model 1/1: glm.fit: fitted probabilities numerically 0 or 1 occurred

## ! Fold5: preprocessor 1/1, model 1/1: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
# Variable importance: fit model once more due to fit_resamples particularities
wflow %>%
  fit(hep_train) %>%
  extract_fit_parsnip() %>%
  vip()
```

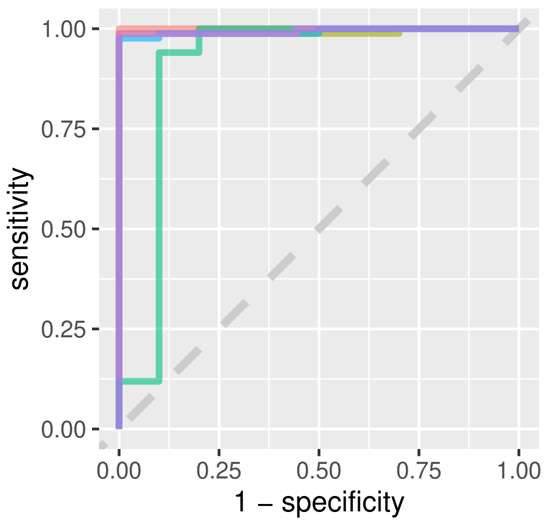
```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```



We receive warnings for fitted probability numerically indistinguishable from 0 or 1. These come from very small probabilities assigned to the prediction of the minority class (Hepatitis) for majority class observations (Donor). The reason for this is the class imbalance within the data. Since there are more controls in the sample than cases, the model is unlikely to falsely assign a control patient to the minority class.

To tune our model before finally evaluating it on the held-out test set, we check its performance on the validation folds generated through cross-validation. We can use these validation scores to estimate the model's performance on new data without making biased decisions by tuning it based on test set scores.

```r
# Plot the ROC curve to estimate performance for all CV folds
wflow_fit %>%
  collect_predictions() %>%
  group_by(id) %>%
  roc_curve(Category, '.pred_0=Donor') %>%
  ggplot(aes(1 - specificity, sensitivity, color = id)) +
  geom_abline(lty = 2, color = "gray80", linewidth = 1.5) +
  geom_path(show.legend = FALSE, alpha = 0.6, linewidth = 1.2) +
  coord_equal()
```



```
# Collect performance metrics
collect_metrics(wflow_fit)
```

```
## # A tibble: 6 x 6
##   .metric   .estimator  mean     n std_err .config
##   <chr>     <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy  binary     0.975     5 0.00925 Preprocessor1_Model1
## 2 mcc       binary     0.857     5 0.0536  Preprocessor1_Model1
## 3 precision binary     0.981     5 0.00862 Preprocessor1_Model1
## 4 recall    binary     0.991     5 0.00237 Preprocessor1_Model1
## 5 roc_auc   binary     0.977     5 0.0178  Preprocessor1_Model1
## 6 spec      binary     0.84      5 0.0748  Preprocessor1_Model1
```

```
conf_mat_resampled(wflow_fit, tidy = FALSE)
```

```
##              0=Donor 1=Hepatitis
## 0=Donor         83.6         1.6
## 1=Hepatitis      0.8         8.2
```

After manually tuning the model parameters mixture and penalty, we may settle on mixture = 0.5 and the default value for penalty. These settings gave the numerically best results for the performance metrics. Due to the class imbalance present in the dataset (~90% of all observations were 0=Donor patients), it is important to look at multiple metrics, such as the Matthew's Correlation Coefficient (MCC), which is more resistant towards class imbalance. However, it was not possible to properly fine-tune these parameters, since most differences in metrics for different parameter values were within standard error range, not allowing us to definitively favor one setting over the other.

4. Final model evaluation

Lastly, we can evaluate the final fit of the model on the held-out test set to provide an unbiased assessment of our model's performance on real-life data.

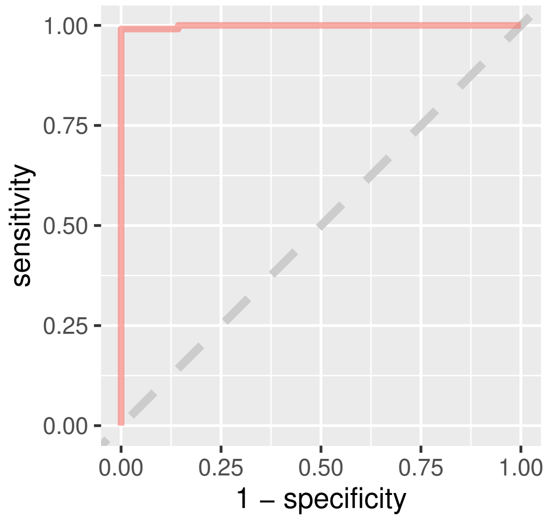
```
# Fit the final model
final_fit <- wflow %>%
  last_fit(hep_split, metrics = perf_metrics)
```

```
## ! train/test split: preprocessor 1/1, model 1/1: glm.fit: fitted probabilities numerically 0 or 1 oc
```

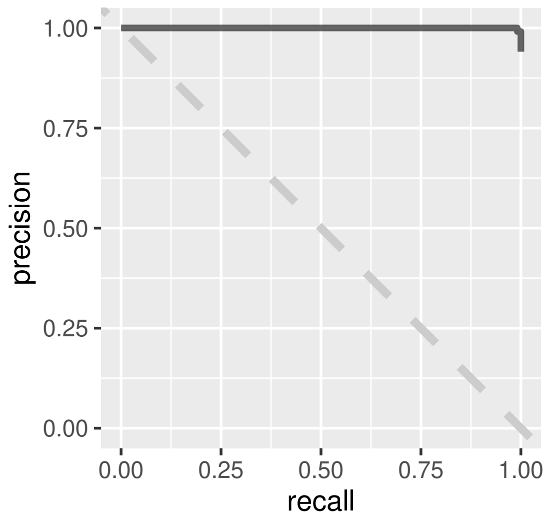
```
# Evaluate on testing data
collect_metrics(final_fit)
```

```
## # A tibble: 6 x 4
##   .metric   .estimator .estimate .config
##   <chr>     <chr>          <dbl> <chr>
## 1 accuracy  binary         0.983 Preprocessor1_Model1
## 2 recall    binary         0.991 Preprocessor1_Model1
## 3 spec      binary         0.857 Preprocessor1_Model1
## 4 precision binary         0.991 Preprocessor1_Model1
## 5 mcc       binary         0.848 Preprocessor1_Model1
## 6 roc_auc   binary         0.999 Preprocessor1_Model1
```

```
# Plot the ROC
final_fit %>%
  collect_predictions() %>%
  group_by(id) %>%
  roc_curve(Category, '.pred_0=Donor') %>%
  ggplot(aes(1 - specificity, sensitivity, color = id)) +
  geom_abline(lty = 2, color = "gray80", linewidth = 1.5) +
  geom_path(show.legend = FALSE, alpha = 0.6, linewidth = 1.2) +
  coord_equal()
```



```
# Plot the precision-recall curve
final_fit %>%
  collect_predictions() %>%
  group_by(id) %>%
  pr_curve(Category, '.pred_0=Donor') %>%
  ggplot(aes(x = recall, y = precision)) +
  geom_abline(lty = 2, slope = -1, intercept = 1, color = "gray80", linewidth = 1.5) +
  geom_path(show.legend = FALSE, alpha = 0.6, linewidth = 1.2) +
  coord_equal(ylim = c(0, 1))
```



In terms of performance interpretation, our model seems to be doing a very good job at predicting which patients have Hepatitis, e.g. performing at 99% accuracy. Both the ROC and the precision-recall curve signify solid performance. We included the latter because these curve are more resistant towards potential class imbalance (mentioned below). These scores can be better than those for the validation set, which would most likely be due chance based on the even smaller size of the test set generating scores that are numerically better.

Limitations

However, there are several limitations to be mentioned regarding this process.

First, concerning performance, our model has a notably higher sensitivity (or recall) than specificity.

That means that it confidently detects Hepatitis cases, but that also includes false positives. Thus, this model would be best suited as a screening device used prior to an assessment by a professional, which should either confirm a case or uncover a false positive. Of course, before being used for actual Hepatitis screening, this model would have to be judged by experts in the field, undergo trial stages and be assessed for potential biases.

Second, logistic regression involves a set of assumptions in terms of the statistical background.

Most importantly, logistic regression assumes a linear relationship between the predictors and the logit of the outcome variable, which means that non-linear problems cannot be solved. By employing logistic regression here, we made the assumption that the relationships of the diagnostic tests and the logit of the patient category is indeed linear. For this model to be employed in practice, proof of the relationships would have to be made available.

Third, this process did not exhaust all ways of optimising model performance.

Further endeavors should include testing different engines for logistic regression as well as tuning for other hyperparameters, linked to employing resampling strategies to combat class imbalance. Techniques such as oversampling could also contribute to enhancing the training set size which, in the world of machine learning, is very limited.

Different approaches to classification such as Random Forests or Support Vector Machines could also deliver improved results. However, these have their limitations as well: A model based on one of these ML algorithms is much more like a “black box” than logistic regression, which allows to independently retrace and interpret the influence of different variables. Since these types of classifiers are also rather prone to overfitting, it would be wise to incorporate resampling techniques. It would also be useful to check the development of the difference in training and validation scores with training iterations. (Should the training performance continue to improve, but performance on the validation set(s) decreases, this is most likely due to overfitting.) Other attempts to avoid overfitting include restricting the number of features or variables used for prediction. This can be approached through variable importance analysis or consulting experts in the field who can shed more light on which diagnostic tests have the most discriminatory power in their experience.