{r setup, include=FALSE} knitr::opts_chunk$set(echo = TRUE) knitr :: opts_chunkset(cache=TRUE, autodep=TRUE) knitr::opts_chunk$set(fig.align="center", fig.pos="tbh")

Module 4

> This module introduced cross-validation and bootstrapping as methods to estimate model performance. The homework assignment will give you the opportunity to apply these methods to two datasets. In assignment (1), you will use cross-validation to compare logistic regression, LDA, and QDA classification models. Assignment (2) uses bootstrap to estimate confidence intervals for the mean absolute error and root mean squared error of a linear regression model.
>
> Use *Tidyverse* and *Tidymodels* packages for the assignments.
>
> You can download the R Markdown file (https://gedeck.github.io/DS-6030/homework/Module-4.Rmd (https://gedeck.github.io/DS-6030/homework/Module-4.Rmd)) and use it as a basis for your solution.
>
> As you will find out, the knitting times for this assignment will be longer than in previous homework. To speed up the knitting process, use caching and parallel processing. You can find more information about caching (https://gedeck.github.io/DS-6030/book/technical-details.html#appendix-caching) and about parallel processing (https://gedeck.github.io/DS-6030/book/technical-details.html#appendix-parallel) in the course material.

1. Diabetes dataset

The Diabetes Health Indicators Dataset contains healthcare statistics and lifestyle survey information about people in general along with their diagnosis of diabetes. The 35 features consist of some demographics, lab test results, and answers to survey questions for each patient. The target variable for classification is whether a patient has diabetes, is pre-diabetic, or healthy. For this study, the target variable was converted to a binary variable with 1 for diabetes or pre-diabetic and 0 for healthy. Information about the dataset can be found here (https://gedeck.github.io/DS-6030/datasets/diabetes.html).

For this exercise use caching and parallel processing to speed up your computations.

```
library(tidymodels)
library(tidyverse)
library(discrim)
library(patchwork)
library(probably)
library(future)
library(doParallel)
plan(multisession, workers = 2)
registerDoParallel(cores = 2)
```

A. Data loading and preprocessing

(1.1) Load the diabetes dataset from https://gedeck.github.io/DS-6030/datasets/diabetes/diabetes_binary_5050split_health_indicators_BRFSS2015.csv.gz (https://gedeck.github.io/DS-6030/datasets/diabetes/diabetes_binary_5050split_health_indicators_BRFSS2015.csv.gz). Convert the `Diabetes_binary` to a factor with labels *Healthy* and *Diabetes*. Convert all other variables that only contain values of 0 and 1 to factors. (1 point - coding)

```
# Load the data

url <- "https://gedeck.github.io/DS-6030/datasets/diabetes/diabetes_binary_5050split_health_indicators_BRFSS2015.csv.gz"

data <- read_csv(url)
```

```
## Rows: 70692 Columns: 22
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## dbl (22): Diabetes_binary, HighBP, HighChol, CholCheck, BMI, Smoker, Stroke,...
##
## 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.
```

```
# Convert target variable

data <- data %>%
  mutate(Diabetes_binary = factor(Diabetes_binary, levels = c(0, 1), labels = c("Healthy", "Diabetes")))

# Convert binary variables to factors

binary_vars <- data %>%
  select(-Diabetes_binary) %>%
  select(where(~ all(.x %in% c(0, 1)))) %>%
  names()

data <- data %>%
  mutate(across(all_of(binary_vars), as.factor))
```

(1.2) Split the data into a training and test set using a 50/50 split. Use the `set.seed()` function to ensure reproducibility. (1 point - coding)

```
set.seed(6030)

split <- initial_split(data,prop = 0.5, strata = Diabetes_binary)
train_data <- training(split)
test_data <- testing(split)
```

B. Model training

(1.3) Build a logistic regression model to predict `Diabetes_binary` using all other variables as predictors. Use the training set to fit the model using 10-fold cross-validation. Report the cross-validation accuracy and ROC-AUC of the model. (see DS-6030: Model validation using cross-validation (https://gedeck.github.io/DS-6030/book/model-validation.html#model-validation-using-cross-validation)) (2 points - coding)

```
set.seed(6030)
logistic_model <- logistic_reg() %>% set_engine("glm")

logistic_wf <- workflow() %>%
  add_model(logistic_model) %>%
  add_formula(Diabetes_binary ~ .)

logistic_res <- fit_resamples(
  logistic_wf,
  vfold_cv(train_data, v = 10, strata = Diabetes_binary),
  metrics = metric_set(accuracy, roc_auc),
  control = control_resamples(save_pred = TRUE)
)

collect_metrics(logistic_res)
```

```
## # A tibble: 2 × 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy binary     0.749    10 0.00201 Preprocessor1_Model1
## 2 roc_auc  binary     0.825    10 0.00239 Preprocessor1_Model1
```

(1.4) Use the approach from **(1.3)** to build LDA and QDA models. Report the cross-validation accuracy and ROC-AUC of each model. (4 points - coding)

```
# LDA

lda_model <- discrim_linear() %>% set_engine("MASS")

lda_wf <- workflow() %>%
  add_model(lda_model) %>%
  add_formula(Diabetes_binary ~ .)

lda_res <- fit_resamples(
  lda_wf,
  vfold_cv(train_data, v = 10, strata = Diabetes_binary),
  metrics = metric_set(accuracy, roc_auc),
  control = control_resamples(save_pred = TRUE)
)

collect_metrics(lda_res)
```

```
## # A tibble: 2 × 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy binary     0.748    10 0.00282 Preprocessor1_Model1
## 2 roc_auc  binary     0.824    10 0.00252 Preprocessor1_Model1
```

```
# QDA

set.seed(6030)

qda_model <- discrim_linear() %>% set_engine("MASS")

qda_wf <- workflow() %>%
  add_model(qda_model) %>%
  add_formula(Diabetes_binary ~ .)

qda_res <- fit_resamples(
  lda_wf,
  vfold_cv(train_data, v = 10, strata = Diabetes_binary),
  metrics = metric_set(accuracy, roc_auc),
  control = control_resamples(save_pred = TRUE)
)

collect_metrics(qda_res)
```

```
## # A tibble: 2 × 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy binary     0.748    10 0.00212 Preprocessor1_Model1
## 2 roc_auc  binary     0.824    10 0.00239 Preprocessor1_Model1
```

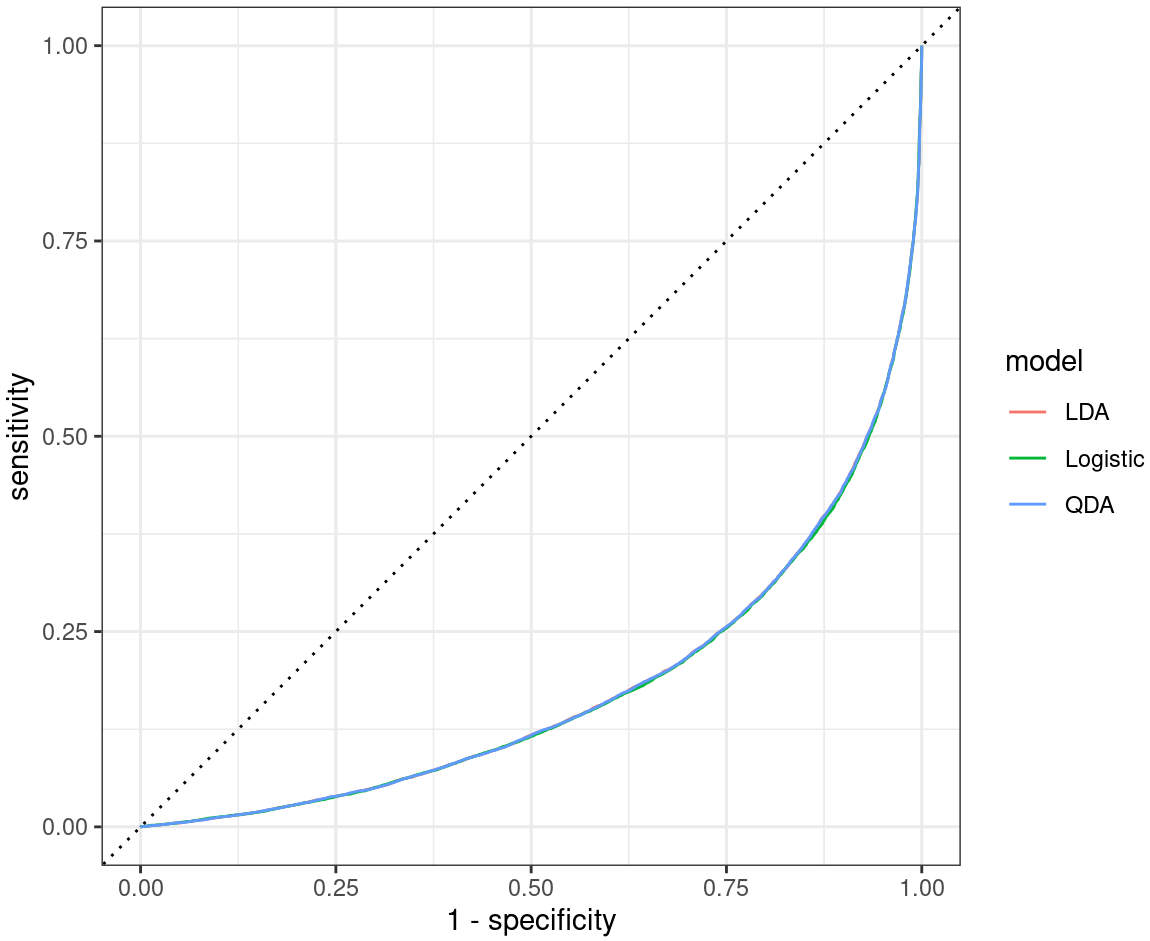
C. Cross-validation and test set performance

(1.5) Create a plot that compares the ROC curves of the three models from **(1.3)** and **(1.4)**. The ROC curve should be based on the predictions from cross-validation. (1 point - coding/discussion)

How do the models compare? Which model would you choose for prediction?

```
logistic_preds <- logistic_res %>% collect_predictions()
lda_preds <- lda_res %>% collect_predictions()
qda_preds <- qda_res %>% collect_predictions()

bind_rows(
  logistic_preds %>% mutate(model = "Logistic"),
  lda_preds %>% mutate(model = "LDA"),
  qda_preds %>% mutate(model = "QDA")
) %>%
  group_by(model) %>%
  roc_curve(Diabetes_binary, .pred_Diabetes) %>%
  autoplot()
```



Based on the visual, all three models seem to have a flexible fit. However, I think QDA would be the best model to choose for prediction.

(1.6) After fitting the three models using the full training set, estimate the performance metrics on the test set. Report the accuracy and ROC-AUC of each model. How do the models compare? Do you see a difference to the cross-validation results? (1 point - coding/discussion)

```
fit_log <- fit(logistic_wf, train_data)
fit_lda <- fit(lda_wf, train_data)
fit_qda <- fit(qda_wf, train_data)

predict_and_eval <- function(fit, name) {
  preds <- predict(fit, test_data, type = "prob") %>%
    bind_cols(predict(fit, test_data), test_data %>% select(Diabetes_binary))

  metrics <- metrics(preds, truth = Diabetes_binary, estimate = .pred_class)
  auc <- roc_auc(preds, truth = Diabetes_binary, .pred_Diabetes)
  bind_rows(metrics, auc) %>% mutate(model = name)
}

bind_rows(
  predict_and_eval(fit_log, "Logistic"),
  predict_and_eval(fit_lda, "LDA"),
  predict_and_eval(fit_qda, "QDA")
)
```

```
## # A tibble: 9 × 4
##   .metric  .estimator .estimate model
##   <chr>    <chr>          <dbl> <chr>
## 1 accuracy binary         0.748 Logistic
## 2 kap      binary         0.495 Logistic
## 3 roc_auc  binary         0.175 Logistic
## 4 accuracy binary         0.746 LDA
## 5 kap      binary         0.492 LDA
## 6 roc_auc  binary         0.176 LDA
## 7 accuracy binary         0.746 QDA
## 8 kap      binary         0.492 QDA
## 9 roc_auc  binary         0.176 QDA
```

Based on the dataframe, it seems the results are consistent. This suggests that the models generalize well.

2. Estimate model performance using bootstrap

(2.1) Use the `mtcars` dataset from DS-6030: The mtcars dataset (https://gedeck.github.io/DS-6030/book/regression-models.html#the-mtcars-dataset) to estimate the mean absolute error (MAE) and root mean squared error (RMSE) of the linear regression model for the prediction of `mpg` using bootstrap. Use 1000 bootstrap samples. Report the mean and standard deviation of the two metrics. (see DS-6030: Model validation using bootstrapping (https://gedeck.github.io/DS-6030/book/model-validation.html#model-validation-using-bootstrapping)) (3 points - coding)

```
set.seed(6030)
boot_res <- bootstraps(mtcars, times = 1000)

lm_model <- linear_reg() %>% set_engine("lm")
lm_wf <- workflow() %>%
  add_model(lm_model) %>%
  add_formula(mpg ~ .)

boot_metrics <- map_df(boot_res$splits, function(split) {
  fit <- fit(lm_wf, analysis(split))
  preds <- predict(fit, assessment(split)) %>%
    bind_cols(assessment(split))
  data.frame(
    MAE = mae(preds, truth = mpg, estimate = .pred)$.estimate,
    RMSE = rmse(preds, truth = mpg, estimate = .pred)$.estimate
  )
})
```

```
## Warning in predict.lm(object = object$fit, newdata = new_data, type =
## "response", : prediction from rank-deficient fit; consider predict(.,
## rankdeficient="NA")
```

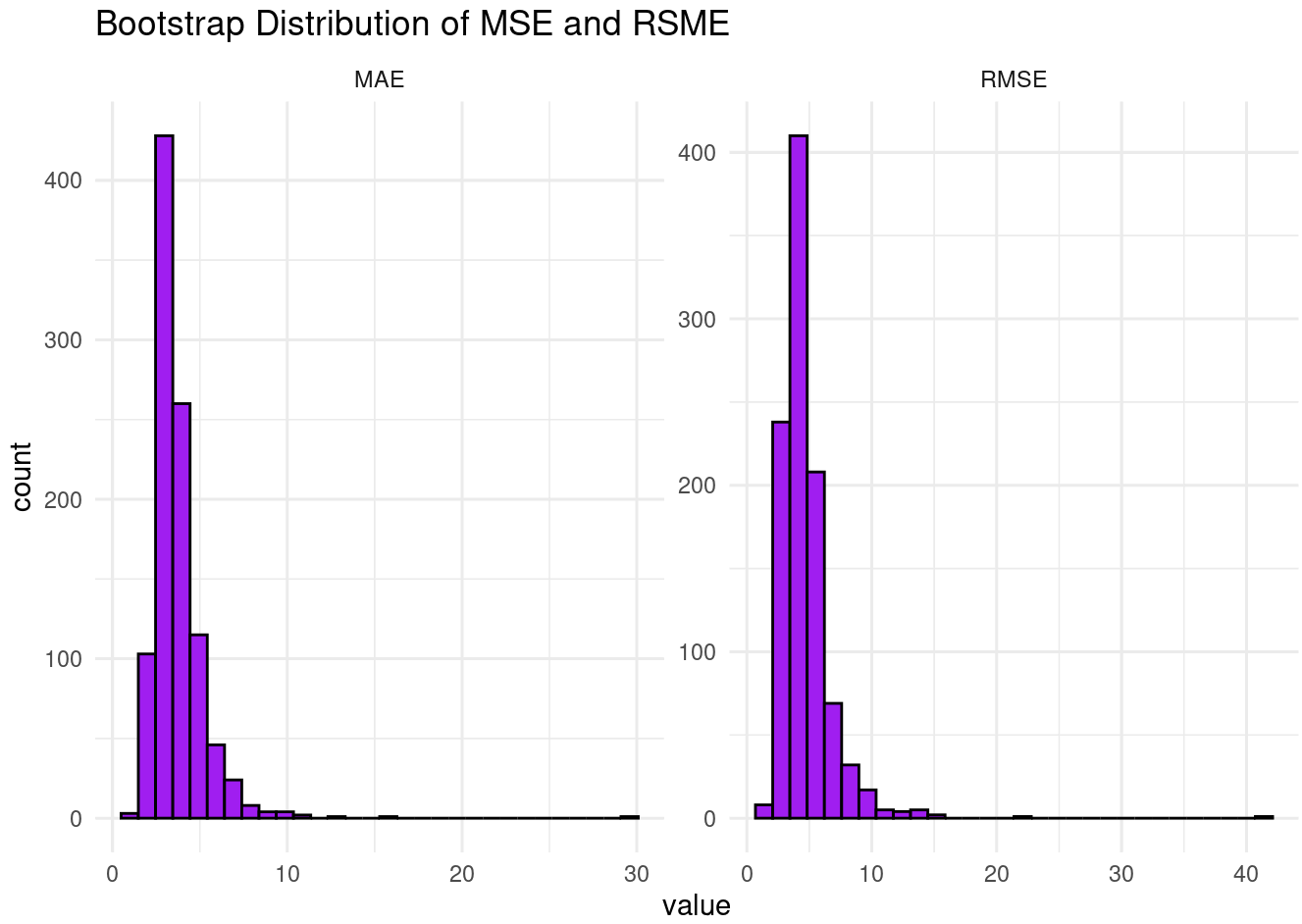
```
summary_stats <- boot_metrics %>%
  summarise(across(everything(), list(mean = mean, sd = sd)))

summary_stats
```

```
##   MAE_mean   MAE_sd RMSE_mean  RMSE_sd
## 1 3.715499 1.608655  4.691715 2.236392
```

(2.2) Create a plot of the distribution of the performance metrics. Comment on the shape of the distribution. (1 point - coding/discussion)

```
boot_metrics %>%
  pivot_longer(everything()) %>%
  ggplot(aes(x = value)) +
  geom_histogram(bins = 30, fill = "purple", color = "black") +
  facet_wrap (~name, scales = "free") +
  theme_minimal() +
  labs(title = "Bootstrap Distribution of MSE and RSME")
```



Based on the histograms, it seems that the shape of both distributions is rightly skewed.

(2.3) Use the performance metrics calculated for the bootstrap samples to estimate the 95% confidence interval for the mean absolute error (MAE) and root mean squared error (RMSE). Report the confidence intervals. (2 points - coding/discussion)

```
boot_metrics %>%
  summarise(
    mae_lower = quantile(MAE,0.025),
    mae_upper = quantile(MAE, 0.975),
    rmse_lower = quantile(RMSE, 0.025),
    rmse_upper = quantile(RMSE, 0.975)
  )
```

```
##   mae_lower mae_upper rmse_lower rmse_upper
## 1  1.975636  7.267294   2.476573   9.739966
```

We are 95% confident that the MAE is between 1.9756 and 7.2672, and we are 95% confident that the RMSE is between 2.477 and 9.740.