# MATHS 7107 Data Taming Practical

## Part 1

First we will start by looking at how to measure a model using yardstick. We will fit a regression model, and also a classification model to the penguins dataset and then have a look at assessing them.

## Load the data and required packages

```
pacman::p_load(tidyverse, tidymodels)
```

```
data("penguins", package = "palmerpenguins")
```

## Create the models

```
penguin_M1 <-
  workflow() %>%
  add_formula(flipper_length_mm ~ body_mass_g) %>%
  add_model(
    linear_reg() %>%
      set_engine("lm")
    ) %>%
  fit(penguins)
penguin_M1
```

```
## == Workflow [trained] ==========================================================
## Preprocessor: Formula
## Model: linear_reg()
##
## -- Preprocessor ----------------------------------------------------------------
## flipper_length_mm ~ body_mass_g
##
## -- Model -----------------------------------------------------------------------
##
## Call:
## stats::lm(formula = ..y ~ ., data = data)
##
## Coefficients:
## (Intercept)  body_mass_g
##   136.72956      0.01528
```

```
penguin_M2 <-
  workflow() %>%
  add_formula(sex ~ body_mass_g) %>%
  add_model(
    logistic_reg() %>% set_engine("glm")
    ) %>%
```

```r
  fit(penguins)
penguin_M2
```

```
## == Workflow [trained] ==========================================================
## Preprocessor: Formula
## Model: logistic_reg()
##
## -- Preprocessor ----------------------------------------------------------------
## sex ~ body_mass_g
##
## -- Model -----------------------------------------------------------------------
##
## Call:  stats::glm(formula = ..y ~ ., family = stats::binomial, data = data)
##
## Coefficients:
## (Intercept)  body_mass_g
##    -5.16254      0.00124
##
## Degrees of Freedom: 332 Total (i.e. Null);  331 Residual
##   (11 observations deleted due to missingness)
## Null Deviance:       461.6
## Residual Deviance: 396.6     AIC: 400.6
```

**Question:** For model 1, what are the response variable and the predictors.

**Question:** For model 2, what are the response variable and the predictors.

## Getting prediction

For yardstick, we will need predicted values, we obtain that using the `predict()` function. Here I will add a variety of predictions to the original dataset.

```r
penguins_pred <-
  penguins %>%
  bind_cols(
    predict(penguin_M1, penguins),
    predict(penguin_M2, penguins),
    predict(penguin_M2, penguins,
            type = "prob"),
  ) %>%
  select(sex, flipper_length_mm,starts_with(".pred"))
penguins_pred
```

```
## # A tibble: 344 x 6
##    sex    flipper_length_mm .pred .pred_class .pred_female .pred_male
##    <fct>              <int> <dbl> <fct>              <dbl>      <dbl>
##  1 male                 181  194. female             0.626      0.374
##  2 female               186  195. female             0.611      0.389
##  3 female               195  186. female             0.756      0.244
##  4 <NA>                  NA   NA  <NA>              NA         NA
##  5 female               193  189. female             0.708      0.292
##  6 male                 190  192. female             0.654      0.346
##  7 female               181  192. female             0.661      0.339
##  8 male                 195  208. male               0.347      0.653
##  9 <NA>                 193  190. female             0.701      0.299
```

```
## 10 <NA>                 190   202. male               0.473      0.527
## # ... with 334 more rows
```

**Question:** What is the predicted flipper length for the first penguin?

**Question:** What is the predicted probability of being male for the first penguin?

## Categorical metrics

For most of the metrics, we will use the hard classification for the categorical variable as given by `.pred_class`.

We can get the confusion matrix:

```
penguins_pred %>%
  conf_mat(
    truth = sex,
    estimate = .pred_class
  )
```

```
##           Truth
## Prediction female male
##     female    109   74
##     male       56   94
```

**Question:** How many of the females, we incorrectly predicted as male?

We can get the sensitivity as follows:

```
penguins_pred %>%
  sens(
    truth = sex,
    estimate = .pred_class
  )
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 sens    binary         0.661
```

**Question:** What is the specificity?

We can obtain a set of the metrics as follows:

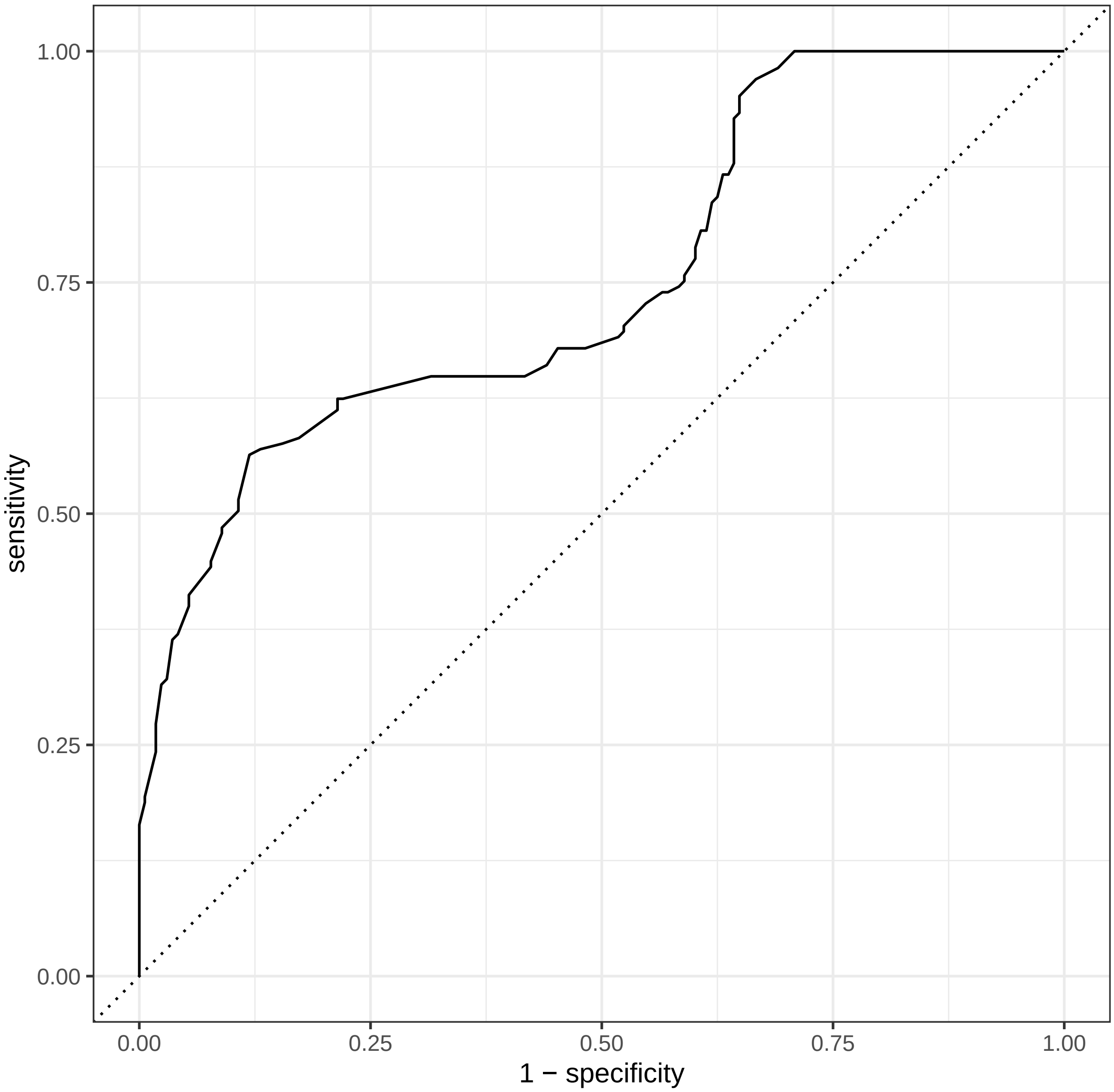
```
categorical_metrics <- metric_set(sens, spec, precision, recall)
penguins_pred %>%
  categorical_metrics(
    truth = sex,
    estimate = .pred_class
  )
```

```
## # A tibble: 4 x 3
##   .metric   .estimator .estimate
##   <chr>     <chr>          <dbl>
## 1 sens      binary         0.661
## 2 spec      binary         0.560
## 3 precision binary         0.596
## 4 recall    binary         0.661
```

**Question:** What is the precision?

We can plot the ROC curve using ggplot2, or quickly using `autoplot()`

```r
penguins_pred %>%
  roc_curve(
    truth = sex,
    estimate = .pred_female
  ) %>%
  autoplot()
```




**Question:** What is the AUC for M2?

We can obtain this as follows:

```r
penguins_pred %>%
  roc_auc(
    truth = sex,
    estimate = .pred_female
  )
```

```
## # A tibble: 1 x 3
```

```
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.752
```

# Part 2

Now we are going to show how to split your data into folds for cross-validation. We will use then use cross-validation to get a more accurate measure of how well our models fit.

# Load and split the data

Back to the penguins - why would you not?

First we are going to split our dataset into a test data to save for the very end, and the training data.

```
set.seed(2021)
penguin_split <- initial_split(penguins)
penguin_split
```

```
## <Analysis/Assess/Total>
## <258/86/344>
```

```
penguins_train <- training(penguin_split)
penguins_test <- testing(penguin_split)
```

**Question:** How many penguins in the test dataset?

Now we are going to split our training dataset into folds:

```
penguin_CV <- vfold_cv(penguins_train)
penguin_CV
```

```
## #  10-fold cross-validation
## # A tibble: 10 x 2
##    splits           id
##    <list>           <chr>
##  1 <split [232/26]> Fold01
##  2 <split [232/26]> Fold02
##  3 <split [232/26]> Fold03
##  4 <split [232/26]> Fold04
##  5 <split [232/26]> Fold05
##  6 <split [232/26]> Fold06
##  7 <split [232/26]> Fold07
##  8 <split [232/26]> Fold08
##  9 <split [233/25]> Fold09
## 10 <split [233/25]> Fold10
```

**Question:** How many folds are produced?

# Fit models and get measures

We will set up two workflows - one regression, and one classification.

For linear regression:

```
linear_model <-
  linear_reg() %>%
  set_engine("lm")
```

```
penguin_linear_workflow <-
  workflow() %>%
  add_model(linear_model) %>%
  add_formula(bill_length_mm ~ body_mass_g)
```

For logistic regression:

```
logistic_model <-
  logistic_reg() %>%
  set_engine("glm")
```

```
penguin_logistic_workflow <-
  workflow() %>%
  add_model(logistic_model) %>%
  add_formula(sex ~ body_mass_g)
```

The key function is `fit_resamples`. This function will take a workflow, and folds and preform multiple fits. It fits the model to the CV training dataset, and then fits the model to the test CV and grabs some metrics.

```
penguin_linear_resamples <-
  fit_resamples(
    penguin_linear_workflow,
    resamples = penguin_CV
  )
penguin_linear_resamples
```

```
## # Resampling results
## # 10-fold cross-validation
## # A tibble: 10 x 4
##    splits           id     .metrics         .notes
##    <list>           <chr>  <list>           <list>
##  1 <split [232/26]> Fold01 <tibble [2 x 4]> <tibble [0 x 3]>
##  2 <split [232/26]> Fold02 <tibble [2 x 4]> <tibble [0 x 3]>
##  3 <split [232/26]> Fold03 <tibble [2 x 4]> <tibble [0 x 3]>
##  4 <split [232/26]> Fold04 <tibble [2 x 4]> <tibble [0 x 3]>
##  5 <split [232/26]> Fold05 <tibble [2 x 4]> <tibble [0 x 3]>
##  6 <split [232/26]> Fold06 <tibble [2 x 4]> <tibble [0 x 3]>
##  7 <split [232/26]> Fold07 <tibble [2 x 4]> <tibble [0 x 3]>
##  8 <split [232/26]> Fold08 <tibble [2 x 4]> <tibble [0 x 3]>
##  9 <split [233/25]> Fold09 <tibble [2 x 4]> <tibble [0 x 3]>
## 10 <split [233/25]> Fold10 <tibble [2 x 4]> <tibble [0 x 3]>
```

If we want to keep the prediction values on the CV test set, then we use:

```
control = control_resamples(save_pred = TRUE)
```

Here is an example with the logistic regression.

```
penguin_logistic_resamples <-
  fit_resamples(
    penguin_logistic_workflow,
    resamples = penguin_CV,
    control = control_resamples(save_pred = TRUE)
  )
```

We can now get the metrics out. `Unnest` returns all of them, while `collect_metrics` gives us the average:

```
penguin_linear_resamples %>% unnest(.metrics)
```

```
## # A tibble: 20 x 7
##    splits           id     .metric .estimator .estimate .config          .notes
##    <list>           <chr>  <chr>   <chr>          <dbl> <chr>            <list>
##  1 <split [232/26]> Fold01 rmse    standard       4.41  Preprocessor1_~ <tibble>
##  2 <split [232/26]> Fold01 rsq     standard       0.305 Preprocessor1_~ <tibble>
##  3 <split [232/26]> Fold02 rmse    standard       3.54  Preprocessor1_~ <tibble>
##  4 <split [232/26]> Fold02 rsq     standard       0.594 Preprocessor1_~ <tibble>
##  5 <split [232/26]> Fold03 rmse    standard       3.08  Preprocessor1_~ <tibble>
##  6 <split [232/26]> Fold03 rsq     standard       0.645 Preprocessor1_~ <tibble>
##  7 <split [232/26]> Fold04 rmse    standard       4.67  Preprocessor1_~ <tibble>
##  8 <split [232/26]> Fold04 rsq     standard       0.288 Preprocessor1_~ <tibble>
##  9 <split [232/26]> Fold05 rmse    standard       3.55  Preprocessor1_~ <tibble>
## 10 <split [232/26]> Fold05 rsq     standard       0.535 Preprocessor1_~ <tibble>
## 11 <split [232/26]> Fold06 rmse    standard       4.19  Preprocessor1_~ <tibble>
## 12 <split [232/26]> Fold06 rsq     standard       0.319 Preprocessor1_~ <tibble>
## 13 <split [232/26]> Fold07 rmse    standard       4.59  Preprocessor1_~ <tibble>
## 14 <split [232/26]> Fold07 rsq     standard       0.150 Preprocessor1_~ <tibble>
## 15 <split [232/26]> Fold08 rmse    standard       5.54  Preprocessor1_~ <tibble>
## 16 <split [232/26]> Fold08 rsq     standard       0.206 Preprocessor1_~ <tibble>
## 17 <split [233/25]> Fold09 rmse    standard       5.01  Preprocessor1_~ <tibble>
## 18 <split [233/25]> Fold09 rsq     standard       0.527 Preprocessor1_~ <tibble>
## 19 <split [233/25]> Fold10 rmse    standard       4.40  Preprocessor1_~ <tibble>
## 20 <split [233/25]> Fold10 rsq     standard       0.275 Preprocessor1_~ <tibble>
```

```
penguin_linear_resamples %>% collect_metrics()
```

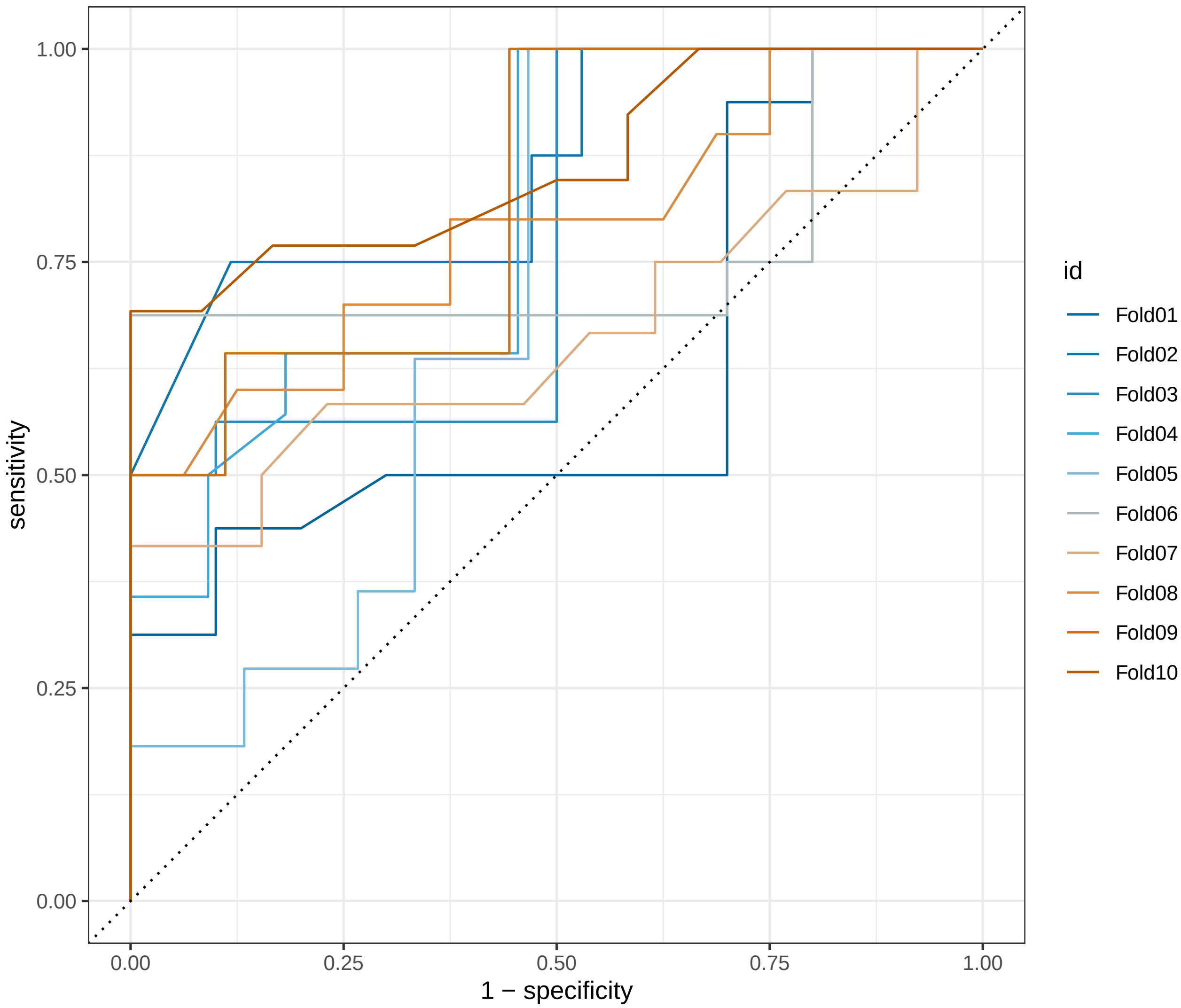
```
## # A tibble: 2 x 6
##   .metric .estimator  mean     n std_err .config
##   <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1 rmse    standard   4.30     10  0.234  Preprocessor1_Model1
## 2 rsq     standard   0.384    10  0.0551 Preprocessor1_Model1
```

**Question:** What is the RMSE for the first fold?

**Question:** What is the mean CV RMSE?

You can produce a plot for the logistic regression model as follows:

```
penguin_logistic_resamples %>%
  collect_predictions() %>%
  group_by(id) %>%
  roc_curve(truth = sex, estimate = .pred_female) %>%
  autoplot() +
  harrypotter::scale_color_hp("Ravenclaw", discrete = TRUE)
```




## Compare to test

Finally, we can get the metrics for the test data we saved at the start using `last_fit`

```
penguin_linear_workflow %>%
  last_fit(penguin_split) %>%
  collect_metrics()
```

```
## # A tibble: 2 x 4
##   .metric .estimator .estimate .config
##   <chr>   <chr>          <dbl> <chr>
## 1 rmse    standard       4.58  Preprocessor1_Model1
## 2 rsq     standard       0.323 Preprocessor1_Model1
```

```
penguin_logistic_workflow %>%
  last_fit(penguin_split) %>%
  collect_metrics()
```

```
## # A tibble: 2 x 4
##   .metric  .estimator .estimate .config
```

```
##   <chr>    <chr>          <dbl> <chr>
## 1 accuracy binary         0.6   Preprocessor1_Model1
## 2 roc_auc  binary         0.771 Preprocessor1_Model1
```

**Question** What is the test AUC?