MATHS 7107
Data Taming
Practical 12

Classification trees

Preliminaries

- Set up a project in RStudio
- Now load the packages
  - tidyverse
  - tidymodels
  - vip
  - ISLR
  - rpart.plot
- Then load the dataset Carseats as a tibble.
  - Save the dataset as a new name, to avoid confusion. We'll assume you name the tibble car_seats.
  - Look at the Carseats help file to see what the variables mean. (Eg. use F1, help() or ?.)

1 Classification tree

1.1 Data taming and cleaning

We will now build a classification tree to predict whether sales are high or low (rather than giving a numerical estimate of sales).

First, let's make this binary variable called sales_high which equals

- high if at least 8,000 sales were made at the store
- low if fewer than 8,000 sales were made at the store

We'll put this new variable just to the right of Sales.

```
car_seats <- car_seats %>%
  mutate("sales_high" = ifelse(Sales>8,"high","low"), .after = Sales)
car_seats
```

```
## # A tibble: 400 x 12
##    Sales sales_high CompPrice Income Advertising Population Price ShelveLoc
##    <dbl> <chr>          <dbl>  <dbl>       <dbl>      <dbl> <dbl> <fct>
##  1  9.5  high             138     73          11        276   120 Bad
##  2 11.2  high             111     48          16        260    83 Good
##  3 10.1  high             113     35          10        269    80 Medium
##  4  7.4  low              117    100           4        466    97 Medium
##  5  4.15 low              141     64           3        340   128 Bad
```

```
##  6 10.8  high             124    113          13        501    72 Bad
##  7  6.63 low              115    105           0         45   108 Medium
##  8 11.8  high             136     81          15        425   120 Good
##  9  6.54 low              132    110           0        108   124 Medium
## 10  4.69 low              132    113           0        131   124 Medium
## # i 390 more rows
## # i 4 more variables: Age <dbl>, Education <dbl>, Urban <fct>, US <fct>
```

Questions:

1. Convert this variable to a factor. (This is so we can apply a tree model to it.)
2. Remove the `Sales` variable from the dataset, and save it as a new tibble called `car_seats_1`.
    - Why do we need to remove `Sales`? What would our model find if we used it as one of the predictors for `sales_high`?

1.2 Fitting the model

Questions:

3. Using the seed `2023` split the data into a testing set `car_ctest` and a training set `car_ctrain`.
4. Make a model specification for a **classification tree**:

```
decision_tree(mode = "classification") %>%
  set_engine("rpart")
```

5. Using `sales_high` as the response variable, fit a classification tree to the **training set** with all other variables as predictors. We'll assume that you name the tree `car_classtree`.

```
car_classtree <- <model specification> %>%
  fit(sales_high~.,data = car_ctrain)
```

   - *Hint: you can use the formula* `<response>` $\sim$ `.` *to use all non-response variables as predictors.*

6. Produce a tree diagram of the classification tree using the command

`rpart.plot(car_classtree$fit, extra=4, type=2)`

7. Obtain a `vip` plot for this model. What is the most important variable in classifying high sales at a store? Does that match what you see in the tree diagram?
8. Does our model predict more than 8,000 sales at a store if the shelf location is bad, the price is $98, advertising expenditure is $18,500 and the average age of the local population was 43? (Give the percentage of our prediction as well.)

2 Testing the model

Questions:

9. Calculate the predicted high or low sales at each store in the **testing** set. Do this using `bind_cols()` to make a tibble containing:
    - the predicted class "high" or "low"

- the probabilities for the predicted class
- the truth given by the `sales_high` variable

10. Find the confusion matrix for this prediction and calculate the **accuracy**.

11. If `sales_low` is considered a "success" in this model, calculate the **sensitivity** and **specificity**. (Do this manually, and then see if the `R` command gives the same answer.)

12. Find the correct ROC curve. If you store the sensitivity and specificity as `sens1` and `spec1` then you can add the following code to your `autoplot()` to check for the correct ROC curve:

```
+ geom_vline(xintercept = 1-spec1) + geom_hline(yintercept = sens1)
```

13. Once you have the correct ROC curve, find the AUC.

3 Cross-validation

Questions:

14. Using the seed `2023` make a 5-fold cross-validation set:

```
set.seed(2023)
car_cv <- vfold_cv(car_seats_1,v=5)
```

15. Again using the seed `2023`, apply to model to the cross-validation set:

```
set.seed(2023)
class_resamples <- fit_resamples(
  object = car_ct_spec,
  sales_high ~ .,
  resamples = car_cv,
  control = control_resamples( save_pred = T ) #To keep the predictions
)
```

16. Use the commands `collect_metrics()` and `collect_predictions()` to find the cross-validated metrics and predictions.

17. Plot the ROC curves for all the cross-validation sets:

```
class_resamples %>%
  collect_predictions() %>%
  group_by(id) %>%
  roc_curve(.pred_low, truth=sales_high, event_level="second") %>%
  autoplot()
```