STATS 7022 - Data Science PG Assignment 3

Dang Thinh Nguyen

2024-08-07

Question 1: Modelling with Data

1 Read in the Data

```
# Read in the data
data <- readRDS('./diamonds2.rds')

# Display the first 10 lines of the data
data %>% head(10)
```

```
## # A tibble: 10 x 10
##    carat depth table price     x     y cut       colour clarity     z
##    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <fct>     <fct>  <fct>   <dbl>
##  1  1.11  62.3    56  4645  6.59  6.64 ideal     F      SI1      4.12
##  2  1.24  58.9    59  5972  7.08  7.02 premium   F      SI1      4.15
##  3  1     64.5    59  4541  6.25  6.31 good      A      SI2      4.05
##  4  1.13  61.4    57  4936  6.72  6.69 ideal     F      SI1      4.12
##  5  0.31  63.5    56   571  4.29  4.31 good      A      SI1      2.73
##  6  0.51  61.5    55  1438  5.16  5.18 ideal     B      SI1      3.18
##  7  0.32  60.9    57   612  4.46  4.44 ideal     D      SI1      2.71
##  8  0.3   63.2    57   675  4.3   4.25 very good B      SI1      2.7
##  9  1.21  61.8    56  8863  6.82  6.86 ideal     B      SI1      4.23
## 10  0.78  64.6    56  2359  5.86  5.79 fair      D      SI1      3.76
```

2 Data Splitting

```
# Set the seed
set.seed(2024)

# Split data
data_split <- initial_split(data, strata = price)
data_train <- training(data_split)
data_test <- testing(data_split)

# Display the training/testing/total sets
data_split
```

```
## <Training/Testing/Total>
## <3862/1289/5151>
```

3 Cross-validation

```
# Set folds cross-validation
data_folds <- vfold_cv(data_train, v = 15, strata = price)

# Display the folds
data_folds
```

```
## #  15-fold cross-validation using stratification
## # A tibble: 15 x 2
##    splits             id
##    <list>             <chr>
##  1 <split [3602/260]> Fold01
##  2 <split [3602/260]> Fold02
##  3 <split [3602/260]> Fold03
##  4 <split [3602/260]> Fold04
##  5 <split [3602/260]> Fold05
##  6 <split [3604/258]> Fold06
##  7 <split [3606/256]> Fold07
##  8 <split [3606/256]> Fold08
##  9 <split [3606/256]> Fold09
## 10 <split [3606/256]> Fold10
## 11 <split [3606/256]> Fold11
## 12 <split [3606/256]> Fold12
## 13 <split [3606/256]> Fold13
## 14 <split [3606/256]> Fold14
## 15 <split [3606/256]> Fold15
```

4 Recipe

```
# Set recipe
data_recipe <- recipe(price ~ ., data = data_train) %>%
  step_log(price)

# Display the recipe
data_recipe
```

5 Workflow

```
# Set model
rf_model <- rand_forest(mtry = tune(),
                        min_n = tune(),
                        trees = 500) %>%
  set_engine('ranger') %>%
  set_mode('regression')

# Set workflow
wf <- workflow() %>%
```

```
  add_recipe(data_recipe) %>%
  add_model(rf_model)

# Display the workflow
wf
```

```
## == Workflow ====================================================================
## Preprocessor: Recipe
## Model: rand_forest()
##
## -- Preprocessor ----------------------------------------------------------------
## 1 Recipe Step
##
## * step_log()
##
## -- Model -----------------------------------------------------------------------
## Random Forest Model Specification (regression)
##
## Main Arguments:
##   mtry = tune()
##   trees = 500
##   min_n = tune()
##
## Computational engine: ranger
```

6 Model Tuning

```
# Set candidates
data_grid <- grid_regular(mtry(c(1,9)),
                          min_n(),
                          levels = 5)

# Display the grid
data_grid
```

```
## # A tibble: 25 x 2
##     mtry min_n
##    <int> <int>
##  1     1     2
##  2     3     2
##  3     5     2
##  4     7     2
##  5     9     2
##  6     1    11
##  7     3    11
##  8     5    11
##  9     7    11
## 10     9    11
## # i 15 more rows
```

```r
# Tune
doParallel::registerDoParallel()
tune_res <- tune_grid(wf,
                      resamples = data_folds,
                      grid = data_grid)

# Plot tuning
tune_res %>% autoplot() +
  labs(title = 'Tuning results',
       x = 'Number of predictors')
```

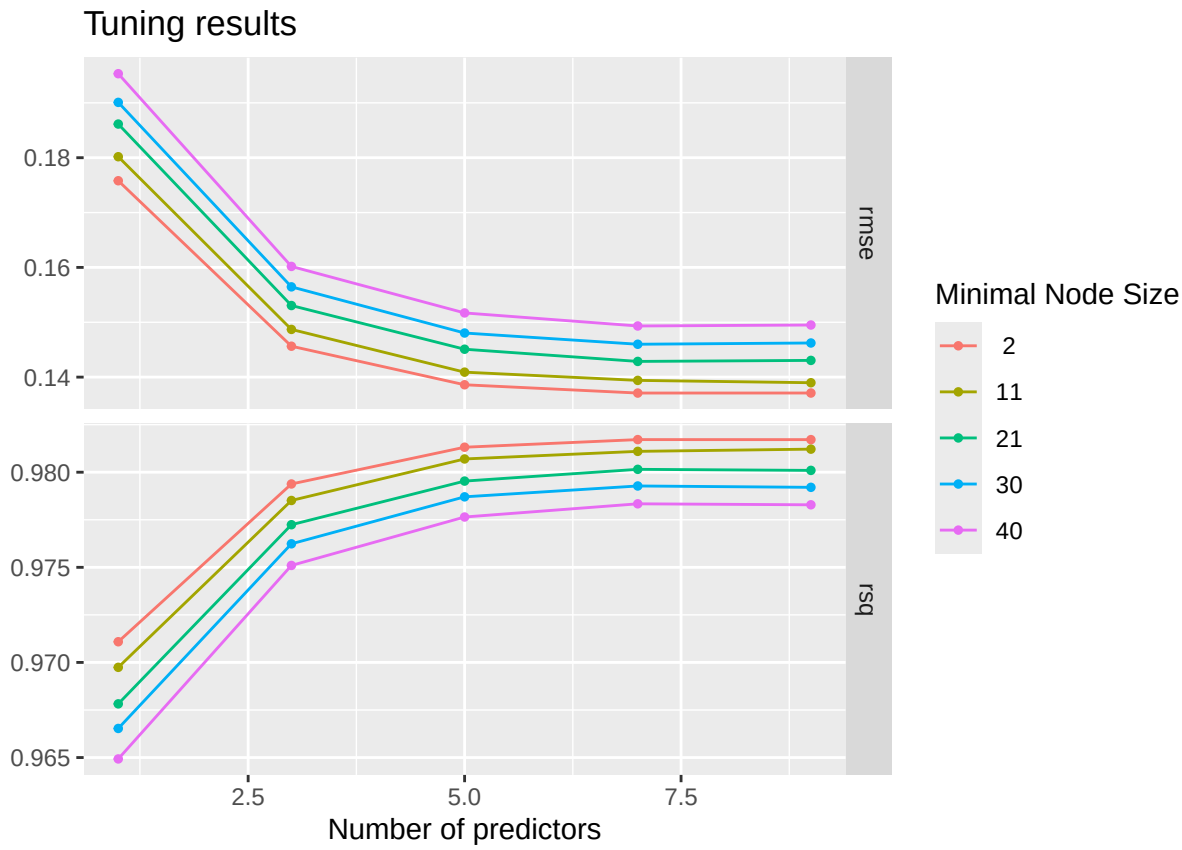

Figure 1: Comparison of model performance across hyperparameters

```r
# Show the top 5 best hyperparameters
show_best(tune_res, metric = 'rmse')
```

```
## # A tibble: 5 x 8
##    mtry min_n .metric .estimator  mean     n std_err .config
##   <int> <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1     7     2 rmse    standard   0.137    15 0.00254 Preprocessor1_Model04
## 2     9     2 rmse    standard   0.137    15 0.00267 Preprocessor1_Model05
## 3     5     2 rmse    standard   0.139    15 0.00230 Preprocessor1_Model03
## 4     9    11 rmse    standard   0.139    15 0.00252 Preprocessor1_Model10
## 5     7    11 rmse    standard   0.139    15 0.00240 Preprocessor1_Model09
```

```r
# Finalize the workflow with the best parameters
final_workflow <- wf %>%
  finalize_workflow(select_best(tune_res, metric = 'rmse'))

# Display the final workflow
final_workflow
```

```
## == Workflow ====================================================================
## Preprocessor: Recipe
## Model: rand_forest()
##
## -- Preprocessor ----------------------------------------------------------------
## 1 Recipe Step
##
## * step_log()
##
## -- Model -----------------------------------------------------------------------
## Random Forest Model Specification (regression)
##
## Main Arguments:
##   mtry = 7
##   trees = 500
##   min_n = 2
##
## Computational engine: ranger
```

7 Model fit

```r
# Fit the best model
final_fit <- last_fit(final_workflow, data_split)

# Evaluate the model on the test data
final_fit %>%
  collect_metrics()
```

```
## # A tibble: 2 x 4
##   .metric .estimator .estimate .config
##   <chr>   <chr>          <dbl> <chr>
## 1 rmse    standard       0.142 Preprocessor1_Model1
## 2 rsq     standard       0.981 Preprocessor1_Model1
```