Final Project

2022-06-09

Contents

Introduction

The purpose of this project is to find out what factors make the evaluation of wine high, and ultimately pass on information to those who do not know much about wine as a result of this. This is prediction project and 'quality' variable is the target variable.

Data Cleaning

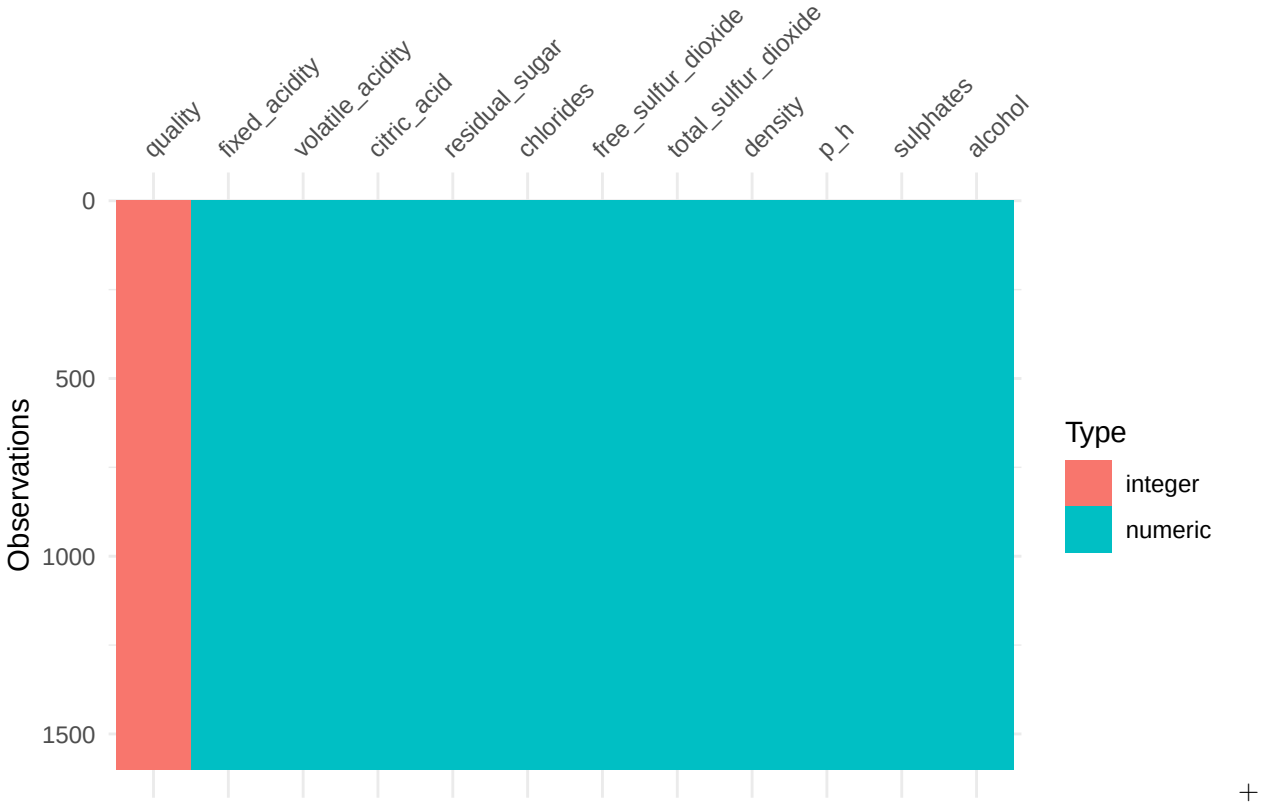
- Clean names

```
data = read.csv('wine.csv')
data = data %>% clean_names()
```

- Check NAs

```
vis_dat(data)
```

```
## Warning: `gather_()` was deprecated in tidyr 1.2.0.
## Please use `gather()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
```



There is no NA value. + All numeric variables.

- Split data

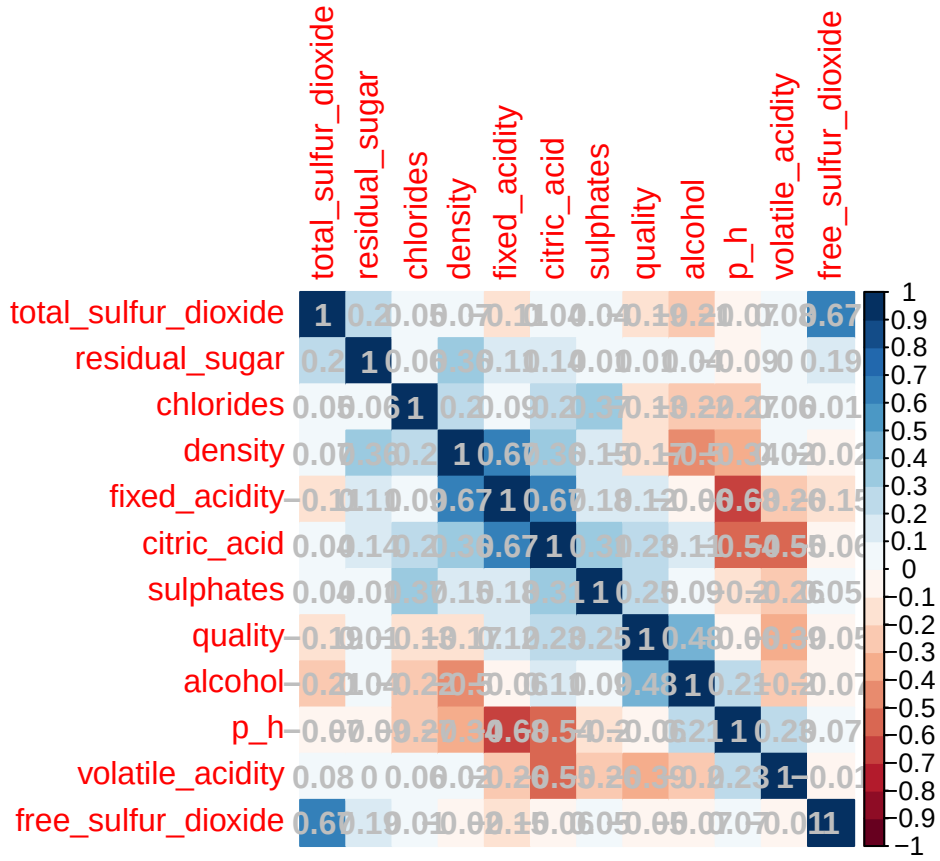
```
data_split <- data %>%
        initial_split(prop = 0.7)

train <- training(data_split)
test <- testing(data_split)
```

```
    + This is to predict 'quality' value, so I do not need to add 'strata' argument.
```

Exploratory Data Analysis

- correlation

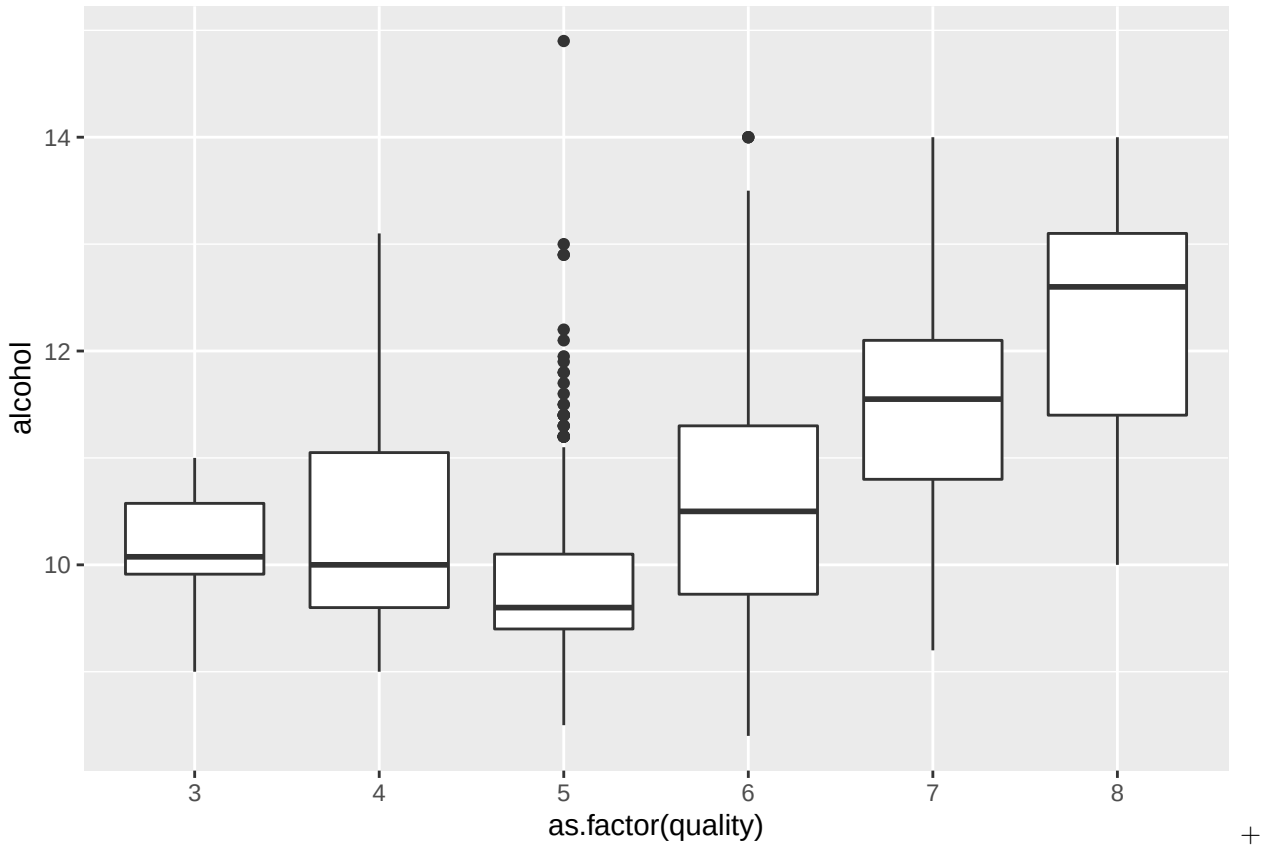
```
M = cor(data)
corrplot(M, method = 'color', col = COL2(n=20), cl.length = 21, order = 'AOE',
         addCoef.col = 'grey')
```



+ 'free_sulfur_dioxide' and 'total_sulfur_dioxide' seem similar. + 'acidity' variables also seem similar as sulfur dioxide. + 'alcohol' is the highest correlated variable with the target variable, 'quality'.

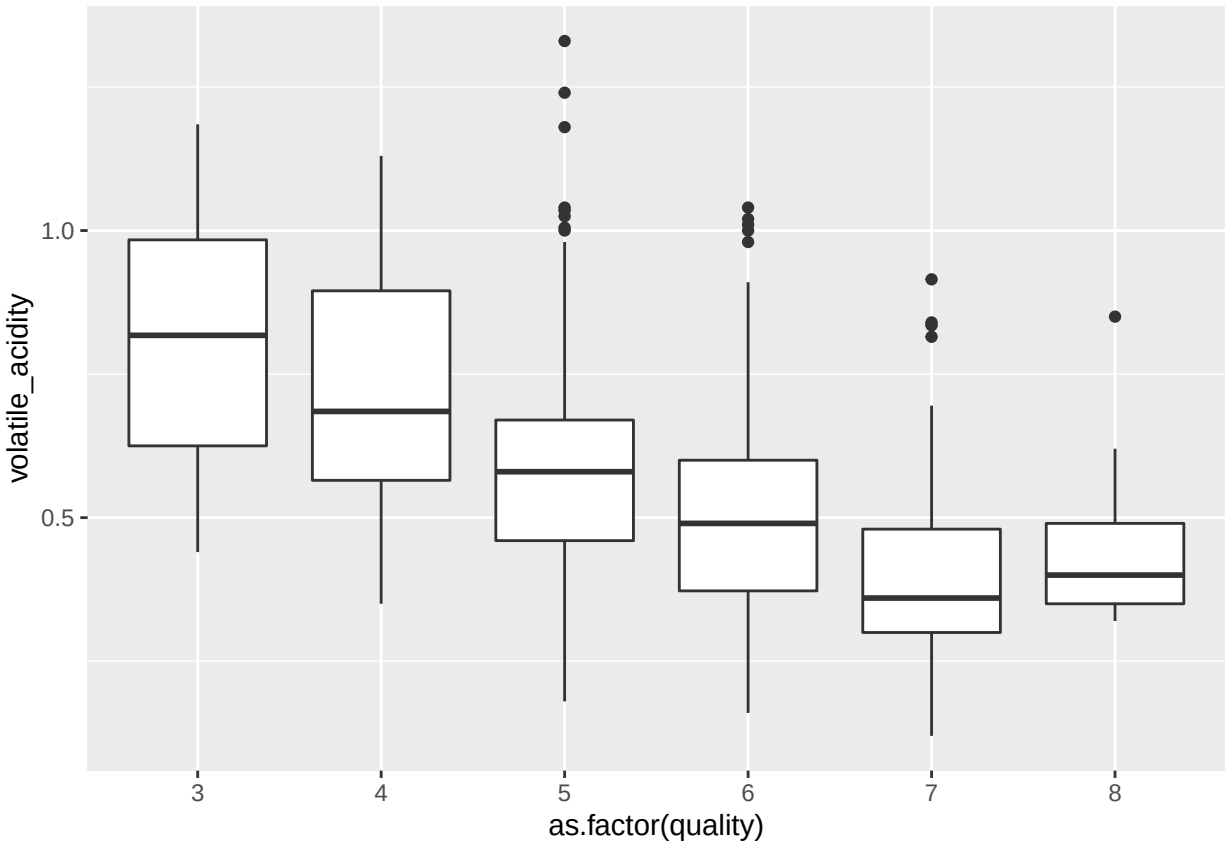
- Plot

```
train %>% ggplot(aes(x = as.factor(quality), y = alcohol)) +
        geom_boxplot()
```



As I saw, 'quality' and 'alcohol' are positivly correlated.

```
train %>% ggplot(aes(x = as.factor(quality), y = volatile_acidity)) +
        geom_boxplot()
```



On the other hand, 'volatile_acidity' is negatively correlated.

Model Building

- CV folds

```
folds <- vfold_cv(train, v = 5)
```

```
    + I divided the train set to 5 folds.
```

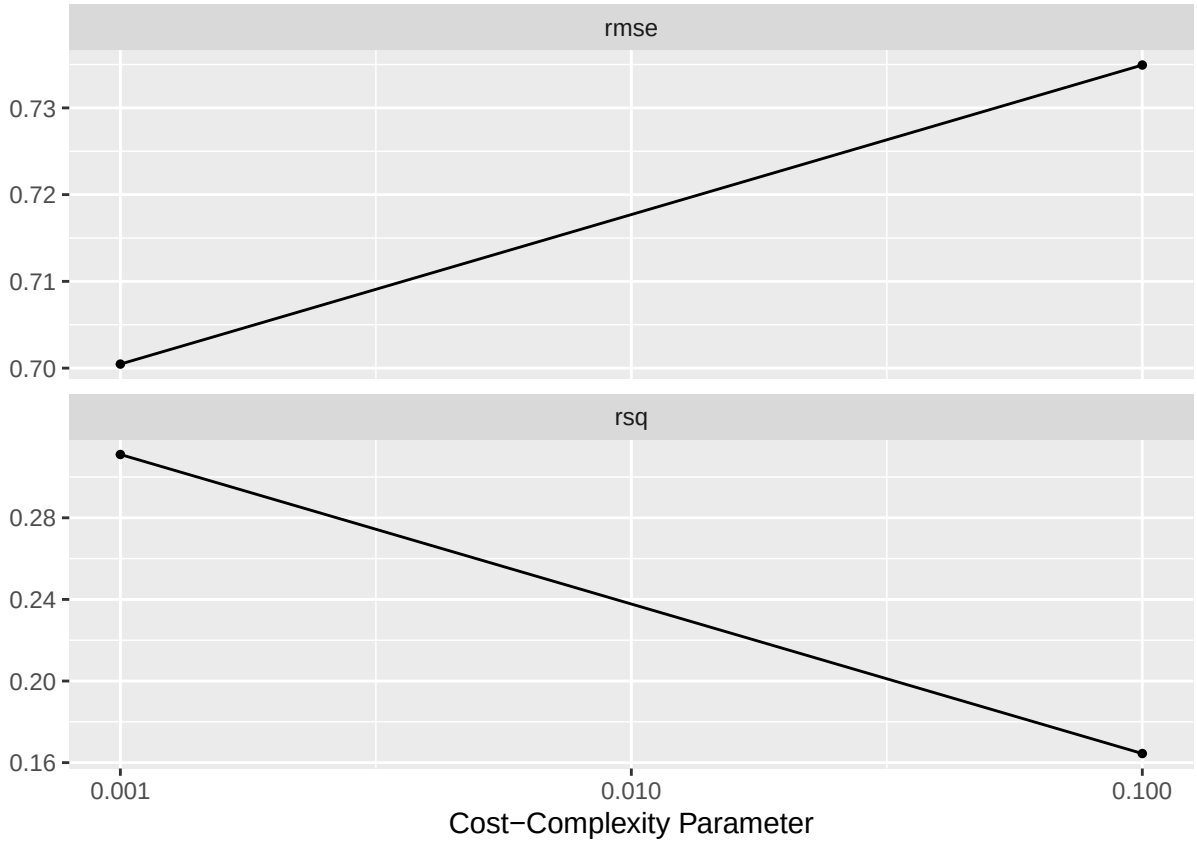
- Recipe

```
recipe = recipe(quality ~ ., data = train) %>%
        step_normalize(all_predictors())
```

- Decision Tree

```
tree_model = decision_tree(cost_complexity = tune()) %>%
        set_engine('rpart') %>%
        set_mode('regression')

tree_wf = workflow() %>%
        add_model(tree_model) %>%
        add_recipe(recipe)

tree_grid <- grid_regular(cost_complexity(range = c(-3, -1)),
                          levels = 2)

tree_tune <- tree_wf %>%
```

```
        tune_grid(resamples = folds,
        grid = tree_grid)

save(tree_tune, tree_wf, file = "tree_tune.rda")

autoplot(tree_tune)
```



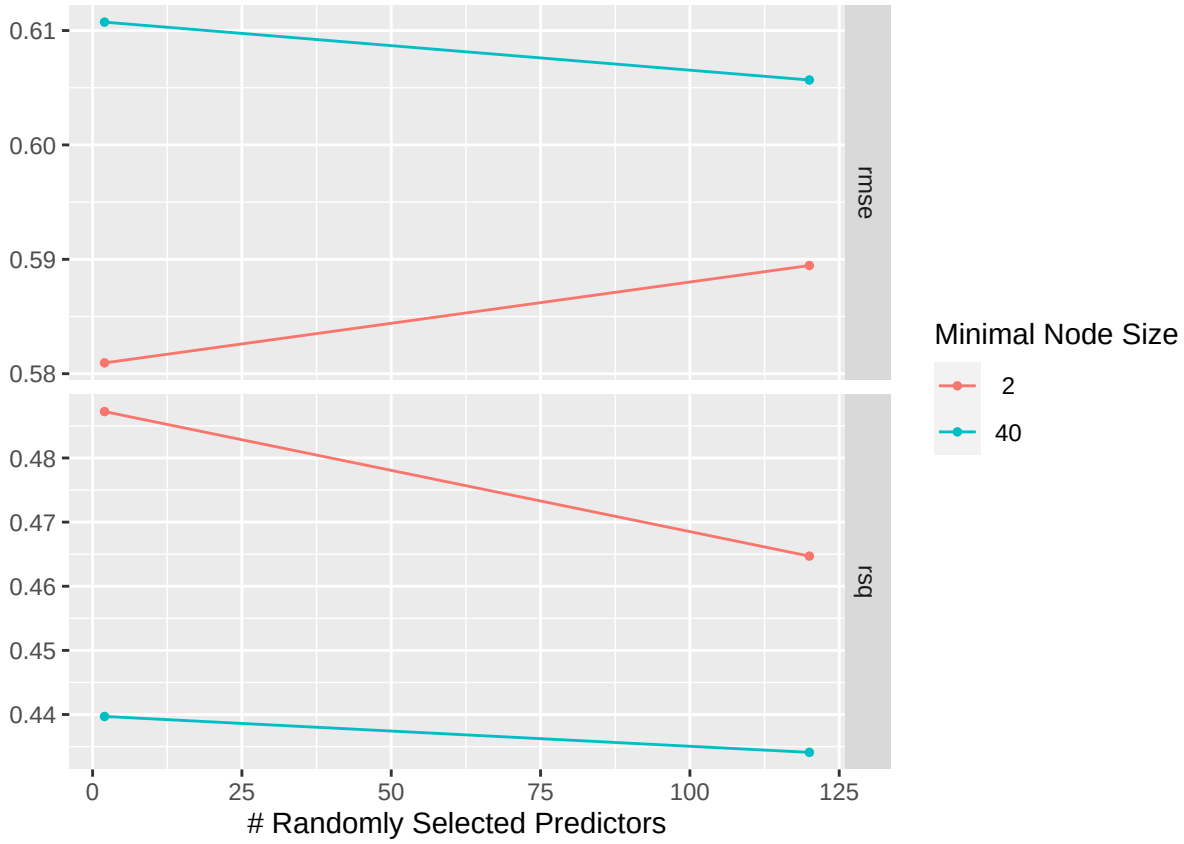
- Random Forest

```
rf_model = rand_forest(min_n = tune(),
                       mtry = tune(),
                       mode = "regression") %>%
        set_engine("ranger")

rf_wf = workflow() %>%
        add_model(rf_model) %>%
        add_recipe(recipe)

rf_params = parameters(rf_model) %>%
        update(mtry = mtry(range= c(2, 120)))
```

```
## Warning: `parameters.model_spec()` was deprecated in tune 0.1.6.9003.
## Please use `hardhat::extract_parameter_set_dials()` instead.
```

```
rf_grid = grid_regular(rf_params,
                       levels = 2)

rf_tune = rf_wf %>%
```

```
        tune_grid(resamples = folds,
                  grid = rf_grid,
                  metrics = metric_set(rmse, rsq))

save(rf_tune, rf_wf, file = "rf_tune.rda")

autoplot(rf_tune)
```

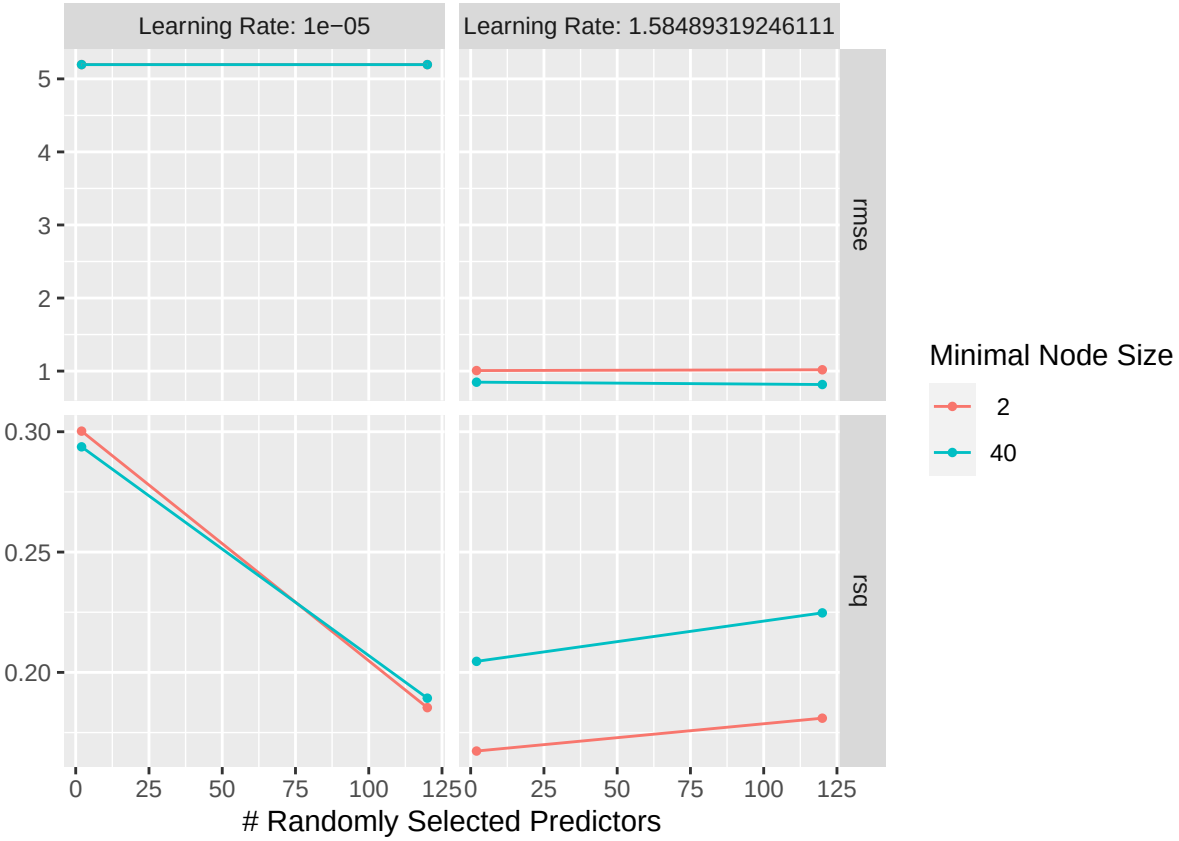


- Boosted Tree

```
bt_model = boost_tree(min_n = tune(),
                      mtry = tune(),
                      learn_rate = tune(),
                      mode = "regression") %>%
        set_engine("xgboost")

bt_wf = workflow() %>%
        add_model(bt_model) %>%
        add_recipe(recipe)

bt_params = parameters(bt_model) %>%
        update(mtry = mtry(range= c(2, 120)),
               learn_rate = learn_rate(range = c(-5, 0.2)))
```

```
## Warning: `parameters.model_spec()` was deprecated in tune 0.1.6.9003.
## Please use `hardhat::extract_parameter_set_dials()` instead.
```

```
bt_grid = grid_regular(bt_params, levels = 2)

bt_tune = bt_wf %>%
        tune_grid(resamples = folds,
                  grid = bt_grid,
                  metrics = metric_set(rmse, rsq))

save(bt_tune, bt_wf, file = "bt_tune.rda")

autoplot(bt_tune)
```



- Test set Prediction

```
rf_best = rf_tune %>% select_best(metric = 'rsq')

final = rf_wf %>% finalize_workflow(rf_best)
final_fit = final %>% fit(data = test)

augment(final_fit, new_data = test) %>%
        rsq(truth = quality, estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rsq     standard       0.950
```

Conclusion

Random forest is the best model for prediction. The higher the complexity, the higher R^2 and it indicates the model is explaining well. However, comprehensive results are quite disappointing. I should try to make a derived variable or other things to make a model works well.