Seminar(Supervised Machine Learning)

2024-09-26

library(readr)  
library(tidymodels)  
library(usemodels)

### Import Data  
apple <- read\_csv("Apple.csv")  
  
apple

The A\_id is not needed and will be dropped from the dataframe

apple <- apple |>   
 select(-A\_id)  
# Explore the data  
unique(apple$Quality)

str(apple)  
summary(apple)  
skimr::skim(apple)

apple |>   
 group\_by(Quality) |>   
 count()

From the summary, we see that the target variable is a character variable. We have to convert it to factor since it is the type required in the prediction model. Also, our interest is predict the good apple, so we have to change the levels and make good the first level.

apple <- apple |>   
 mutate(Quality = factor(Quality, levels = c("good","bad")))  
  
levels(apple$Quality)

# Check the distribution of the target variable  
  
ggplot(apple, aes(x = Quality, fill = Quality))+   
 geom\_bar()+   
 guides(fill = "none")

The two classes of the target variable are evenly distributed

################## Correlation of the variables  
  
apple |>   
 psych::pairs.panels(gap = 0,pch=21)

There is no evidence of multicolinearity among the predictors since there is no strong correlations between any two variable. Additionally, all the features are normally distributed as shown in the plot, therefore, no scaling is required.

Set the random number stream using set.seed() so that the results can be reproduced later. Split a balanced data (80/20). The data split is done using stratified sampling

set.seed(202)  
apple\_split <- initial\_split(apple, prop = 0.80, strata = Quality)  
apple\_train <- training(apple\_split)  
apple\_test <- testing(apple\_split)  
  
dim(apple\_train)  
dim(apple\_test)

The function strata divides the strata variable by four (by default) and samples within each stratum. The default number (four) can be changed using the breaks function.The change is necessary if the distribution of the test set is different from the training set

# re-sampling the training set using 10 fold CV (stratified sampling)  
set.seed(200)  
apple\_CV <- vfold\_cv(apple\_train, v = 10,   
 strata = Quality, repeats = 5)  
apple\_CV

# Preprocessing the data using the `recipe` function  
  
apple\_recipe <-   
 recipe(Quality ~., data = apple\_train) # specify the formula  
  
summary(apple\_recipe)

#Specifying the model ### STEPS 1. Choose a model 2. Specify an engine 3. Set the mode

# Specify Engine  
rf\_model <- rand\_forest(trees = 2000, min\_n = tune(),mtry = tune()) |>   
 set\_engine("ranger", verbose = TRUE) |>   
 set\_mode("classification")

* The hyperparameters for random forest are the number of trees (tress),
* mtry: The number of features to randomly select select at each split
* min\_n: A minimum number of data points in a node that are required for the node to be split further

# Setting the workflow

A workflow is a container object that aggregates information required to fitand predict from a model

rf\_wflow <-   
 workflow() |>   
 add\_model(rf\_model) |>   
 add\_recipe(apple\_recipe)

# Setting evaluation metrics

* accuracy: the proportion of the data that are predicted correctly
* roc\_auc: Area Under the Receiver Operating Characteristic Curve
* pr\_auc: Area Under the Precision-Recall Curve

rf.reg\_metric <- metric\_set(accuracy,roc\_auc, pr\_auc)

# Tune hyperparameters

rf\_grid <- grid\_regular(  
 mtry(range = c(3, 7)),  
 min\_n(range = c(5, 20)),  
 levels = 5  
)

# Control aspects of the grid search process  
ctrl <- control\_resamples(save\_pred = TRUE, verbose = TRUE)

# Search grid  
set.seed(203)  
start.time <- Sys.time()  
apple\_rf\_model <-   
 tune\_grid(  
 rf\_wflow,  
 resamples = apple\_CV,  
 control = ctrl,   
 metrics = rf.reg\_metric  
 )  
end.time <- Sys.time()  
run.time <- end.time - start.time  
run.time

The run time is 2.135667 hours

# View results  
collect\_metrics(apple\_rf\_model)  
  
collect\_predictions(apple\_rf\_model)

# Find the best hyperparameter configuration   
apple\_rf\_model |>   
 show\_best(metric = "accuracy")

The best hyperparameters configuration here based on accuracy is trees=2000, mtry =5 and min\_n=10.

best\_params <- apple\_rf\_model |>   
 select\_best(metric = "accuracy")  
  
# Final model  
final\_workflow <- rf\_wflow |>   
 finalize\_workflow(best\_params)  
  
final\_workflow  
  
final\_model <- fit(final\_workflow, data = apple\_train,)

# Performance of the training set

* Confusion matrix A confusion matrix is a table that is used to define the performance of a classification algorithm.

augment(final\_model, apple\_train) |>   
 conf\_mat(truth = Quality, estimate = .pred\_class) |>   
 autoplot(type = "heatmap")

The confusion matrix shows that 6 good apples are wrongly classified as bad and 7 bad apples are wrongly classified as good.

# setting the metrics  
metrix <- metric\_set(accuracy, sensitivity, specificity)  
# Accuracy is the proportion of all classifications that were correct, whether   
# positive or negative

Sensitivity (recall) measures the proportion of actual positives that are correctly identified by the model as positive. TP/(TP+FN)

Specificity measures the proportion of actual negatives that are correctly identified as negative TN/(TN+FP)

augment(final\_model, apple\_train) |>   
 metrix(truth = Quality, estimate = .pred\_class)  
# The accuracy of the final model is 99.6%  
# sensitivity is 99.6%  
# specificity is 99.6%

# Test set

Using a workflow method

set.seed(205)  
result <- final\_workflow |>   
 last\_fit(apple\_split, metrics = rf.reg\_metric)  
  
result

result |> collect\_metrics()  
  
mod\_pred <- collect\_predictions(result)  
  
augment(final\_model, apple\_test) |>   
 metrix(truth = Quality, estimate = .pred\_class)

For the test set,the accuracy is 88.8%, sensitivity is 91.8% and specificity is 85.8%

augment(final\_model, apple\_test) |>   
 conf\_mat(truth = Quality, estimate = .pred\_class) |>   
 autoplot(type = "heatmap")

The confusion matrix shows that 33 bad apples are wrongly classified as good and 57 good are wrongly classified as bad.