Tidy Tuesday exercise 2

#Packages Load

library(readr)  
library(here)  
library(tidyverse)  
library(skimr)  
library(rsample)  
library(recipes)  
library(rpart)  
library(parsnip)  
library(doParallel) # for parallel computing   
library(workflows)  
library(dials)  
library(dplyr)  
library(tune)  
library(rpart.plot)  
library(broom.mixed)  
library(ggplot2)

#Data Import

marbles <- read\_csv('https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/data/2020/2020-06-02/marbles.csv')

## Rows: 256 Columns: 14

## ── Column specification ────────────────────────────────────────────────────────  
## Delimiter: ","  
## chr (9): date, race, site, source, marble\_name, team\_name, pole, host, notes  
## dbl (5): time\_s, points, track\_length\_m, number\_laps, avg\_time\_lap

##   
## ℹ Use `spec()` to retrieve the full column specification for this data.  
## ℹ Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

#Exploration

glimpse(marbles)

## Rows: 256  
## Columns: 14  
## $ date <chr> "15-Feb-20", "15-Feb-20", "15-Feb-20", "15-Feb-20", "15…  
## $ race <chr> "S1Q1", "S1Q1", "S1Q1", "S1Q1", "S1Q1", "S1Q1", "S1Q1",…  
## $ site <chr> "Savage Speedway", "Savage Speedway", "Savage Speedway"…  
## $ source <chr> "https://youtu.be/JtsQ\_UydjEI?t=356", "https://youtu.be…  
## $ marble\_name <chr> "Clementin", "Starry", "Momo", "Yellow", "Snowy", "Razz…  
## $ team\_name <chr> "O'rangers", "Team Galactic", "Team Momo", "Mellow Yell…  
## $ time\_s <dbl> 28.11, 28.37, 28.40, 28.70, 28.71, 28.72, 28.96, 29.11,…  
## $ pole <chr> "P1", "P2", "P3", "P4", "P5", "P6", "P7", "P8", "P9", "…  
## $ points <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,…  
## $ track\_length\_m <dbl> 12.81, 12.81, 12.81, 12.81, 12.81, 12.81, 12.81, 12.81,…  
## $ number\_laps <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 10, 10,…  
## $ avg\_time\_lap <dbl> 28.11, 28.37, 28.40, 28.70, 28.71, 28.72, 28.96, 29.11,…  
## $ host <chr> "No", "No", "No", "No", "No", "No", "No", "No", "No", "…  
## $ notes <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,…

skim(marbles)

Data summary

|  |  |
| --- | --- |
| Name | marbles |
| Number of rows | 256 |
| Number of columns | 14 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Column type frequency: |  |
| character | 9 |
| numeric | 5 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Group variables | None |

**Variable type: character**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| skim\_variable | n\_missing | complete\_rate | min | max | empty | n\_unique | whitespace |
| date | 0 | 1.00 | 8 | 9 | 0 | 16 | 0 |
| race | 0 | 1.00 | 4 | 4 | 0 | 16 | 0 |
| site | 0 | 1.00 | 7 | 15 | 0 | 8 | 0 |
| source | 0 | 1.00 | 34 | 34 | 0 | 16 | 0 |
| marble\_name | 0 | 1.00 | 4 | 9 | 0 | 32 | 0 |
| team\_name | 0 | 1.00 | 6 | 16 | 0 | 16 | 0 |
| pole | 128 | 0.50 | 2 | 3 | 0 | 16 | 0 |
| host | 0 | 1.00 | 2 | 3 | 0 | 2 | 0 |
| notes | 249 | 0.03 | 37 | 100 | 0 | 7 | 0 |

**Variable type: numeric**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| skim\_variable | n\_missing | complete\_rate | mean | sd | p0 | p25 | p50 | p75 | p100 | hist |
| time\_s | 3 | 0.99 | 190.84 | 169.13 | 17.76 | 28.40 | 36.28 | 338.16 | 492.01 | ▇▁▁▇▁ |
| points | 128 | 0.50 | 6.45 | 7.74 | 0.00 | 0.00 | 3.00 | 11.25 | 26.00 | ▇▂▂▁▁ |
| track\_length\_m | 0 | 1.00 | 13.22 | 0.95 | 11.90 | 12.62 | 13.02 | 14.13 | 14.55 | ▅▅▂▁▇ |
| number\_laps | 0 | 1.00 | 6.25 | 5.53 | 1.00 | 1.00 | 5.00 | 10.25 | 16.00 | ▇▁▃▂▂ |
| avg\_time\_lap | 3 | 0.99 | 29.70 | 5.55 | 17.76 | 25.94 | 30.05 | 33.65 | 41.62 | ▃▆▇▇▂ |

select\_data = marbles %>% select(c(race, site,marble\_name,team\_name,time\_s,track\_length\_m,number\_laps,avg\_time\_lap))  
marbles\_updated = select\_data %>% drop\_na()  
marbles\_updated$track\_length\_m = as.factor(marbles\_updated$track\_length\_m)  
marbles\_updated$number\_laps = as.factor(marbles\_updated$number\_laps)  
marbles\_updated$avg\_time\_lap = as.factor(marbles\_updated$avg\_time\_lap)

glimpse(marbles\_updated)

## Rows: 253  
## Columns: 8  
## $ race <chr> "S1Q1", "S1Q1", "S1Q1", "S1Q1", "S1Q1", "S1Q1", "S1Q1",…  
## $ site <chr> "Savage Speedway", "Savage Speedway", "Savage Speedway"…  
## $ marble\_name <chr> "Clementin", "Starry", "Momo", "Yellow", "Snowy", "Razz…  
## $ team\_name <chr> "O'rangers", "Team Galactic", "Team Momo", "Mellow Yell…  
## $ time\_s <dbl> 28.11, 28.37, 28.40, 28.70, 28.71, 28.72, 28.96, 29.11,…  
## $ track\_length\_m <fct> 12.81, 12.81, 12.81, 12.81, 12.81, 12.81, 12.81, 12.81,…  
## $ number\_laps <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 10, 10,…  
## $ avg\_time\_lap <fct> 28.11, 28.37, 28.4, 28.7, 28.71, 28.72, 28.96, 29.11, 2…

#calculating average time by race and saving as a new df  
avg\_race\_times <- marbles\_updated %>% group\_by(race) %>%  
 summarize(mean\_race\_time = mean(time\_s, na.rm=TRUE))

#Merge this to dataset  
marbles\_cleaned <- left\_join(marbles\_updated, avg\_race\_times, by = "race")

#Divide each marble's race time by average time/race  
marbles\_cleaned <- marbles\_cleaned %>% mutate(standardized\_time = time\_s/mean\_race\_time)

#Get median standaridized time per marble across races to order boxplots  
new\_data <- marbles\_cleaned %>%   
 group\_by(marble\_name) %>%  
 summarize(median\_standardized\_time = median(standardized\_time, na.rm=TRUE))

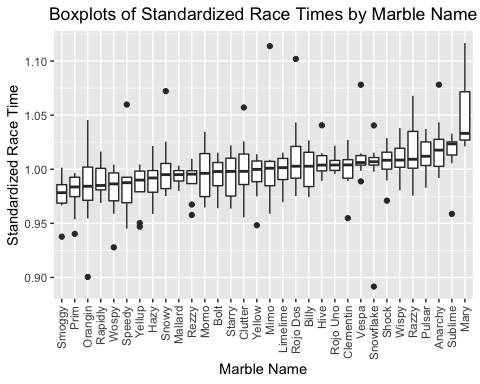
#Merge back  
marbles\_new <- left\_join(marbles\_cleaned, new\_data)

## Joining, by = "marble\_name"

glimpse(marbles\_new)

## Rows: 253  
## Columns: 11  
## $ race <chr> "S1Q1", "S1Q1", "S1Q1", "S1Q1", "S1Q1", "S1Q1…  
## $ site <chr> "Savage Speedway", "Savage Speedway", "Savage…  
## $ marble\_name <chr> "Clementin", "Starry", "Momo", "Yellow", "Sno…  
## $ team\_name <chr> "O'rangers", "Team Galactic", "Team Momo", "M…  
## $ time\_s <dbl> 28.11, 28.37, 28.40, 28.70, 28.71, 28.72, 28.…  
## $ track\_length\_m <fct> 12.81, 12.81, 12.81, 12.81, 12.81, 12.81, 12.…  
## $ number\_laps <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, …  
## $ avg\_time\_lap <fct> 28.11, 28.37, 28.4, 28.7, 28.71, 28.72, 28.96…  
## $ mean\_race\_time <dbl> 29.4400, 29.4400, 29.4400, 29.4400, 29.4400, …  
## $ standardized\_time <dbl> 0.9548234, 0.9636549, 0.9646739, 0.9748641, 0…  
## $ median\_standardized\_time <dbl> 1.0040201, 0.9980154, 0.9962279, 0.9998530, 0…

# boxplots of standardized times by marble  
ggplot(marbles\_new, aes(x=reorder(marble\_name, median\_standardized\_time ), y=standardized\_time)) + geom\_boxplot() + labs(title= "Boxplots of Standardized Race Times by Marble Name") + xlab("Marble Name") + ylab("Standardized Race Time") +  
theme(axis.text.x = element\_text(angle = 90, vjust = 0.5, hjust=1))



#ML workflow

set.seed(123)

# Put 3/4 of the data into the training set  
data\_split <- initial\_split(marbles\_new, prop = 3/4)

# Create data frames for the two sets:  
train\_data <- training(data\_split)  
test\_data <- testing(data\_split)

#CV  
folds <- vfold\_cv(train\_data, v = 5, repeats = 5, strata = site)  
folds

## # 5-fold cross-validation repeated 5 times using stratification   
## # A tibble: 25 × 3  
## splits id id2   
## <list> <chr> <chr>  
## 1 <split [147/42]> Repeat1 Fold1  
## 2 <split [148/41]> Repeat1 Fold2  
## 3 <split [151/38]> Repeat1 Fold3  
## 4 <split [155/34]> Repeat1 Fold4  
## 5 <split [155/34]> Repeat1 Fold5  
## 6 <split [147/42]> Repeat2 Fold1  
## 7 <split [148/41]> Repeat2 Fold2  
## 8 <split [151/38]> Repeat2 Fold3  
## 9 <split [155/34]> Repeat2 Fold4  
## 10 <split [155/34]> Repeat2 Fold5  
## # … with 15 more rows

#Make recipe  
data\_rec <- recipe(time\_s ~ ., data = train\_data) %>% step\_dummy(all\_nominal())

#Model 1: Decision Tree Model #run parallels to determine number of cores

cores <- parallel::detectCores() - 1  
cores

## [1] 7

cl <- makeCluster(cores)  
  
registerDoParallel(cl)  
  
  
#define the tree model  
tune\_spec <-   
 decision\_tree(  
 cost\_complexity = tune(),  
 tree\_depth = tune()  
 ) %>%   
 set\_engine("rpart") %>%   
 set\_mode("regression")  
  
tune\_spec

## Decision Tree Model Specification (regression)  
##   
## Main Arguments:  
## cost\_complexity = tune()  
## tree\_depth = tune()  
##   
## Computational engine: rpart

#define workflow for tree  
tree\_wflow <- workflow() %>%  
 add\_model(tune\_spec) %>%  
 add\_recipe(data\_rec)

#tuning grid specification  
tree\_grid <- grid\_regular(cost\_complexity(),  
 tree\_depth(),  
 levels = 5)  
  
tree\_grid

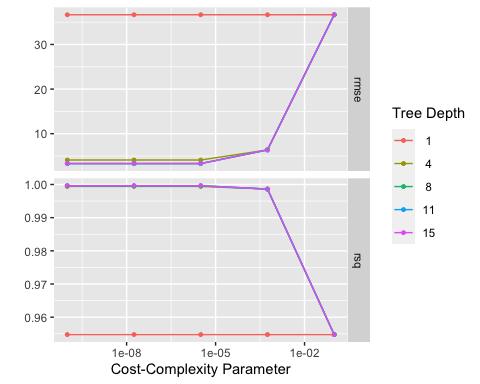
## # A tibble: 25 × 2  
## cost\_complexity tree\_depth  
## <dbl> <int>  
## 1 0.0000000001 1  
## 2 0.0000000178 1  
## 3 0.00000316 1  
## 4 0.000562 1  
## 5 0.1 1  
## 6 0.0000000001 4  
## 7 0.0000000178 4  
## 8 0.00000316 4  
## 9 0.000562 4  
## 10 0.1 4  
## # … with 15 more rows

#tree depth  
tree\_grid %>%  
 count(tree\_depth)

## # A tibble: 5 × 2  
## tree\_depth n  
## <int> <int>  
## 1 1 5  
## 2 4 5  
## 3 8 5  
## 4 11 5  
## 5 15 5

tree\_res <- tree\_wflow %>%  
 tune\_grid(  
 resamples = folds,  
 grid = tree\_grid  
 )

#default visualization  
tree\_res %>% autoplot()



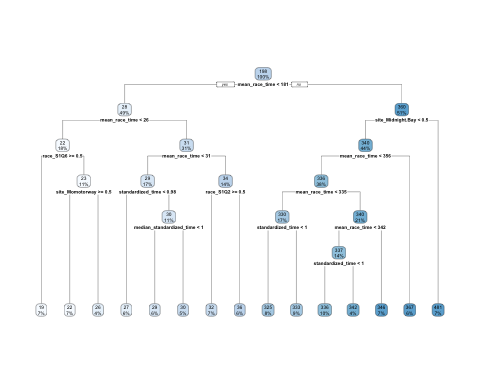
#Get the tuned model that performs best   
best\_tree <- tree\_res %>%   
 select\_best(metric = "rmse")  
  
#Finalize workflow with best model  
best\_tree\_wf <- tree\_wflow %>%   
 finalize\_workflow(best\_tree)

#Fit final model to training data and evaluates finalized model on the testing data  
best\_tree\_fit <- best\_tree\_wf %>%  
 last\_fit(data\_split)  
  
#On training data  
best\_fit <- best\_tree\_wf %>%   
 fit(data = train\_data)  
  
best\_pred <- predict(best\_fit, train\_data)  
  
best\_tree\_fit %>%  
 collect\_metrics()

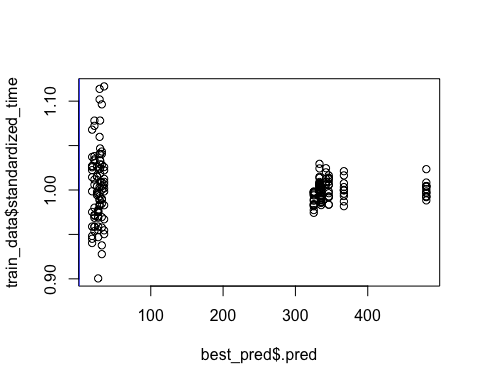
## # A tibble: 2 × 4  
## .metric .estimator .estimate .config   
## <chr> <chr> <dbl> <chr>   
## 1 rmse standard 2.56 Preprocessor1\_Model1  
## 2 rsq standard 1.00 Preprocessor1\_Model1

#Plot final tree  
rpart.plot(extract\_fit\_parsnip(best\_tree\_fit)$fit)

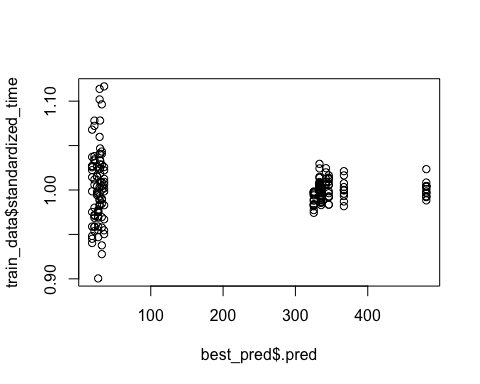
## Warning: Cannot retrieve the data used to build the model (model.frame: object '..y' not found).  
## To silence this warning:  
## Call rpart.plot with roundint=FALSE,  
## or rebuild the rpart model with model=TRUE.



#Predicted versus observed  
plot(best\_pred$.pred,train\_data$standardized\_time)  
abline(a=0,b=1, col = 'blue')



#Residuals  
plot(best\_pred$.pred,train\_data$standardized\_time)  
abline(a=0,b=0, col = 'green')



#I ’ll do the same with LASSO and Random Forest Model and compare