Starcraft Skill Data - Exploration

Tree Model Selection

Sergei Dakov

2023-07-19

Preparing The Data

First, load in the libraries used in model selection:

```
library(tidyverse)
library(rsample)
library(parsnip)
library(recipes)
library(themis)
library(glmnet)
library(ggplot2)
library(ggmosaic)
library(yardstick)
```

Second, load in some helper functions:

```
source("model_selection_skeleton.r")
```

Next, load in the data.

Some changes need to be made to be data so the model selection process works, such as converting the LeagueIndex column to a factor, and removing the age variable (as it is not a measurable metric by the game).

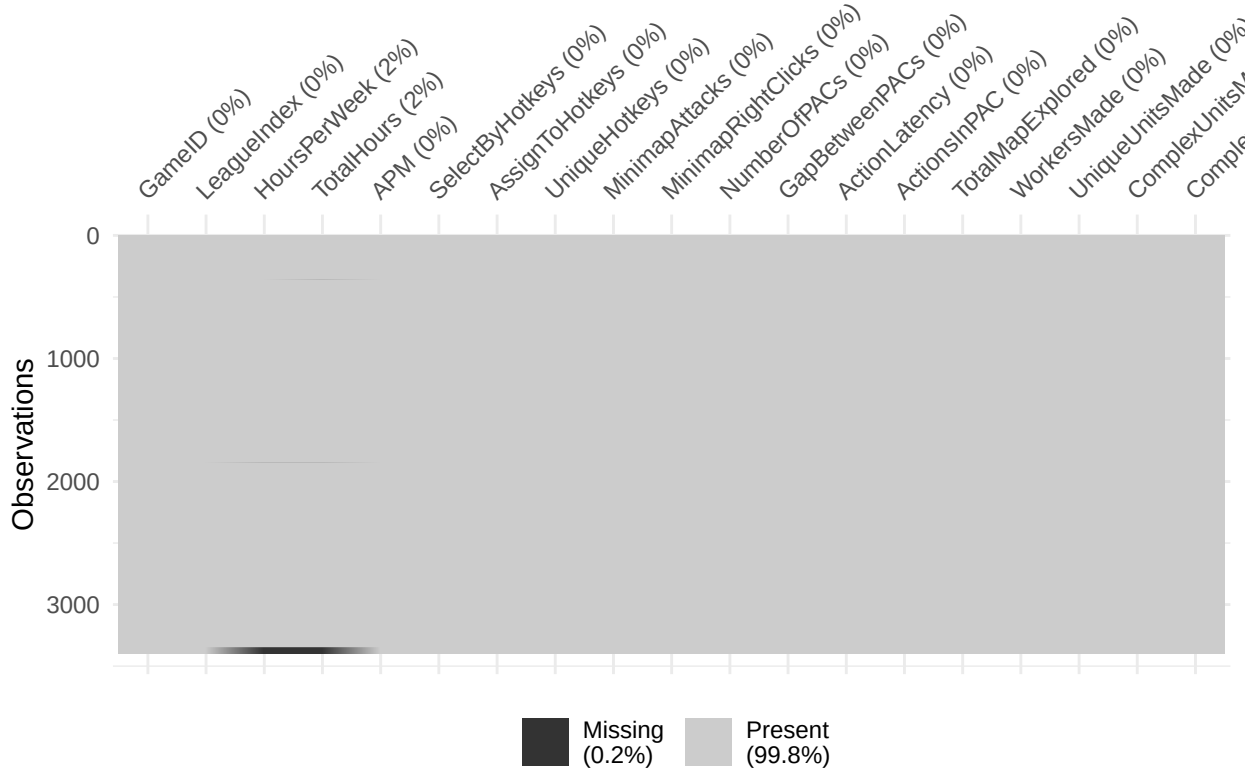
```
skillData <- read_csv("SkillCraft1_Dataset.csv") %>%
  mutate(LeagueIndex = factor(LeagueIndex)) %>%
  select(-Age) %>%
  mutate(across(c("HoursPerWeek","TotalHours"),~as.numeric(.x)))
```

```
## Rows: 3395 Columns: 20
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (3): Age, HoursPerWeek, TotalHours
## dbl (17): GameID, LeagueIndex, APM, SelectByHotkeys, AssignToHotkeys, Unique...
##
## 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.
```

```
## Warning: There were 2 warnings in ‘mutate()‘.
## The first warning was:
## i In argument: ‘across(c("HoursPerWeek", "TotalHours"), ~as.numeric(.x))‘.
## Caused by warning:
## ! NAs introduced by coercion
## i Run ‘dplyr::last_dplyr_warnings()‘ to see the 1 remaining warning.
```

Visualize missing data:

```
library(naniar)
vis_miss(skillData)
```



Define the initial model:

```
mod_tree <- decision_tree(mode="classification",cost_complexity = 0.01)
```

Perform the initial split:

```
set.seed(1234)
reg_split <- initial_split(skillData,strata = "LeagueIndex")
regression_train <- training(reg_split)
regression_test <- testing(reg_split)
```

As a baseline we will fit the data using the initial model, to assess the efficiency of the optimization process.

```
fit_tree <- mod_tree %>% fit(LeagueIndex~.,regression_train)
pred_tree <- predict(fit_tree,regression_test,type="prob")
pred_tree <- pred_tree %>% cbind(truth = regression_test$LeagueIndex)
roc_auc(pred_tree,"truth",starts_with(".pred"))
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc hand_till      0.831
```

```
guess_tree <- predict(fit_tree,regression_test)
pred_tree <- pred_tree %>% cbind(guess = guess_tree$.pred_class)
accuracy(pred_tree,"truth","guess")
```

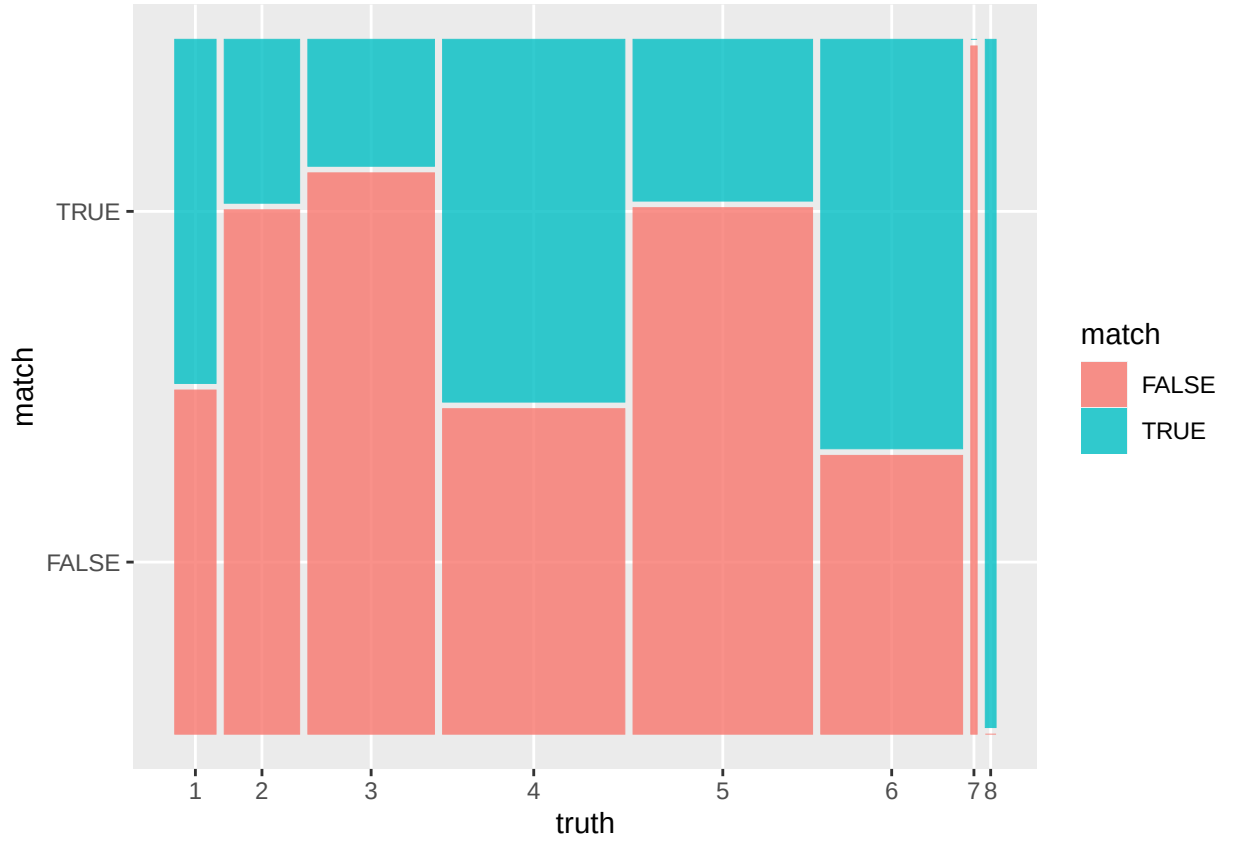
```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.387
```

In the basic scenario the result of the model is an AUC is 0.831, which is slightly worse than the logistic regression model.

The accuracy is 0.387, also somewhat lower than the regression model.

```
pred_tree <- pred_tree %>% mutate(match = (truth==guess))
pred_tree %>% ggplot() + geom_mosaic(aes(x=product(match,truth),fill=match))
```

```
## Warning: `unite_()` was deprecated in tidyr 1.2.0.
## i Please use `unite()` instead.
## i The deprecated feature was likely used in the ggmosaic package.
##   Please report the issue at <https://github.com/haleyjeppson/ggmosaic>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



In this model case the tree model successfully identifies the 8th class (pro players) butt struggles more with some of the middle classes, a major difference being the prediction quality in class 5.

Model Selection

First, preform the splits to independently optimize every element of the model, We select the splits to favor handling component ananlysis followed by class imbalance.

```
split_sizes <- c("imbalance"=900,"tuning"=644,"components"=1000)


set.seed(123)
imbalance_split <- initial_split(regression_train,strata="LeagueIndex",prop = ((split_sizes["imbalance"]
train_imbalance <- training(imbalance_split)
rest_split <- testing(imbalance_split)

set.seed(234)
tuning_split <- initial_split(rest_split,strata="LeagueIndex",prop=((split_sizes["tuning"])/nrow(rest_sp
train_tuning <- training(tuning_split)
train_components <- testing(tuning_split)
```

Since trees do not suffer from the presence of missing values, we do not have to impute any data, but since trees might be weak to class imbalance we will test to see if re-sampling of the data can bring out effects drowned out due to sparsity.

```
rec_nothing <- recipe(LeagueIndex~.,data=train_imbalance) %>%
  step_rm(GameID)

rec_upsample <- recipe(LeagueIndex~.,data=train_imbalance) %>%
  step_rm(GameID) %>%
  step_upsample(LeagueIndex,over_ratio=1,seed=111)

rec_downsample <- recipe(LeagueIndex~.,data=train_imbalance) %>%
  step_rm(GameID) %>%
  step_downsample(LeagueIndex,under_ratio = 1,seed=111)

rec_smote <- recipe(LeagueIndex~.,data=train_imbalance) %>%
  step_rm(GameID) %>%
  step_impute_mean(all_predictors()) %>%
  step_downsample(LeagueIndex,under_ratio = 1.5,seed = 111)%>%
  step_smotenc(LeagueIndex,over_ratio = 1,seed=111)

rec_puresmote <- recipe(LeagueIndex~.,data=train_imbalance) %>%
  step_rm(GameID) %>%
  step_impute_mean(all_predictors()) %>%
  step_smotenc(LeagueIndex,over_ratio = 1,seed =111)

set.seed(100)
cv_splits <- vfold_cv(train_imbalance,v=5,strata = 'LeagueIndex')


lst_recs <- list("nothing" = rec_nothing,
                 "upsample" = rec_upsample,
                 "downsample" = rec_downsample,
                 "smote" = rec_smote,
                 "only_smote" = rec_puresmote
)

cv_splits <- calculate_splits(cv_splits,lst_recs,mod_tree)
```

```
## Warning: No observations were detected in `truth` for level(s): '7'
## Computation will proceed by ignoring those levels.
## No observations were detected in `truth` for level(s): '7'
## Computation will proceed by ignoring those levels.
## No observations were detected in `truth` for level(s): '7'
## Computation will proceed by ignoring those levels.
## No observations were detected in `truth` for level(s): '7'
## Computation will proceed by ignoring those levels.
## No observations were detected in `truth` for level(s): '7'
## Computation will proceed by ignoring those levels.
```

```
cv_res_imbalance <- cv_splits %>% pivot_longer(cols=c(names(lst_recs)),names_to = "recipe",values_to =
  select(id,recipe,AUC) %>%
  group_by(recipe) %>%
  summarise (AUC = mean(AUC)) %>% arrange(-AUC)

cv_res_imbalance
```

```
## # A tibble: 5 x 2
##   recipe       AUC
##   <chr>      <dbl>
## 1 only_smote 0.811
## 2 nothing    0.807
## 3 upsample   0.781
## 4 downsample 0.724
## 5 smote      0.720
```

The best result is using SMOTE to re-balance the classes, with an AUC of 0.81.

Tree models naturally consider interactions due to its tiered structure, but sometimes the model may over-fit due to the many features, we can lower this effect by lowering dimensionality.

```
library(tune)

set.seed(100)
cv_splits <- vfold_cv(train_components,v=5,strata = 'LeagueIndex')

components <- c(seq(5,10,by=1),18)
components <- cbind.data.frame("num_comp" = components)




recipe_components <- recipe(LeagueIndex~.,data=train_components) %>%
  step_rm(GameID) %>%
  step_impute_mean(all_predictors()) %>%
  step_smotenc(LeagueIndex,over_ratio = 1,seed=111) %>%
  step_pca(all_predictors(),num_comp = tune())

formula_res <- mod_tree %>% tune_grid(object = mod_tree,
                                                preprocessor = recipe_components,
                                                resamples = cv_splits,
                                                metrics = metric_set(roc_auc),
                                                grid = components
                                      )
```

```
## Warning: The '...' are not used in this function but one or more objects were
## passed: ''
```

```
## > A | warning: No observations were detected in 'truth' for level(s): '8'
##                Computation will proceed by ignoring those levels.
```

```
## There were issues with some computations   A: x1
```

```
## > B | warning: No observations were detected in 'truth' for level(s): '7'
##                Computation will proceed by ignoring those levels.
```

```
## There were issues with some computations   A: x1There were issues with some computations   A: x1   B
## There were issues with some computations   A: x1   B: x1
```

```
col_metric <- collect_metrics(formula_res)

col_metric %>% arrange(-mean) %>% head(10)
```

```
## # A tibble: 7 x 7
##   num_comp .metric .estimator  mean     n std_err .config
##      <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1        6 roc_auc hand_till  0.791     5  0.0151 Preprocessor2_Model1
## 2        5 roc_auc hand_till  0.788     5  0.0178 Preprocessor1_Model1
## 3        9 roc_auc hand_till  0.783     5  0.0164 Preprocessor5_Model1
## 4        7 roc_auc hand_till  0.783     5  0.0166 Preprocessor3_Model1
## 5        8 roc_auc hand_till  0.783     5  0.0165 Preprocessor4_Model1
## 6       10 roc_auc hand_till  0.782     5  0.0156 Preprocessor6_Model1
## 7       18 roc_auc hand_till  0.774     5  0.0120 Preprocessor7_Model1
```

```
lst_recs <- list("nothing" = rec_nothing)

cv_splits <- calculate_splits(cv_splits,lst_recs,mod_tree)
```

```
## Warning: No observations were detected in `truth` for level(s): '8'
## Computation will proceed by ignoring those levels.
```

```
## Warning: No observations were detected in `truth` for level(s): '7'
## Computation will proceed by ignoring those levels.
```

```
cv_res_components <- cv_splits %>% pivot_longer(cols=c(names(lst_recs)),names_to = "recipe",values_to =
  select(id,recipe,AUC) %>%
  group_by(recipe) %>%
  summarise (AUC = mean(AUC)) %>% arrange(-AUC)

cv_res_components
```

```
## # A tibble: 1 x 2
##   recipe    AUC
##   <chr>   <dbl>
## 1 nothing 0.763
```

Reducing the dimensions to 7 or 9 primary components has the best performance, keeping 7 components will make model fitting faster thus we choose it.

AUC of 0.776 compared to AUC of 0.756 when not performing PCA.

When attempting to perform smote: 5,6,7 components performed the same, but smote failed due to lack of observations of certain classes in some cases.

Tuning

We can tune the parameters of the tree, as well as parameters of pre-processing.

```
set.seed(100)
cv_splits <- vfold_cv(train_components,v=5,strata = 'LeagueIndex')

tune_values <- expand.grid("over_ratio"=seq(0.5,1.5,by=0.1),"cost_complexity"=c(seq(0,1,by=0.1),2:5))


mod_tuning <- decision_tree(mode="classification",cost_complexity = tune())

recipe_tuning <- recipe(LeagueIndex~.,data=train_components) %>%
  step_rm(GameID) %>%
  step_impute_mean(all_predictors()) %>%
  step_smotenc(LeagueIndex,over_ratio = tune(),seed=111) %>%
  step_pca(all_predictors(),num_comp = 7)

formula_res <-tune_grid(object = mod_tuning,
                        preprocessor = recipe_tuning,
                        resamples = cv_splits,
                        metrics = metric_set(roc_auc),
                        grid = tune_values
                        )
```

```
## > A | warning: No observations were detected in `truth` for level(s): '8'
##                Computation will proceed by ignoring those levels.

## There were issues with some computations   A: x1

## > B | warning: No observations were detected in `truth` for level(s): '7'
##                Computation will proceed by ignoring those levels.

## There were issues with some computations   A: x1There were issues with some computations   A: x1   B
## There were issues with some computations   A: x1   B: x1
```

```
col_metric <- collect_metrics(formula_res)

col_metric %>% arrange(-mean) %>% head(10)
```

```
## # A tibble: 10 x 8
##    cost_complexity over_ratio .metric .estimator  mean     n std_err .config
##              <dbl>      <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
##  1             0          1   roc_auc hand_till  0.731     5  0.0209 Preprocess~
##  2             0          0.8 roc_auc hand_till  0.730     5  0.0299 Preprocess~
##  3             0.1        1.3 roc_auc hand_till  0.724     5  0.0126 Preprocess~
##  4             0          1.3 roc_auc hand_till  0.722     5  0.0171 Preprocess~
##  5             0          1.5 roc_auc hand_till  0.722     5  0.0206 Preprocess~
##  6             0          0.6 roc_auc hand_till  0.722     5  0.0189 Preprocess~
##  7             0.1        1.5 roc_auc hand_till  0.721     5  0.0140 Preprocess~
##  8             0          0.5 roc_auc hand_till  0.721     5  0.0218 Preprocess~
##  9             0.1        1.1 roc_auc hand_till  0.719     5  0.0150 Preprocess~
## 10             0          0.7 roc_auc hand_till  0.718     5  0.0222 Preprocess~
```

The best performing option is a non-penalized model with a ratio of 1:1 in SMOTE generation.

Final prediction.

```
recipe_final <- recipe(LeagueIndex~.,data=regression_train) %>%
  step_rm(GameID) %>%
  step_impute_mean(all_predictors()) %>%
  step_smotenc(LeagueIndex,over_ratio = 1,seed=111) %>%
  step_pca(all_predictors(),num_comp = 7) %>%
  prep()

train_final <- bake(recipe_final,new_data = NULL)
test_final <- bake(recipe_final,new_data = regression_test)

fit_final <- fit(mod_tree,LeagueIndex~.,train_final)
prob_final <- predict(fit_final,new_data = test_final,type = "prob")
pred_final <- predict(fit_final,new_data= test_final)

res_final <- prob_final %>% mutate(truth=test_final$LeagueIndex,guess = pred_final$.pred_class)
roc_auc(res_final,"truth",starts_with(".pred"))
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc hand_till      0.835
```

```
accuracy(res_final,"truth","guess")
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.317
```

Using the resulting model gives a worse prediction than the initial model, so we revert back to it. The model brings us to a final AUC of 0.835, and accuracy of 0.317.

Finally, save the model to access it from the main file:

```
saveRDS(recipe_final,"tree_model")
```