Starcraft Skill Data - Exploration

Multinomial Regression Model Selection

Sergei Dakov

2023-07-20

Preparation

First, load in required packages:

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.1     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.2     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(rsample)
library(parsnip)
library(recipes)
```

```
##
## Attaching package: 'recipes'
##
## The following object is masked from 'package:stringr':
##
##     fixed
##
## The following object is masked from 'package:stats':
##
##     step
```

```
library(themis)
library(glmnet)
```

```
## Loading required package: Matrix
##
```

```
## Attaching package: 'Matrix'
##
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
##
## Loaded glmnet 4.1-7
```

```
library(tune)
library(yardstick)
```

```
##
## Attaching package: 'yardstick'
##
## The following object is masked from 'package:readr':
##
##     spec
```

Next, load in some helper functions:

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

Finally, we load in the data, some of the data has missing values so we drop these columns (since the logistic regression model cannot handle those):

```
skillData <- read_csv("SkillCraft1_Dataset.csv") %>% select(-c('GameID','TotalHours','Age','HoursPerWee
```

```
## 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.
```

Split the data to produce a validation set:

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

Define the model to be used as baseline for the multinomial regression setting, and perform the initial split:

```
mod_multi <- multinom_reg(mode="classification",penalty = 0.01,engine = "glmnet")

fit_multi <- mod_multi %>% fit(LeagueIndex~.,data=regression_train)
```

```
predicted_guess <- predict(fit_multi,regression_test) %>% pull(.pred_class) #%>% as.numeric(.) %>% roun
```

Let's look at the AUC and accuracy for the initial prediction

```
comp_trial <- cbind.data.frame(truth = regression_test$LeagueIndex,prediction = predicted_guess)
comp_trial <- comp_trial %>% mutate(match = truth==prediction)
mean(comp_trial$match)
```

```
## [1] 0.4230317
```

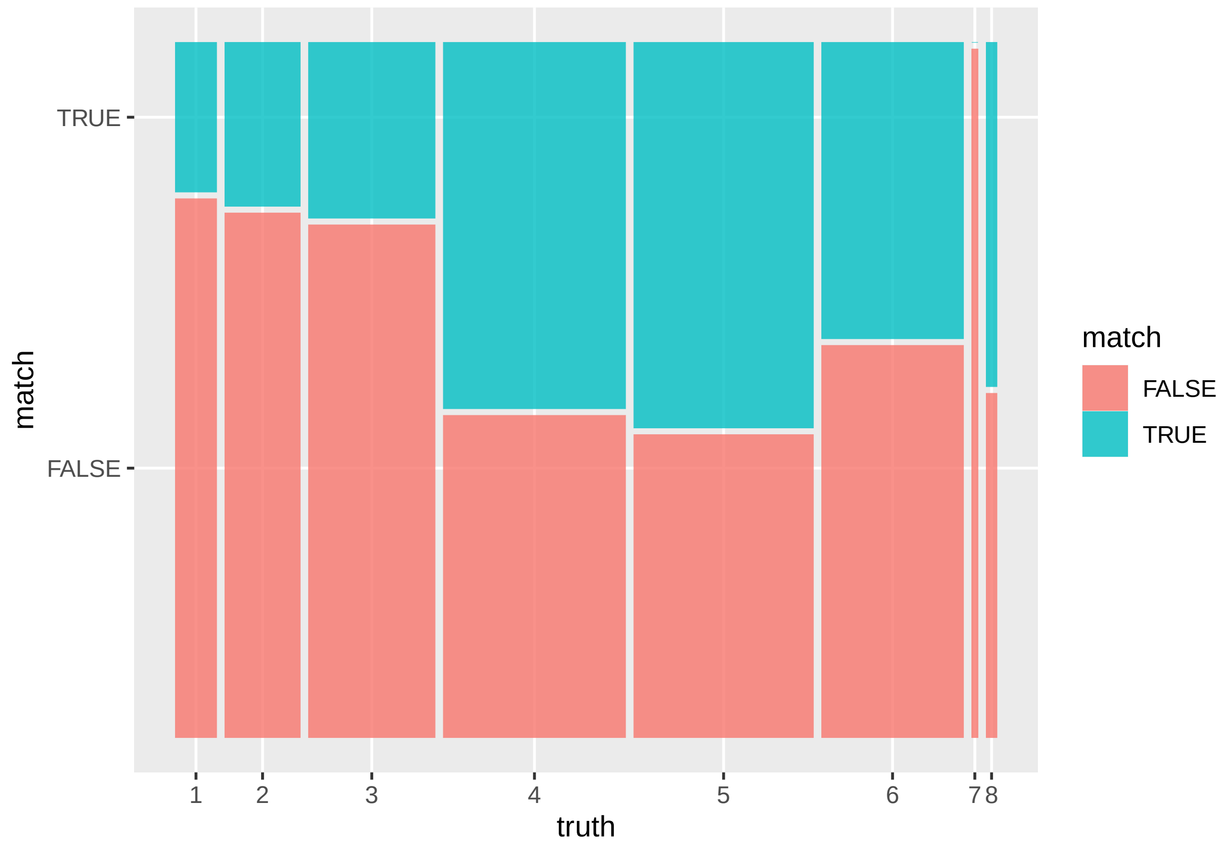
```
OnevRest(comp_trial,"truth","prediction")
```

```
## Warning: Using an external vector in selections was deprecated in tidyselect 1.1.0.
## i Please use `all_of()` or `any_of()` instead.
##   # Was:
##   data %>% select(guess)
##
##   # Now:
##   data %>% select(all_of(guess))
##
## See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## [1] 0.6252136
```

```
library(ggplot2)
library(ggmosaic)
comp_trial %>% 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.
```



For the AUC, the result for this model is 0.625, meaning the model struggles somewhat distinguishing between classes. The accuracy is 0.423, which while not very high is at least an improvement over the naive predictions.

Model Optimization

To test the validity of the arbitrary decisions made in the benchmark model, we load in the data again (since one of those arbitrary decisions was dropping some columns).

```
skillData <- read_csv("SkillCraft1_Dataset.csv") %>% mutate(LeagueIndex = factor(LeagueIndex)) %>% sele
```

```
## 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.
```

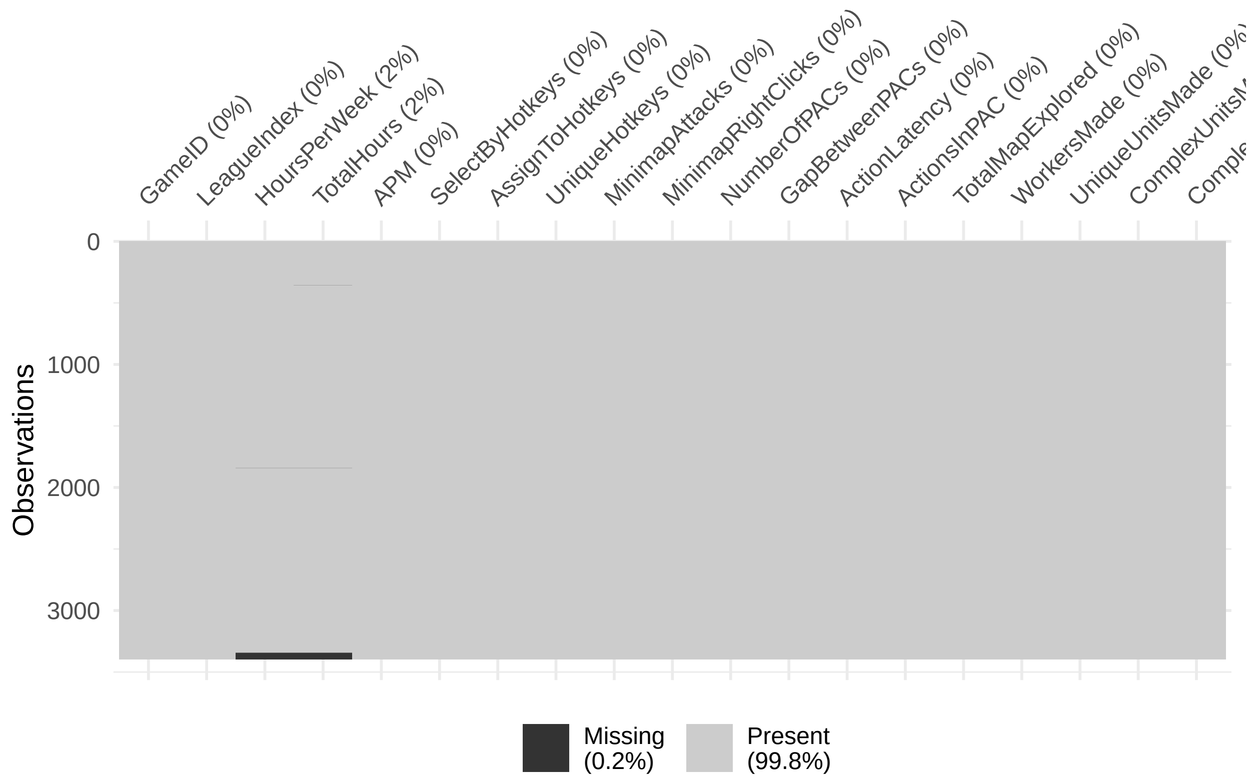
```
glimpse(skillData)
```

```
## Rows: 3,395
## Columns: 19
## $ GameID               <dbl> 52, 55, 56, 57, 58, 60, 61, 72, 77, 81, 83, 93, 9~
## $ LeagueIndex          <fct> 5, 5, 4, 3, 3, 2, 1, 7, 4, 4, 3, 4, 3, 3, 5, 5, 4~
## $ HoursPerWeek         <dbl> 10, 10, 10, 20, 10, 6, 8, 42, 14, 24, 16, 4, 12, ~
## $ TotalHours           <dbl> 3000, 5000, 200, 400, 500, 70, 240, 10000, 2708, ~
## $ APM                  <dbl> 143.7180, 129.2322, 69.9612, 107.6016, 122.8908, ~
## $ SelectByHotkeys      <dbl> 0.0035151591, 0.0033038124, 0.0011010906, 0.00103~
## $ AssignToHotkeys      <dbl> 2.196974e-04, 2.594617e-04, 3.355705e-04, 2.13101~
## $ UniqueHotkeys        <dbl> 7, 4, 4, 1, 2, 2, 6, 6, 2, 8, 4, 3, 1, 2, 2, 4, 1~
## $ MinimapAttacks       <dbl> 1.098487e-04, 2.940566e-04, 2.936242e-04, 5.32753~
## $ MinimapRightClicks   <dbl> 3.923169e-04, 4.324362e-04, 4.614094e-04, 5.43408~
## $ NumberOfPACs         <dbl> 0.004849036, 0.004307064, 0.002925755, 0.00378255~
## $ GapBetweenPACs       <dbl> 32.6677, 32.9194, 44.6475, 29.2203, 22.6885, 76.4~
## $ ActionLatency        <dbl> 40.8673, 42.3454, 75.3548, 53.7352, 62.0813, 98.7~
## $ ActionsInPAC         <dbl> 4.7508, 4.8434, 4.0430, 4.9155, 9.3740, 3.0965, 4~
## $ TotalMapExplored     <dbl> 28, 22, 22, 19, 15, 16, 15, 45, 29, 27, 24, 19, 1~
## $ WorkersMade          <dbl> 0.00139660, 0.00119350, 0.00074455, 0.00042620, 0~
## $ UniqueUnitsMade      <dbl> 6, 5, 6, 7, 4, 6, 5, 9, 7, 6, 7, 7, 7, 5, 7, 6, 6~
## $ ComplexUnitsMade     <dbl> 0.000000e+00, 0.000000e+00, 0.000000e+00, 0.00000~
## $ ComplexAbilitiesUsed <dbl> 0.0000e+00, 2.0757e-04, 1.8876e-04, 3.8358e-04, 1~
```

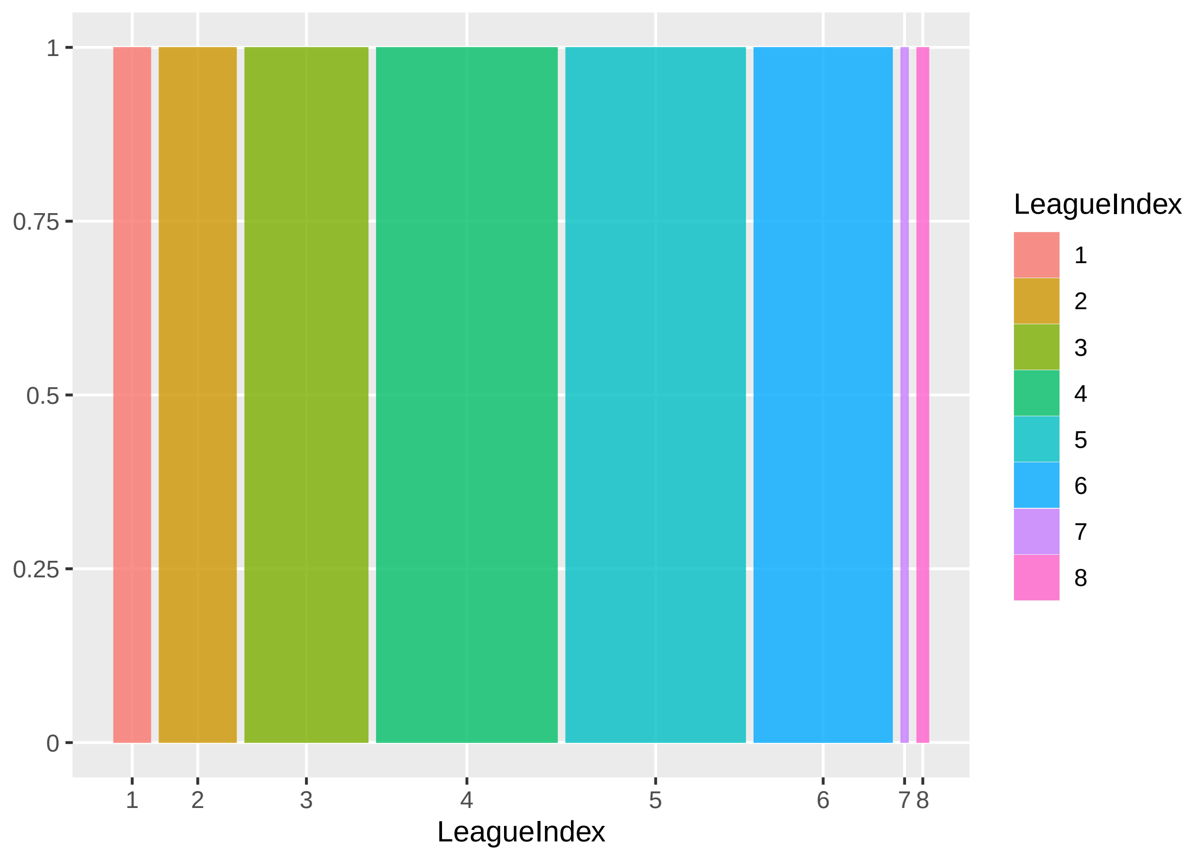
We can see the data has some missing values, and is quite unbalanced.

These are things we will need to account for in model selection.

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



```
skillData %>% ggplot() + geom_mosaic(aes(x=product(LeagueIndex),fill=LeagueIndex))
```



Perform the initial split again, we maintain the same seed so this split is identical to the previous one. This ensures the comparison is as fair as possible.

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

Conveniently all the data is numeric, so the main steps of model optimization are:

- dealing with NA
- rebalancing the data
- interactions
- tuning

Dealing with interactions might require the most data points, while tuning is less significant.

Missing values are somewhat rare but might have a significant effect (exploration of the data reveals most missing data comes from one specific class)

```
split_sizes <- c("imbalance"=600,"interactions"=800,"tuning"=500,"missing"=645)


set.seed(1234)
```

```
miss_split <- initial_split(regression_train,strata = "LeagueIndex",prop = (split_sizes["missing"]+2)/n
train_missing <- training(miss_split)
rest_split <- testing(miss_split)

set.seed(1234)
interaction_split <- initial_split(rest_split,strata="LeagueIndex",prop = ((split_sizes["interactions"]
train_interaction <- training(interaction_split)
rest_split <- testing(interaction_split)

set.seed(1234)
imbalance_split <- initial_split(rest_split,strata="LeagueIndex",prop = ((split_sizes["imbalance"]+1)/n
train_imbalance <- training(imbalance_split)
train_tuning <- testing(imbalance_split)
```

Imbalance:

We will consider two possible approaches, imputing the missing date and noting in a separate column if the row was missing.

We will do it the following ways:

- mean imputation
- KNN imputation
- dropping the columns with missing values

for the missing column the two options we will check is adding the columns or not.

This in total gives us 6 models to compare at this stage, (doing nothing is not a viable course of action due to the fact a logistic regression model cannot handle missing values).

```
rec_mean_nothing <- recipe(LeagueIndex~.,data=train_missing) %>%
  step_rm(GameID) %>%
  step_impute_mean(all_numeric()) %>%
  step_normalize(all_numeric_predictors())

rec_mean_extra <- recipe(LeagueIndex~.,data=train_missing) %>%
  step_rm(GameID) %>%
  step_mutate(missing_total=is.na(TotalHours)) %>%
  step_mutate(missing_weekly=is.na(HoursPerWeek)) %>%
  step_impute_mean(all_numeric()) %>%
  step_normalize(all_numeric_predictors())

rec_knn_nothing <- recipe(LeagueIndex~.,data=train_missing) %>%
  step_rm(GameID) %>%
  step_impute_knn(all_numeric()) %>%
  step_normalize(all_numeric_predictors())

rec_knn_extra <- recipe(LeagueIndex~.,data=train_missing) %>%
  step_rm(GameID) %>%
  step_mutate(missing_total=is.na(TotalHours)) %>%
  step_mutate(missing_weekly=is.na(HoursPerWeek)) %>%
  step_impute_knn(all_numeric()) %>%
  step_normalize(all_numeric_predictors())

rec_drop_nothing <- recipe(LeagueIndex~.,data=train_missing) %>%
  step_rm(GameID,HoursPerWeek,TotalHours) %>%
  step_normalize(all_numeric_predictors())

rec_drop_extra <- recipe(LeagueIndex~.,data=train_missing) %>%
  step_rm(GameID) %>%
  step_mutate(missing_total=is.na(TotalHours)) %>%
  step_mutate(missing_weekly=is.na(HoursPerWeek)) %>%
  step_rm(HoursPerWeek,TotalHours) %>%
  step_normalize(all_numeric_predictors())
```

We can now compare the performance of the different approaches by using cross-validation.

since the data is somewhat thin, we will only do a 4-fold split as we do not want to stretch the data too thin (especially considering how sparse some of the classes are).

```
set.seed(100)
cv_splits <- vfold_cv(train_missing,v=4,strata = 'LeagueIndex')


lst_recs <- list("mean_not" = rec_mean_nothing,
                 "knn_not" = rec_knn_nothing,
                 "mean_extra" = rec_mean_extra,
                 "knn_extra" = rec_knn_extra,
                 "drop_not" = rec_drop_nothing,
                 "drop_extra" = rec_drop_extra)

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

```
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground
```

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

```
## `summarise()` has grouped output by 'missing'. You can override using the
## `.groups` argument.
```

```
cv_res_missing
```

```
## # A tibble: 6 x 3
## # Groups:   missing [3]
##   missing column   AUC
##   <chr>   <chr>  <dbl>
## 1 mean    extra  0.817
## 2 knn     extra  0.815
## 3 drop    extra  0.810
## 4 mean    not    0.792
## 5 knn     not    0.790
## 6 drop    not    0.780
```

The best performing option is to impute the mean while noting which values were missing as a separate variable, leading with a ROC AUC of 0.817.

Imbalance

To re-balance the data and deal with some of the sparsity we can resample the data.

another option worth considering is SMOTE, which generates new synthetic values based on already existing values, but unfortunately the sparseness of some classes does not allow us to use this method.

```
regression_train  %>% group_by(LeagueIndex) %>% summarise(n=n(),ratio = n()/nrow(regression_train))
```

```
## # A tibble: 8 x 3
##   LeagueIndex     n  ratio
##   <fct>       <int>  <dbl>
## 1 1             121 0.0476
## 2 2             263 0.103
## 3 3             412 0.162
## 4 4             608 0.239
## 5 5             606 0.238
## 6 6             463 0.182
## 7 7              28 0.0110
## 8 8              43 0.0169
```

```
rec_nothing <- recipe(LeagueIndex~.,data=train_imbalance) %>%
  step_rm(GameID) %>%
  step_mutate(missing_total=is.na(TotalHours)) %>%
  step_mutate(missing_weekly=is.na(HoursPerWeek)) %>%
  step_impute_mean(all_numeric()) %>%
  step_normalize(all_numeric_predictors())

rec_upsample <- recipe(LeagueIndex~.,data=train_imbalance) %>%
  step_rm(GameID) %>%
  step_mutate(missing_total=is.na(TotalHours)) %>%
  step_mutate(missing_weekly=is.na(HoursPerWeek)) %>%
  step_impute_mean(all_numeric()) %>%
  step_upsample(LeagueIndex,over_ratio=1,seed=123) %>%
  step_normalize(all_numeric_predictors())


rec_downsample <- recipe(LeagueIndex~.,data=train_imbalance) %>%
  step_rm(GameID) %>%
  step_mutate(missing_total=is.na(TotalHours)) %>%
  step_mutate(missing_weekly=is.na(HoursPerWeek)) %>%
  step_impute_mean(all_numeric()) %>%
  step_downsample(LeagueIndex,under_ratio=1,seed=234) %>%
  step_normalize(all_numeric_predictors())
```

Calculate the AUC for each approach:

```
set.seed(100)
cv_splits <- vfold_cv(train_imbalance,v=4,strata = 'LeagueIndex')


lst_recs <- list("nothing" = rec_nothing,
                 "upsample" = rec_upsample,
                 "downsample" = rec_downsample
                 )

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

```
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## 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.

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning: 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,accuracy) %>%
  group_by(recipe) %>%
  summarise (accuracy = mean(accuracy)) %>% arrange(-accuracy)

cv_res_imbalance
```

```
## # A tibble: 3 x 2
##   recipe     accuracy
##   <chr>         <dbl>
## 1 upsample      0.823
## 2 nothing       0.816
## 3 downsample    0.758
```

Upsampling provides the best result, with an AUC of 0.823.

System warnings also inform us that the sparsity of the classes is indeed harming the model in non-upsampling modes.

Note that the data does not fulfill normality assumption, we can attempt to fix this by using a Yeo-Johnson transformation, and compare the results to confirm whether it is significant.

```
rec_nothing_yeo <- recipe(LeagueIndex~.,data=train_imbalance) %>%
  step_rm(GameID) %>%
  step_mutate(missing_total=is.na(TotalHours)) %>%
  step_mutate(missing_weekly=is.na(HoursPerWeek)) %>%
  step_YeoJohnson(all_numeric_predictors()) %>%
  step_impute_mean(all_numeric_predictors()) %>%
  step_normalize(all_numeric_predictors())

rec_downsample_yeo <- recipe(LeagueIndex~.,data=train_imbalance) %>%
  step_rm(GameID) %>%
  step_mutate(missing_total=is.na(TotalHours)) %>%
  step_mutate(missing_weekly=is.na(HoursPerWeek)) %>%
  step_YeoJohnson(all_numeric_predictors()) %>%
  step_impute_mean(all_numeric_predictors()) %>%
  step_normalize(all_numeric_predictors()) %>%
  step_downsample(LeagueIndex,under_ratio=1,seed=234)

rec_upsample_yeo <- recipe(LeagueIndex~.,data=train_imbalance) %>%
  step_rm(GameID) %>%
  step_mutate(missing_total=is.na(TotalHours)) %>%
  step_mutate(missing_weekly=is.na(HoursPerWeek)) %>%
  step_YeoJohnson(all_numeric_predictors()) %>%
  step_impute_mean(all_numeric_predictors()) %>%
```

```r
  step_upsample(LeagueIndex,over_ratio=1,seed=123) %>%
  step_normalize(all_numeric_predictors())

rec_smote_yeo <- recipe(LeagueIndex~.,data=train_imbalance) %>%
  step_rm(GameID) %>%
  step_mutate(missing_total=is.na(TotalHours)) %>%
  step_mutate(missing_weekly=is.na(HoursPerWeek)) %>%
  step_YeoJohnson(all_numeric_predictors()) %>%
  step_impute_mean(all_numeric_predictors()) %>%
  step_normalize(all_numeric_predictors()) %>%
  step_downsample(under_ratio= 1.5, seed = 11) %>%
  step_smotenc(over_ratio = 1, seed = 111)


set.seed(100)
cv_splits <- vfold_cv(train_imbalance,v=4,strata = 'LeagueIndex')


lst_recs <- list("nothing" = rec_nothing,
                 "nothing_yeo" = rec_nothing_yeo,
                 "downsample" = rec_downsample,
                 "upsample_yeo" = rec_upsample_yeo,
                 "downsample_yeo" = rec_downsample_yeo,
                 "upsample" = rec_upsample)

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

```
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

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

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground
## Warning: No observations were detected in ‘truth‘ for level(s): ’7’
## Computation will proceed by ignoring those levels.
```

```
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground
```

```
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground
```

```
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground
```

```
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground
```

```
## 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.
```

```
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground
```

```
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground
```

```
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground
```

```
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground
```

```
## 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.
```

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

cv_res_imbalance
```

```
## # A tibble: 6 x 2
##   recipe         accuracy
##   <chr>             <dbl>
## 1 upsample_yeo      0.839
## 2 upsample          0.823
## 3 nothing_yeo       0.819
## 4 nothing           0.816
## 5 downsample_yeo    0.766
## 6 downsample        0.758
```

When also accounting for Yeo Johnoson transformation, we can see that it does improve the model performance. The upsample method still is best performing (AUC of 0.839).

Interactions

The multinomial model does not inherently deal with interactions between predictors, since some of the parameters might be linked we can test whether there are any significant interactions we need to account for.

We will do it the following way:

- he no interaction model will be the base line to compare to
- some interactions seem naturally appealing, we can handpick some interactions and check only those (in this case we will use the interactions of APM with everything else)
- we can account for all interactions, then filter out all predictors which have too low of a variance as to not overwhelm the model (thus overfit the data)

```
rec_nothing <- recipe(LeagueIndex~.,data=train_interaction) %>%
  step_rm(GameID) %>%
  step_mutate(missing_total=is.na(TotalHours)) %>%
  step_mutate(missing_weekly=is.na(HoursPerWeek)) %>%
  step_YeoJohnson(all_numeric_predictors()) %>%
  step_impute_mean(all_numeric_predictors()) %>%
  step_normalize(all_numeric_predictors()) %>%
  step_upsample(LeagueIndex,over_ratio = 1,seed = 112)

rec_handpicked <- recipe(LeagueIndex~.,data=train_interaction) %>%
  step_rm(GameID) %>%
  step_mutate(missing_total=is.na(TotalHours)) %>%
  step_mutate(missing_weekly=is.na(HoursPerWeek)) %>%
  step_YeoJohnson(all_numeric_predictors()) %>%
  step_impute_mean(all_numeric_predictors()) %>%
  step_interact(~APM:all_numeric_predictors()) %>%
  step_nzv(all_numeric_predictors(),freq_cut = 99/1) %>%
  step_normalize(all_numeric_predictors())%>%
  step_upsample(LeagueIndex,over_ratio = 1,seed = 112)

rec_all_interact <- recipe(LeagueIndex~.,data=train_interaction) %>%
  step_rm(GameID)%>%
  step_mutate(missing_total=is.na(TotalHours)) %>%
  step_mutate(missing_weekly=is.na(HoursPerWeek)) %>%
  step_YeoJohnson(all_numeric_predictors()) %>%
  step_impute_mean(all_numeric_predictors()) %>%
  step_normalize(all_numeric_predictors()) %>%
  step_interact(~all_numeric_predictors():all_numeric_predictors()) %>%
  step_nzv(all_numeric_predictors(),freq_cut = 95/5) %>%
  step_upsample(LeagueIndex,over_ratio = 1,seed = 112)

rec_expand <- recipe(LeagueIndex~.,data=train_interaction) %>%
  step_impute_mean(all_numeric_predictors()) %>%
  step_upsample(LeagueIndex,over_ratio = 0.2,seed=112) %>%
  prep()

train_interaction_big <-rec_expand %>% bake(new_data=NULL)
```

```
set.seed(100)
cv_splits<- vfold_cv(train_interaction,v=4,strata = 'LeagueIndex')


lst_recs <- list("handpicked" = rec_handpicked,
                     "all" = rec_all_interact,
                     "nothing" = rec_nothing
                     )


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

```
## 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.
```

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

cv_res_interact
```

```
## # A tibble: 3 x 2
##   recipe       AUC
##   <chr>      <dbl>
## 1 nothing    0.857
## 2 handpicked 0.856
## 3 all        0.841
```

no interactions is best performing (AUC 0.857).

Tuning:

The hyper-parameters that we can further tune are:

- the complexity penalty for the model
- the mixture parameter for the elastic net
- the ratio up to which we are going to be upsamling the sparse classes to

Since we will be testing each combination of these hyper-parameters, this step roughly 850 different models, thus this step is noticeably slow and takes time to complete.

```
set.seed(100)
cv_splits <- vfold_cv(train_tuning,v=4,strata = 'LeagueIndex')

penalty <- seq(6,0,length=7)
mixture <- seq(0,1,by=0.1)
ratio <- seq(0.5,1.5,len=11)
tuning_vars <- expand.grid("penalty"=penalty
                           ,"mixture"=mixture
                           ,"over_ratio"=ratio
)


mod_multi_tuning <- multinom_reg(mode="classification",engine = "glmnet",penalty = tune(),mixture = tun

recipe_tuning <- recipe(LeagueIndex~.,data=train_tuning) %>%
  step_rm(GameID)%>%
  step_mutate(missing_total=as.numeric(is.na(TotalHours))) %>%
  step_mutate(missing_weekly=as.numeric(is.na(HoursPerWeek))) %>%
  step_YeoJohnson(all_numeric_predictors()) %>%
  step_impute_mean(all_numeric_predictors()) %>%
  step_normalize(all_numeric_predictors()) %>%
  step_upsample(LeagueIndex,over_ratio = tune(),seed = 112)

formula_res <- mod_multi_tuning %>% tune_grid(object = mod_multi_tuning,
                                              preprocessor = recipe_tuning,
                                              resamples = cv_splits,
                                              metrics = metric_set(roc_auc),
                                              grid = tuning_vars
)
```

```
## 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): '7'
##                Computation will proceed by ignoring those levels.

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

##
```

```
col_metric <- collect_metrics(formula_res)

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

```
## # A tibble: 10 x 9
##    penalty mixture over_ratio .metric .estimator  mean     n std_err .config
##      <dbl>   <dbl>      <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
##  1       0       0        1.5 roc_auc hand_till  0.833     4 0.00605 Preprocess~
##  2       0     0          1.4 roc_auc hand_till  0.833     4 0.00477 Preprocess~
##  3       0     0.1        1.5 roc_auc hand_till  0.829     4 0.00899 Preprocess~
##  4       0     0          0.8 roc_auc hand_till  0.828     4 0.00333 Preprocess~
##  5       0     0.2        1.5 roc_auc hand_till  0.828     4 0.00947 Preprocess~
##  6       0     0.3        1.5 roc_auc hand_till  0.828     4 0.00933 Preprocess~
##  7       0     0.4        1.5 roc_auc hand_till  0.828     4 0.00943 Preprocess~
##  8       0     0.5        1.5 roc_auc hand_till  0.828     4 0.00925 Preprocess~
##  9       0     0.1        0.8 roc_auc hand_till  0.828     4 0.00483 Preprocess~
## 10       0     0.2        0.8 roc_auc hand_till  0.827     4 0.00402 Preprocess~
```

The best results were - using an unpenalized model, with no mixture, and and oversampling ratio of 1.5, with an AUC of 0.833.

Final Prediction:

```
mod_multi_final <- multinom_reg(mode="classification",engine = "glmnet",penalty=0)

rec_final <- recipe(LeagueIndex~.,data=regression_train) %>%
  step_rm(GameID)%>%
  step_mutate(missing_total=as.numeric(is.na(TotalHours))) %>%
  step_mutate(missing_weekly=as.numeric(is.na(HoursPerWeek))) %>%
  step_YeoJohnson(all_numeric_predictors()) %>%
  step_impute_mean(all_numeric_predictors()) %>%
  step_normalize(all_numeric_predictors()) %>%
  #step_upsample(LeagueIndex,over_ratio = 1.5,seed = 112) %>%
  prep()

fit_final <- rec_final %>% bake(new_data=NULL)
fit_final <- fit(mod_multi_final,LeagueIndex~.,fit_final)
pred_final <- predict(fit_final,new_data=bake(rec_final,new_data=regression_test),type="prob")
pred_final <- pred_final %>% mutate(truth = regression_test$LeagueIndex)
pred_guess <- predict(fit_final,new_data=bake(rec_final,new_data=regression_test))
pred_final <- pred_final %>% mutate(estimate = pred_guess$.pred_class)
roc_auc(pred_final,"truth",starts_with(".pred"))
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc hand_till      0.875
```

```
accuracy(pred_final,"truth","estimate")
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.455
```

The final optimized model has an AUC of 0.878, and an accuracy of 0.397.

Of note is that the model selection process has not improved prediction quality by much, it is possible that due to the low ammount of data and sparse data the optimization process was mislead by poorly representative data.

If we ignore some of the guidance from the optimization process (specifically the upsampling) we get an AUC of 0.875, accuracy of 0.455.

```
saveRDS(rec_final,"multinomial_model")
```