

# Math158\_Semesterproject\_part3\_MLR

Kevin Loun & Tesfa Asmara

4/3/2022

## Introduction

The dataset for this project contains 10,000 League of Legends ranked matches from the North American region with 775 variables offered through the Riot Games API, provided on Kaggle [ @riot ][ @james\_2020 ]. Each match is pulled from players who rank Gold in the League system, a ranking system that matches players of a similar skill level to play with and against each other. Amongst North American players, the Gold skill level was the second most common tier, achieved by 27.7 percent of players, or approximately 49.86 million players when considered against Riot Games' player base of 180 million [ @statista\_2021 ][ @riot\_tweet ]. This dataset will be referred to as `lol10`.

For this part of the project, the following variables are of interest: `lane`, `deaths`, `gold_earned`, `player_level` and `total_damage_dealt` with the goal of predicting the amount of `kills` a player receives during a game. A figure including all the relevant variables and their description is attached at the end.

## Computational Model Building

Using computational model building, we developed two models containing some similar and different features with the goal of finding a model to best predict the number of `kills` players receive in a game. To achieve this we split the data into a training and test set with the training set containing 75% of the initial data and the test set containing 25%.

## Feature Engineering

Based on our knowledge of the dataset and variables we decided to create two recipes that would later develop into two models. In our first recipe we decided to add minimal features: removing zero variance predictors and converting categorical data into numerical binary variables. We choose these features so that we could convert our categorical variables into a format that could better contribute to our model and we remove zero variance predictors to remove any columns that could contain all zero values.

In our second recipe we included the same features as recipe one as well as added additional features. In our second recipe we choose to include an interaction variable between `b_summoner1_total_damage_dealt` and `b_summoner1_gold_earned` because the total amount of gold a player receives in game directly impacts the amount of damage they output due to increase the amount of items they are able to buy. We then chose to normalize and scale `b_summoner1_gold_earned` because of the vast difference in its range compared to other variables. Gold Earned ranged into the ten thousands while other variables ranged to 30, this could have given gold earned more weight in the model than other variables so it was scaled and normalized in order to avoid this.

## Cross Validation and Fitting Model

```
## # A tibble: 2 x 6
##   .metric .estimator mean      n std_err .config
##   <chr>   <chr>      <dbl> <int>   <dbl> <fct>
## 1 rmse    standard    2.36      5 0.0178 Preprocessor1_Model1
## 2 rsq      standard    0.709      5 0.00456 Preprocessor1_Model1

## # A tibble: 2 x 6
##   .metric .estimator mean      n std_err .config
##   <chr>   <chr>      <dbl> <int>   <dbl> <fct>
## 1 rmse    standard    2.35      5 0.0163 Preprocessor1_Model1
## 2 rsq      standard    0.711      5 0.00436 Preprocessor1_Model1
```

## Fitting Model to Test Data

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>      <dbl>
## 1 rsq      standard    0.706

## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>      <dbl>
## 1 rmse    standard    2.34
```

## Statistical Model Building