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`.

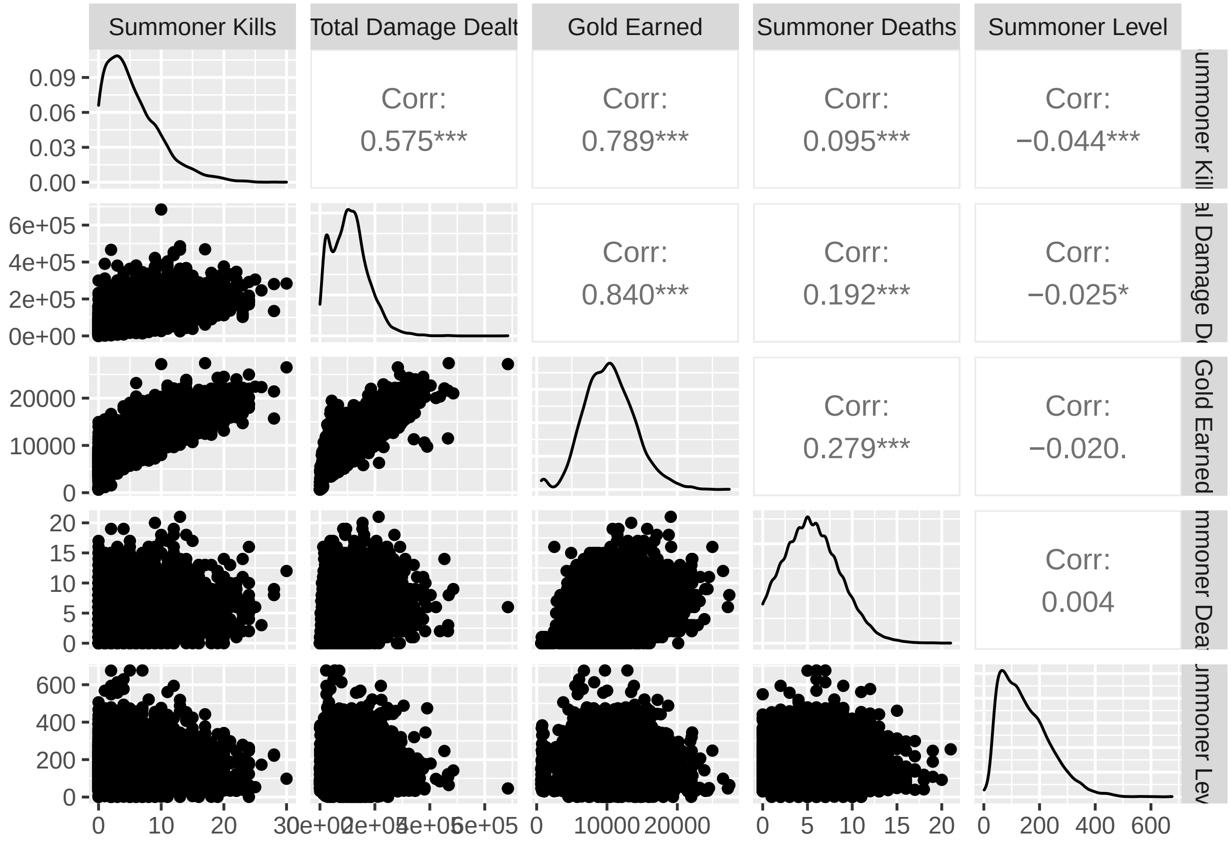
For this part of the project, the following variables are of interest: `lane`, `deaths`, `gold earned`, `player level` and `total damage dealt` with the goal of predicting the amount of `kills`a player recieves during a game. A figure including all the relevant variables and their description is attached 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 recieve 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%.

Looking for relationships

We decided to start by look for any strong relationships and correlations between our variables using a pairsplot. Upon generating a pairplot we notice strong correlations between `b_summoner1_total_damage_dealt` and `b_summoner1_gold_earned` since they have a correlation of 0.84. This indicates that a change in one of these variables will change and affect the value of the other variables, which indicates that we should create and interaction variable using these variables. Additionally, we notice that our explanatory variable `b_summoner1_kills` is highly correlated with `b_summoner1_gold_earned` meaning that this variable may have more weight than other varibles in the model. Using these observations we can begin to choose features to be added and develop two models to compare.



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 choos to include an interaction variable between `b_summoner1_total_damage_dealt` and `b_summoner1_gold_earned` because the total amount of gold a player recieves 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

After creating two models, we put the models through 5-fold cross validation to determine which model would provide better predictions for our test set. Our first model, `lol_fit_rs1` has a root mean squared error of 2.357 and a R^2 of 0.708 meaning that 70.8% of the variability in `b_summoner1_kills` can be explained by the training model. Our second model, `lol_fit_rs2` has a root mean squared error of 2.347 and a R^2 of 0.7110795 meaning that 71.1% of the variability in `b_summoner1_kills` can be explained by the training model.

Since our second model yielded a smaller RSME value and a larger R^2 value we decided to use this model to fit our testing data. It should be noted that the difference in RSME is small and that the difference in R^2 could be because our second model contains more predictor variables.

```
## # 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

Since our second model had a smaller RSME out of the two models, we chose to fit our testing data to the second model. Our model with the test data yielded a RMSE of 2.342259 and a R^2 of 0.7057904 meaning that 70.5% of the variability in `b_summoner1_kills` is explained by the model from the training data. One thing to note is that it appears that our RMSE for our training data is **lower** than our training data. This indicates that there is possible sampling bias in our 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