Ethan Cruz
Projects in ML, RPI
December 2022

Progress Report

The connected notebook has the current successful iteration of my champ selector. The initial project, of predicting which team turned out to be a simple task to complete. I decided to take it one step further and create a model which tries to predict which champion to pick next. This took more than just the initial logistic regression system.

The entire formula envelops four matrices. The first matrix, "freq" is a simple vector which each value v_i represents the total number of times champ v_i was selected divided by the total number of picks.

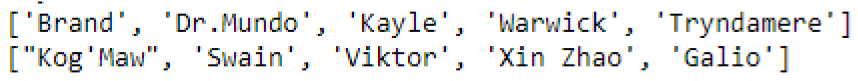
$$\begin{bmatrix} v_0 & v_1 & \dots & v_{147} \end{bmatrix}$$

The index is hardcoded to a champion's number. Unfortunately this means I have some massive functions which only function is to translate an index to a champ name. I then have two adjacency matrixes, "adjmat" and "enemat" whose jobs are to correlate a champion and their relations to teammates and enemies. adjmat rows represent champions and each column is that champion's percent chance to be teammates with the column index.

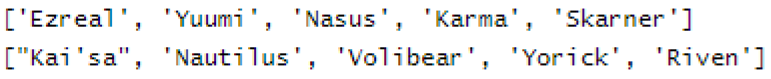
$$\begin{bmatrix} a_{0|0} & a_{1|0} & \dots & a_{147|0} \\ a_{0|1} & a_{1|1} & \dots & a_{147|1} \\ \dots & \dots & \dots & \dots \\ a_{0|147} & a_{1|147} & \dots & a_{147|147} \end{bmatrix}$$

a_{ij} is champion i's chance to be teammates with j. enemat is champion i's chance have j as an enemy.
The final factor is the logistic regression model "LogR" which takes into consideration teammates, enemies, and your next pick all at once.

The issue I came across when trying to do just LogR is that it kept choosing esoteric champions which have high winrate probability (what LogR is looking for). To sort-of regularize the initial picks I use the matrices to emphasize picking more popular champions (while still picking the champion which gives higher winrates). A team selection with and without the popularity matrices are shown below:



```
['Brand', 'Dr.Mundo', 'Kayle', 'Warwick', 'Tryndamere']
["Kog'Maw", 'Swain', 'Viktor', 'Xin Zhao', 'Galio']
```



```
['Ezreal', 'Yuumi', 'Nasus', 'Karma', 'Skarner']
["Kai'sa", 'Nautilus', 'Volibear', 'Yorick', 'Riven']
```

The first image's teams have no concept of roles. The selections have no correlation between each other. The second image has selections which actively counter champions from the enemy team and synergize well with teammates. The emphasis is shifted from just pure winrate calculations to a winrate-to-popularity model.
The best improvement from the pure LogR model is the ability to select champions other than the 10th (final) selection.

For the LogR model itself, I used the scikit's logistic regression model. I made sure to use regularization to slightly underfit the model. The data is taken from the Korea region's master, grandmaster, challenger games. Lots of team combinations never occur at a high level because it's just not winning at all. Because the data is binary classification, liblinear works quickly at solving for the weights, even if the dataset is of medium size.

Logistic regression was chosen over other methods because the chance of winning with a certain team is more chance than absolute. LogR does great at mapping statistic probability to a set of classification data.

In this case, wins/loss is the classification and the output of LogR is the chance of winning or losing. The next move is to implement an option to select from the top k best champion picks instead of just giving the number one. I also want to try using the data on a neural network to see what a network would believe is a good pick. NNs can also work well with percentage outputs so it's not all wishful thinking!