Project Final Report

for

League of Legends Rank Predictor

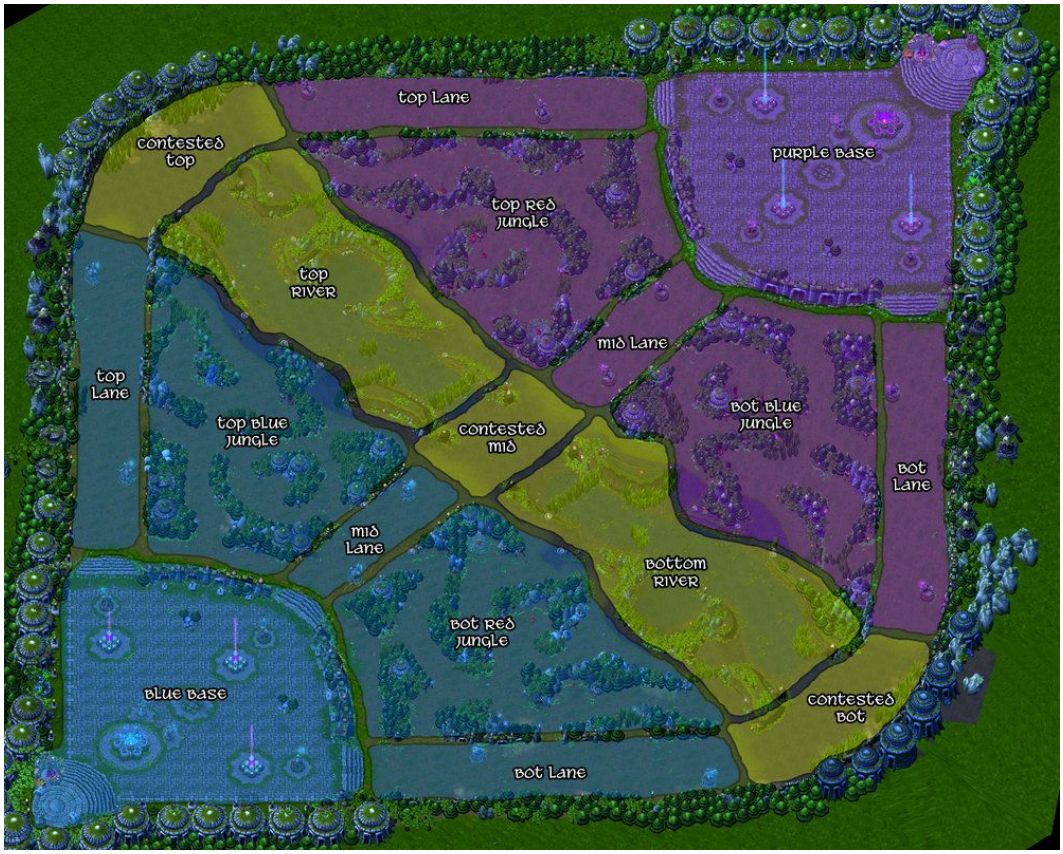
Prepared by
De Guzman, Nikko
Ho, Henry
MacBride, Koenrad
Rosas, Robert
Sarenas, Justin

November 26, 2017

1. Project Description, Goals, Data

1.1 Description of League of Legends

League of Legends (LoL) is a Multiplayer Online Battle Arena (MOBA) video game created by the company Riot Games. The game has 10 players split into two teams battling each other on Summoner's Rift which is the name of the map featured below. The main goal of each team is to get to the other player's base and destroy their nexus. Along the way, there are tower turrets that players will have to destroy before getting to the other team's base. There are creeps and jungle camps that spawn that players can kill which grants gold to buy items from the shop to boost their stats. They can also earn gold by killing players of the opposite team. There are five different roles that players will take which are Support, Attack Damage Carry (ADC), Jungler, Middle, and Top. The support and ADC take bot lane and all the other lanes take their respective places. Players will have to continuously kill creeps, jungle camps, and opposing players in order to push to and destroy the enemy nexus to win the game.



1.2 Description and Goals of Rank Predictor

In LoL, there is a ranked matchmaking system that players can participate in. There are 7 ranks in LoL which are Bronze, Silver, Gold, Platinum, Diamond, Master, Challenger. Players are split into these ranks according to their skill level which is most commonly referred to as ELO. Players can climb and drop ranks by winning and losing games.

The goal for our rank predictor is to predict the highest achieved rank of a player based on multiple in-game stats. The reason we chose highest achieved rank instead of current rank is because players in a low ranked match might actually have a higher skill level and rank than the average skill level and rank of the match.

1.3 Data

The data set we have is from the Riot Games Developer site. They provide users with seed data which contains 1,000 ranked solo-queue matches. The data is then divided into ten 100-match JSON files. In the JSON file, Riot provides a multitude of team statistics and player statistics. The rank predictor only cares about the player statistics listed at the end of the previous section. All this data is from Season 3 of LoL and the game is currently in the preseason of Season 8. In Season 3 ranks the ranks did not have Challenger tier. The way we decided to organize our data is we split off the data into 8 different features. The names of each of those features include: role, kills, deaths, assists, goldEarned, totalMinionsKilled, damageDealtToObjectives, totalDamageDealt, and highestAchivevedSeasonTier. The reason we chose these as features is because it was the most essential factors that would allow us to be able to predict the rank.

2. Methods and Algorithms

2.1 Train-Test Split

The data will be split into two different sets called the training data set and the testing data set. The training stage allows us to build a predictive model based on the training data set. The testing stage allows us to use the model from the training stage to predict the outcome on future data sets.

We will be splitting the data set into 30% or .3 for testing data and 70% or .7 for the training data.

2.2 K-fold Cross Validation

K-fold Cross Validation is used on supervised learning algorithms such as KNN, DT, Linear and Logistic Regression to solve the problems of overfitting of data model. The method partitions the data set into k amount of bins. A model is then trained on k-1 as training data. The model is then validated on the remaining k data. This method loops through all k bins and calculates the average. For example, our data set contains data for 10000 games undergoing 10-fold Cross Validation will have 10 bins with data from 1000 games in each bin. There will be 9 bins that will be used as the training set and 1 bin used as the testing set. The method will loop through all the bins until all the bins have been used as a testing set and will calculate the average. The benefits of this method is that it will not waste too much data however with big data, computations will become expensive.

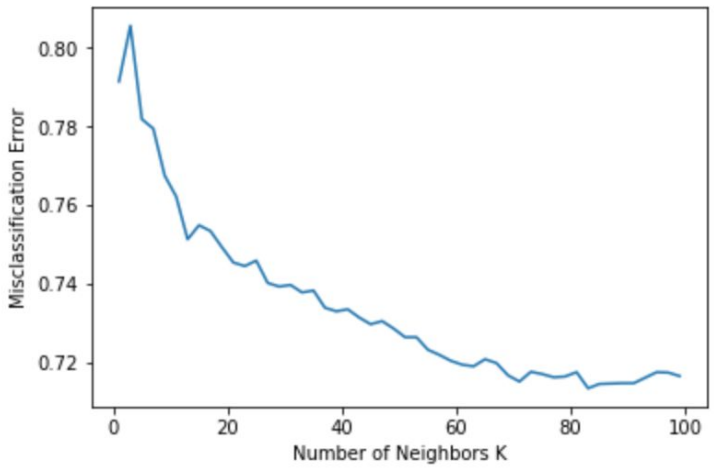
We will be using 10-fold Cross Validation for our data set.

2.3 K-Nearest Neighbor Classifier (KNN)

This algorithm classifies objects based on the closest training samples in the feature matrix. The k denotes the number of closest training samples the object will be compared to. The advantages of a larger k is that outliers will have little effect on the prediction while making the prediction more accurate. The advantages of a smaller k is that the computations are much simpler and the predictive model becomes simpler. For example, if we use 5-nearest neighbors then the testing object will be compared to the five nearest training objects. If our testing object is near 3 yellows and 2 greens, the prediction will be the testing object is also yellow. On the other hand if we use 1-nearest neighbor, the training object that is most closest to the object might not necessarily be the same as the testing object. The pros of this algorithm is that it is simple and easy to code with low computational requirements. The cons of this algorithm is that

it will become inaccurate with big data with multiple features and deciding how many objects the testing data compares to becomes challenging.

To calculate the most optimal K we ran cross-validation 100 times to see which K is best to use. We calculated that the most optimal K was 83.



2.4 Decision Tree Classifier (DT)

This algorithm builds a binary tree based based on the features provided in a data set. The algorithm calculates which features to prioritize at each node level by the minimum amount of entropy or maximum amount of information gained from the features. The nodes keep getting filled with features until there are no features left. The algorithm also calculates the threshold values where the node would split. The pros of using this algorithm is it is easily understandable for humans and can use both categorical and numerical data. The cons of this algorithm is it uses a trial and error/brute force method when calculating threshold values and the data model might be overfitted.

2.5 Linear Regression

This algorithm predicts the testing variable by fitting the best linear relationship to the training samples. It does this by minimizing the difference between the prediction value and the training value to find the best fit line for the training data.

2.6 Logistic Regression Classifier

This algorithms uses a logistic function to measure the relationship between a categorical dependent feature and independent features by estimating a probability.

2.7 Random Forest Classifier

This algorithm constructs multiple decision trees at training stage and combines all the results into a prediction. The features that the algorithm selects are all random so that the decision trees constructed would be different from each other. The advantages of using this classifier is that it resolves the problem of overfitting that the decision tree classifier has. This algorithm also runs well with big data with multiple features as well as data with missing values.

2.8 One-Hot Encoding

This method takes into account features that are categorical and converts them into a form that could be used by the algorithms above . For example, a feature such as color might have three states such as red, blue, or green. The method splits color into three different features which are red, blue, and green and assigns a binary value of 1 or 0 wherever the data contains that color. Assignment of numbers to categorical features are meaningless.

We will be using one-hot encoding with all the algorithms we used due to the roles feature being categorical.

3. Results

3.1 K-Nearest Neighbor Classifier (KNN)

After implementing KNN to our training and testing set we calculated the accuracy of our data to be .28500 or 28.5%.

3.2 Decision Tree Classifier (DT)

After implementing DT to our training and testing set we calculated the accuracy of our data to be .214 or 21.4%.

3.3 Linear Regression

After splitting the data and fitting the training data to our linear regression classifier, our intercept was 3.689201. After acquiring our coefficients we are presented with the following model for linear regression.

Predictive model will be:

Y = 3.689201
+ 8.43080737e-03 x **KILLS**
+ 1.04620479e-03 x **DEATHS**
+ 1.61815563e-03 x **ASSISTS**
+ -1.36772404e-05 x **GOLDEARNED**
+ -1.02733895e-04 x **TOTALMINIONSKILLED**
+ -8.54342063e-06 x **DAMAGEDEALTTOOBJECTIVES**
+ 1.38079362e-06 x **TOTALDAMAGEDEALT**
+ 1.81496225e-01 x **ROLEDUO**
+-6.62828634e-01 x **ROLEDUOCARRY**
+ -2.33479521e-02 x **ROLEDUOSUPPORT**
+-1.24761007e-02 x **ROLENONE**
+ 3.28955974e-03 x **ROLESOLO**

We calculated our RSME to be 2.097. To validate this we performed 10 fold cross-validation and the RSME mean is 2.081.

3.4 Logistic Regression Classifier

After implementing logistic regression to our split data, we calculated the accuracy score to be .294000 or 29.4%.

3.5 Random Forest Classifier

After implementing Random Forest to our training and testing set, we calculated the accuracy of our data to be .271667 or 27.1%.

4. Conclusion

The accuracy on the classifications we used were around the 20%-30% which is very low. We believe the problem lies with the machine learning having a bias towards the rank with the most number of occurrences in the data. It also may have learned the noise in the data which is also known as overfitting. After counting the total number of ranks in the data, we can see that rank Silver appears the most:

```
df = pd.DataFrame()
df = pd.get_dummies(label_vector, columns=['highestAchievedSeasonTier']).sum()
print(df)
```

```
BRONZE      1404
DIAMOND      293
GOLD        2199
MASTER         8
PLATINUM    1107
SILVER      3010
UNRANKED    1979
dtype: int64
```

Another problem that might be causing inaccurate predictions is, as it was stated earlier in section 1.2 a player's stats of a single game is not indicative of the highest rank they may have achieved. A player's stats are often varied from game to game. There also may be better techniques and algorithms for approaching this type of data. Feature selection is also another problem that might have caused inaccuracies. We excluded features that we believed weren't important for prediction.

For future improvements in our predictor we could utilize other algorithms (not yet learned) to improve the accuracy of our data. Our feature selection could be more efficient. We could also use a bigger sample size since we only had 10,000 players when the player base of LoL exceeds 10 million players in season 3.