League of Legends eSports: Machine Learning Clustering Report

Ryan Transfiguracion

July 19, 2018

Machine Learning the Data

Now that we have analyzed the data sets and confirmed that the five team roles have distinguishing characteristics, we can start running some machine learning algorithms to classify the individual observations into these team roles.

These are the 17 variables/features that we'll be using for our machine learning approach.

variableName	description
kills	The number of enemy champions killed.
assists	The number of enemy champions assisted in killing.
magicDamageDealt	The amount of magic damage dealt.
physicalDamageDealt	The amount of physical damage dealt.
magicDamageDealtToChampions	The amount of magic damage dealt to enemy champions only.
physicalDamageDealtToChampions	The amount of physical damage dealt to enemy champions only.
totalHeal	The amount of health points the player has regained.
totalUnitsHealed	The number of entities a player healed.
damageSelfMitigated	The amount of health points that were not lost from damage.
totalDamageTaken	The amount of damage a player took from various sources.
neutralMinionsKilled	The number of neutral monsters killed by a player.
timeCCingOthers	The weighted sum of all CC applied
totalTimeCrowdControlDealt	The sum of all CC applied
champLevel	The (experience) level of a player at the end of a match.
visionWardsBoughtInGame	The number of wards (i.e. surveillance items) a player purchased.
wardsPlaced	The number of wards a player placed in the arena.
wardsKilled	The number of enemy wards a player destroyed.

Approach overview

In attempting to improve the team role auto-assignment system implemented by Riot Games, I will use two classification algorithms in these steps:

1. ***k-means*** will be used on a training set to create a training model out of the resulting centroids of the clusters.

For our training set, I will use the per-game season averages of each player in each of the 2018 Spring Split leagues we have available (NA LCS, EU LCS, LCK, and LMS). I will concatenate the four data sets together, split the new data set by 40/60 according to (assumed) team role, and use the 40% split as the training set.

2. ***k nearest neighbor*** (with `k = 1`) will be used on test sets to classify individual observations to the centroids of the training model created from Step 1.

For our test sets, the first set will be the 60% split of the per-game player season averages set, and then the second set will be the game-by-game player performance data of the four leagues plus the 2018 Mid-Season Invitational.

Note that the second test set is the one most directly connected to solving the problem: predicting a player's team role at the conclusion of a match.

How will success be evaluated?

We will evaluate success by placing the predicted results in a confusion matrix and calculating some characteristics of the matrix. Specifically, we will be using the `confusionMatrix()` function from the `caret` library.

Now, let's get on with the machine learning!

Prepping the Data

First, we will put all the per-game player averages data sets together.

```
# Putting all leagues together
all_leagues_summoner_avgs <-
  eulcs_season_summoner_avgs %>%
  bind_rows(lms_season_summoner_avgs) %>%
  bind_rows(lck_season_summoner_avgs) %>%
  bind_rows(nalcs_season_summoner_avgs) %>%
  # Removing players who haven't played at least six games.
  filter(wins + losses >= 6)
str(all_leagues_summoner_avgs %>% select(1:10))
```

```
## 'data.frame':    233 obs. of  10 variables:
##  $ teamName    : chr  "FC Schalke 04" "FC Schalke 04" "FC Schalke 04" "FC Schalke 04" ...
##  $ summonerName: chr  "S04 Nukeduck" "S04 Pride" "S04 Upset" "S04 Vander" ...
##  $ wins        : int  7 7 6 6 7 20 8 20 20 20 ...
##  $ losses      : int  11 11 10 10 11 5 1 5 5 5 ...
##  $ teamRole    : Factor w/ 5 levels "BOTCARRY","JUNGLE",..: 3 2 1 4 5 2 5 3 4 1 ...
##  $ duration    : num  2262 2262 2272 2272 2262 ...
##  $ kills       : num  2.778 1.278 3.562 0.188 1.389 ...
##  $ deaths      : num  1.56 1.28 1.62 2.31 2.33 ...
##  $ assists     : num  3.72 5.17 3.44 6.38 3.78 ...
##  $ KDA         : num  4.18 5.04 4.31 2.84 2.21 ...
```

We will also put the match-by-match players data sets together.

```
all_leagues_match_player_stats <-
  nalcs_season_match_player_stats %>%
  bind_rows(eulcs_season_match_player_stats) %>%
  bind_rows(lck_season_match_player_stats) %>%
  bind_rows(lms_season_match_player_stats) %>%
  bind_rows(msi_season_match_player_stats) %>%
  mutate(roleLane = paste(role, lane, sep = ", "))
str(all_leagues_match_player_stats %>% select(1:10))
```

```
## 'data.frame':    6960 obs. of  10 variables:
##  $ participantId            : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ teamId                   : Factor w/ 2 levels "Blue","Red": 1 1 1 1 1 2 2 2 2 2 ...
##  $ championId               : int  41 79 90 18 44 150 102 13 429 12 ...
##  $ spell1Id                 : int  4 11 4 4 3 4 11 4 4 4 ...
##  $ spell2Id                 : int  12 4 7 7 4 12 4 7 7 14 ...
##  $ highestAchievedSeasonTier: Factor w/ 1 level "UNRANKED": 1 1 1 1 1 1 1 1 1 1 ...
##  $ summonerName             : chr  "TL Impact" "TL Xmithie" "TL Pobelter" "TL Doublelift" ...
```

```
##  $ profileIcon              : int  7 28 7 7 7 4 7 4 7 23 ...
##  $ name                     : chr  "Gangplank" "Gragas" "Malzahar" "Tristana" ...
##  $ win                      : logi  TRUE TRUE TRUE TRUE TRUE FALSE ...
```

Now, we will split the "averages" data set into training and testing sets. As stated before, 40% of the set will be used for training, while 60% will be used for testing.

```
# Split dataset into training and testing
library(caret)
set.seed(1234)
train_index <- caret::createDataPartition(all_leagues_summoner_avgs$teamRole, p = 0.4, list = FALSE, ti
train_avgs_data <- all_leagues_summoner_avgs[train_index,]
test_avgs_data <- all_leagues_summoner_avgs[-train_index,]
str(train_avgs_data %>% select(1))
```

```
## 'data.frame':    96 obs. of  1 variable:
##  $ teamName: chr  "FC Schalke 04" "FC Schalke 04" "Fnatic" "Fnatic" ...
```

```
str(test_avgs_data %>% select(1))
```

```
## 'data.frame':    137 obs. of  1 variable:
##  $ teamName: chr  "FC Schalke 04" "FC Schalke 04" "FC Schalke 04" "Fnatic" ...
```

With all the data that we need prepared, we can start the training phase.

Training a Model

First, since the numeric values of the 17 features can vary dramatically amongst each other, we will scale the variables of the data using ***z-score***. Thus, the values represent how many standard deviations away they are from the mean, where a value of zero represents the mean.

```
train_avgs_data_scaled <- train_avgs_data %>%
  select(kills, assists, magicDamageDealt, physicalDamageDealt,
         magicDamageDealtToChampions, physicalDamageDealtToChampions,
         totalHeal, totalUnitsHealed, damageSelfMitigated, totalDamageTaken,
         neutralMinionsKilled, timeCCingOthers, totalTimeCrowdControlDealt,
         champLevel, visionWardsBoughtInGame, wardsPlaced, wardsKilled) %>%
  # Use z-value scaling of the features.
  scale()
```

Now, we will run the ***k-means*** algorithm through this training set to create our training model.

```
library("cluster")
# Using k-means on the training set to make centroids
set.seed(1234)
train_fit.km <- kmeans(train_avgs_data_scaled, 5, iter.max = 1000)
```

Let's see how accurate the ***k-means*** algorithm was in creating the training model.

```
km_train_table <- table(train_fit.km$cluster, train_avgs_data$teamRole)
# Re-order the table rows for aesthetic purposes
km_train_table <- km_train_table[c(2,1,3,5,4),]
rownames(km_train_table) <- c("BOTCARRY", "JUNGLE", "MID", "SUPPORT", "TOP")
kable(km_train_table, row.names = TRUE, caption =
        "k-means: Creating Training Model (Predicted Rows vs. Actual Columns)")
```

Table 2: k-means: Creating Training Model (Predicted Rows vs. Actual Columns)

	BOTCARRY	JUNGLE	MID	SUPPORT	TOP
BOTCARRY	18	0	0	0	0
JUNGLE	0	22	0	0	0
MID	0	0	20	0	0
SUPPORT	0	0	0	17	0
TOP	0	0	0	0	19

We see that in this run, the training model is 100% accurate. Every observation in the training set was clustered into the correct team role.

Now, we will use the centroids created from the ***k-means*** algorthm as the model for running the ***knn*** algorithm through the testing set.

Testing the Model (Per Game Player Averages Test Set)

Just like with the training set, we will scale our selected features using z-score scaling. We will then use the scaled set to run the ***knn*** algorithm through its observations. Each observation will select the nearest centroid from the training model as the cluster that it belongs to.

```
# Using knn to classify observations in the test set
test_avgs_data_scaled <- test_avgs_data %>%
  select(kills, assists, magicDamageDealt, physicalDamageDealt,
         magicDamageDealtToChampions, physicalDamageDealtToChampions,
         totalHeal, totalUnitsHealed, damageSelfMitigated, totalDamageTaken,
         neutralMinionsKilled, timeCCingOthers, totalTimeCrowdControlDealt,
         champLevel, visionWardsBoughtInGame, wardsPlaced, wardsKilled) %>%
  # Use z-score scaling of the features.
  scale()
library(FNN)
set.seed(1234)
knn_pred_test_avgs <- get.knnx(train_fit.km$centers, test_avgs_data_scaled, 1)$nn.index[, 1]
```

As with the training phase, we will look at a table of the results to see how accurate the model was.

```
knn_test_avgs_table <- table(knn_pred_test_avgs, test_avgs_data$teamRole)
knn_test_avgs_table <- knn_test_avgs_table[c(2,1,3,5,4),]
rownames(knn_test_avgs_table) <- c("BOTCARRY", "JUNGLE", "MID", "SUPPORT", "TOP")
kable(knn_test_avgs_table, row.names = TRUE, caption =
        "knn: Predict Team Roles w/Per-Game Averages Test Data Set (Predicted Rows vs. Actual Columns)")
```

Table 3: knn: Predict Team Roles w/Per-Game Averages Test Data Set (Predicted Rows vs. Actual Columns)

	BOTCARRY	JUNGLE	MID	SUPPORT	TOP
BOTCARRY	27	0	0	0	0
JUNGLE	0	31	0	0	0
MID	0	0	28	0	0
SUPPORT	0	0	0	24	0
TOP	0	0	0	0	27

The ***knn*** algorithm has proven to be accurate with the per-game player averages data set.

Finally, we will see how accurate this model is when predicting team roles for individual game performances.

Testing the Model (Game-By-Game Player Match Statistics Test Set)

Just like with the previous two steps, we scale the features of the test set by z-score and then run ***knn*** through the set.

```
all_leagues_match_player_stats_scaled <- all_leagues_match_player_stats %>%
  select(kills, assists, magicDamageDealt, physicalDamageDealt,
         magicDamageDealtToChampions, physicalDamageDealtToChampions,
         totalHeal, totalUnitsHealed, damageSelfMitigated, totalDamageTaken,
         neutralMinionsKilled, timeCCingOthers, totalTimeCrowdControlDealt,
         champLevel, visionWardsBoughtInGame, wardsPlaced, wardsKilled) %>%
  # Use z-score scaling of the features.
  scale()

set.seed(1234)

knn_pred_test_single_games <- get.knnx(train_fit.km$centers,
                                       all_leagues_match_player_stats_scaled,
                                       1)$nn.index[, 1]

knn_test_singles_table <- table(knn_pred_test_single_games,
                                all_leagues_match_player_stats$teamRole)
knn_test_singles_table <- knn_test_singles_table[c(2,1,3,5,4),]
rownames(knn_test_singles_table) <- c("BOTCARRY", "JUNGLE", "MID", "SUPPORT", "TOP")
kable(knn_test_singles_table, row.names = TRUE, caption =
        "knn: Predict Team Roles Single Games Test Data Set (Predicted Rows vs. Actual Columns)")
```

Table 4: knn: Predict Team Roles Single Games Test Data Set (Predicted Rows vs. Actual Columns)

	BOTCARRY	JUNGLE	MID	SUPPORT	TOP
BOTCARRY	1346	11	15	0	150
JUNGLE	0	1351	1	27	11
MID	10	2	1325	1	116
SUPPORT	0	15	1	1361	0
TOP	36	13	50	3	1115

We can see that, finally, the model is not 100% accurate, as expected. Let's look at the confusion matrix statistics.

```
caret::confusionMatrix(knn_test_singles_table)
```

```
## Confusion Matrix and Statistics
##
##
## knn_pred_test_single_games BOTCARRY JUNGLE  MID SUPPORT  TOP
##                   BOTCARRY     1346     11   15       0  150
##                   JUNGLE          0   1351    1      27   11
##                   MID            10      2 1325       1  116
##                   SUPPORT         0     15    1    1361    0
##                   TOP            36     13   50       3 1115
##
## Overall Statistics
##
##                Accuracy : 0.9336
##                  95% CI : (0.9275, 0.9394)
##     No Information Rate : 0.2
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.917
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: BOTCARRY Class: JUNGLE Class: MID
## Sensitivity                   0.9670        0.9705     0.9519
## Specificity                   0.9684        0.9930     0.9768
## Pos Pred Value                0.8844        0.9719     0.9113
## Neg Pred Value                0.9915        0.9926     0.9878
## Prevalence                    0.2000        0.2000     0.2000
## Detection Rate                0.1934        0.1941     0.1904
## Detection Prevalence          0.2187        0.1997     0.2089
## Balanced Accuracy             0.9677        0.9818     0.9643
##                      Class: SUPPORT Class: TOP
## Sensitivity                  0.9777     0.8010
## Specificity                  0.9971     0.9817
## Pos Pred Value               0.9884     0.9162
## Neg Pred Value               0.9944     0.9518
## Prevalence                   0.2000     0.2000
## Detection Rate               0.1955     0.1602
```

```
## Detection Prevalence         0.1978     0.1749
## Balanced Accuracy            0.9874     0.8913
```

A 93% **accuracy** sounds pretty good, as do the 0.917 **Kappa** value and the 20% **no-information rate** – if one were to guess one team role for every single observation, then they'd be correct 20% of the time, which is ideal.

However, the ***Top Laner*** role suffers from a considerably lower **sensitivity** (true positive rate) compared to the other team roles because of a large number of false negatives, though its **specificity** (true negative rate) is right up there with the others.

We will add a couple more features to the model to see if we can get it even more accurate while also increasing the **sensitivity** of the ***Top Laner*** role. Here are the added features:

newVarName	newDescription
physDmgToChampsToDmgTakenRatio	Physical damage dealt to champions / Total damage taken
totalMinionsKilled	The number of enemy minions killed.
damageTakenPerMinDeltas.0.10	The amount of damage taken per minute from 0:00 to 10:00 minutes.
creepsPerMinDeltas.0.10	The number of enemy minions killed per minute from 0:00 to 10:00 minutes.

Now, we will evaluate the updated model with these four additional features with a new confusion matrix.

```
## Confusion Matrix and Statistics
##
##
## knn_pred_test_single_games BOTCARRY JUNGLE  MID SUPPORT  TOP
##                   BOTCARRY     1321      1   10       0   68
##                   JUNGLE          0   1378    1      14    4
##                   MID            19      1 1321       0  120
##                   SUPPORT         0      7    1    1376    0
##                   TOP            52      5   59       2 1200
##
## Overall Statistics
##
##                Accuracy : 0.9477
##                  95% CI : (0.9422, 0.9528)
##     No Information Rate : 0.2
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9346
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: BOTCARRY Class: JUNGLE Class: MID
## Sensitivity                   0.9490        0.9899     0.9490
## Specificity                   0.9858        0.9966     0.9749
## Pos Pred Value                0.9436        0.9864     0.9042
## Neg Pred Value                0.9872        0.9975     0.9871
## Prevalence                    0.2000        0.2000     0.2000
## Detection Rate                0.1898        0.1980     0.1898
## Detection Prevalence          0.2011        0.2007     0.2099
## Balanced Accuracy             0.9674        0.9933     0.9619
##                      Class: SUPPORT Class: TOP
## Sensitivity                  0.9885     0.8621
```

```
## Specificity                 0.9986     0.9788
## Pos Pred Value              0.9942     0.9105
## Neg Pred Value              0.9971     0.9660
## Prevalence                  0.2000     0.2000
## Detection Rate              0.1977     0.1724
## Detection Prevalence        0.1989     0.1894
## Balanced Accuracy           0.9935     0.9204
```

With the four features added to the model, we were able to increase its accuracy from 93.36% to 94.77%, while the Kappa value increasred from .917 to .9346.

Concerning the team roles, while the sensitivities of ***Botcarry*** and ***Mid*** decreased slightly, those of ***Jungle*** and ***Support*** increased to nearly 99%, while the ***Top Laner Role*** received a six percentage points boost.

Analyzing the Model

A 94.7% accurate model looks great, but let's analyze the ***false negatives*** of the lowest-sensitivity team role, the ***Top Laner***:

Table 6: False Negative Top Laner Champions vs Predicted Role

	BOTCARRY	JUNGLE	MID
Camille	2	0	0
Cassiopeia	0	0	3
Cho'Gath	0	0	1
Fiora	4	0	0
Gangplank	40	0	0
Illaoi	1	0	0
Jayce	12	0	0
Kassadin	0	0	1
Kennen	0	0	2
Lucian	7	0	0
Riven	1	0	0
Rumble	0	0	3
Ryze	0	0	11
Swain	0	0	23
Vladimir	0	4	76
Yasuo	1	0	0

As we see, the three largest ***false negative*** champions of the ***Top Laner*** role are **Vladimir**, **Swain**, and **Gangplank**.

In the cases of **Swain** and **Vladimir**, both are champions who are capable of dealing high amounts of magic damage, which is a defining feature of a typical ***Mid Laner***, while also having high sustainability and survivability, which is typical of a ***Top Laner***. It is very likely that their magic damage is the culprit for mis-classifiying these particular observations for these champions as ***Mid Laners***.

Additionally, **Vladimir** has abilities that are capable of life steal, which is a defining feature (as **totalHeal**) of items almost always equipped by ***Junglers***, which would explain why four **Vladimir** observations were mis-classified as ***Junglers***.

Finally, for **Gangplank**, while he's not an inherently durable champion, he has a set of abilities that give him high survivability that is important for the ***Top Laner*** role. However, he's also capable of being one of the highest physical damage-dealing champions, which is more typical of a ***Botlane Carry*** role.

Furthermore, **Gangplank** deals such an atypically high amount of damage for a ***Top Laner*** that he was considered **overpowering** (OP), and he was banned from **half of all games** in our data sets, as displayed below. Note that **Swain** and **Vladimir** are also in the Top 25 of most-banned champions, at around 20% each.

```
all_leagues_champ_bans <-
  read.csv("../datasets/nalcs/nalcs_spring2018_champ_bans.csv") %>%
  bind_rows(read.csv("../datasets/eulcs/eulcs_spring2018_champ_bans.csv")) %>%
  bind_rows(read.csv("../datasets/lck/lck_spring2018_champ_bans.csv")) %>%
  bind_rows(read.csv("../datasets/lms/lms_spring2018_champ_bans.csv")) %>%
  bind_rows(read.csv("../datasets/msi/msi_2018_champ_bans.csv"))
champ_bans_ranked <- all_leagues_champ_bans %>%
  group_by(name) %>% tally(sort = TRUE) %>% rename(bans = n) %>%
  mutate(banRate = round(bans / (nrow(all_leagues_champ_bans) / 10), digits = 3))
champ_bans_ranked$rank <- 1:nrow(champ_bans_ranked)
champ_bans_ranked <- champ_bans_ranked[, c(4, 1:3)] %>%
  top_n(25)
kable(champ_bans_ranked, caption = "Top 25 Most-Banned Champions, Spring 2018 Season + MSI")
```

Table 7: Top 25 Most-Banned Champions, Spring 2018 Season + MSI

rank	name	bans	banRate
1	Gangplank	348	0.501
2	Camille	332	0.478
3	Galio	313	0.450
4	Ryze	309	0.445
5	Azir	271	0.390
6	Zoe	257	0.370
7	Tahm Kench	242	0.348
8	Sejuani	217	0.312
9	Kalista	204	0.294
10	Braum	196	0.282
11	Varus	195	0.281
12	Skarner	191	0.275
13	Alistar	184	0.265
14	Xayah	184	0.265
15	Ezreal	173	0.249
16	Gnar	165	0.237
17	Olaf	152	0.219
18	Ornn	152	0.219
19	Taliyah	152	0.219
20	Kha'Zix	151	0.217
21	Zac	146	0.210
22	Sion	144	0.207
23	Swain	142	0.204
24	Vladimir	137	0.197
25	Kog'Maw	122	0.176

Conclusion

While exploring, analyzing, and visualizing the players' match data of four professional eSports leagues and an international tournament, we were able to discover distinguishable features of the five team roles (Top, Jungle, Mid, Bot Carry, and Support) when we calculated the per-game averages of individual players.

Based on this discovery, we first split the per-game averages per-player dataset into a 40/60 train/test pair. Then, we ran the **k-means clustering** algorithm through the training set, which showed 100 percent accuracy in classifying each per-game averages observation to one of five team roles. We then used the centroids computed by the k-means algorithm as the ***model*** for running the **k-nearest-neighbor (knn) clustering** algorithm through the per-game averages testing/validating set, which also showed 100 percent accuracy. Finally, we used the same **k-means model** to run the **knn algorithm** through the ***real*** test set, which contained single-game performance data of individual players. The **knn** run through the single-game performance data set showed that the model was **94.7 percent accurate** and achieved a **Kappa score of 0.9346**.

To conclude, the **k-means centroids model** proved to be very accurate in classifying individual game performances of the Spring 2018 Season to an appropriate team role via the **knn algorithm**. While not absolutely perfect, this model (and model-making process) could be an excellent tool for classifying team roles in Normal/Blind Pick Summoner's Rift game modes, where team roles cannot be assumed based on Participant ID alone. However, we must also consider caveats such as these:

Metas Change: Riot Games routinely makes changes to League of Legends – sometimes minor but sometimes paradigm-shifting. What may be a winning strategy in the present could be a losing one in the future, and that occasionally includes changing certain champions' team roles.

Normal/Blind Pick Game Mode is Lower-Stakes: Because the competitiveness of this game mode can often times be less intense, players are more likely to experiment, which includes playing champions in atypical team roles.

Unique Champions: As mentioned before, some champions can play and succeed in a team role but may output performance data that is fitting of a different role. Swain and Vladimir are powerful Top Laners but are also mages. Gangplank is a dominant Top Laner with a damage output comparable to a Botlane Carry. Ivern is a Jungler but almost never equips jungler items and plays more like a Support.

Pros are Pros: The level of competition in a professional LoL eSports league is considerable step up from the Regular Joes playing the game. Even within the LoL game itself, the ranking system divides players in multiple tiers, where the "worst" players are in the Bronze division, while professionals (playing in their free time) and near-professionals are usually in the Master or Challenger divisions. Thus, the average output data of team roles may vary considerably from division to division.