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.

Let's take a look at the champions who are guilty of creating the **false negatives** for the ***Top Laner*** role aka, actual ***Top Laners*** who were mis-classified to another role:

	BOTCARRY	JUNGLE	MID
Camille	31	0	0
Cassiopeia	0	0	3
Cho'Gath	0	0	1
Darius	1	0	0
Ezreal	1	0	0
Fiora	9	0	0
Gangplank	72	0	0
Gnar	1	0	0
Illaoi	3	1	0
Irelia	1	0	0
Jax	2	0	0
Jayce	14	1	0
Kassadin	0	0	1
Kennen	0	0	1
Kled	1	0	0
Lucian	8	0	0
Rammus	0	0	1
Riven	2	0	0
Rumble	0	0	3
Ryze	0	0	11
Shen	2	0	0
Swain	0	0	24
Vladimir	0	9	71
Yasuo	2	0	0