Searching for Similarity: Dimensionality Reduction

Isabelle Kirby, Bridgette Bryant, Rikita Patangay, Zuhayr Ali

Dimensionality Reduction for Korea's top high elo teams in League of Legends

I utilized dimensionality reduction to examine the dataset. It is used to reduced the training data into less varaiables and re-examine it. It is commonly used on datasets that have large numbers of observations such as the dataset we have which has over 100k values. -Rikita

Organizing the data

We have to start out by cleaning the data meaning organize it in the way we want to explore it. We will start off by unzipping the archive and save the seperate dataset files into data frames.

```
win_stats_df <- read.csv((unz("League_Data/league_korea_high_elo_team_stats.zip", "win_team_stats.csv")

lose_stats_df <- read.csv((unz("League_Data/league_korea_high_elo_team_stats.zip", "lose_team_stats.csv

str(win_stats_df)
```

```
## 'data.frame':    90500 obs. of  31 variables:
##  $ win_kills1                     : num  10 3 7 11 0 7 9 8 18 4 ...
##  $ win_kills2                     : num  4 7 3 4 4 1 10 10 5 2 ...
##  $ win_kills3                     : num  4 0 5 7 2 11 9 5 2 2 ...
##  $ win_kills4                     : num  6 7 9 2 11 10 9 10 2 6 ...
##  $ win_kills5                     : num  7 2 4 3 8 8 2 0 11 4 ...
##  $ win_deaths1                    : num  4 5 5 2 1 3 2 5 2 3 ...
##  $ win_deaths2                    : num  1 0 1 2 4 0 2 5 6 0 ...
##  $ win_deaths3                    : num  4 0 2 2 4 3 3 4 8 1 ...
##  $ win_deaths4                    : num  4 0 1 3 3 6 1 7 4 0 ...
##  $ win_deaths5                    : num  2 3 2 4 4 5 5 6 8 1 ...
##  $ win_totalDamageDealtToChampions1: num  17898 16662 16241 12111 10900 ...
##  $ win_totalDamageDealtToChampions2: num  15800 11674 7572 5510 11362 ...
##  $ win_totalDamageDealtToChampions3: num  10786 7498 10024 8440 14494 ...
##  $ win_totalDamageDealtToChampions4: num  16964 13016 15115 5373 27391 ...
##  $ win_totalDamageDealtToChampions5: num  11568 11393 12395 11038 22399 ...
##  $ win_goldEarned1                : num  9802 8452 9029 10175 7217 ...
##  $ win_goldEarned2                : num  9203 9069 6921 5552 10497 ...
##  $ win_goldEarned3                : num  11127 6023 8331 7439 10323 ...
##  $ win_goldEarned4                : num  9286 9868 11860 5873 13499 ...
##  $ win_goldEarned5                : num  10414 7660 8589 7033 12720 ...
##  $ win_visionScore1               : num  28 14 11 17 42 41 19 23 72 9 ...
##  $ win_visionScore2               : num  16 27 35 25 38 41 46 55 50 21 ...
```

```
##  $ win_visionScore3                : num  23 46 25 17 30 21 25 47 29 13 ...
##  $ win_visionScore4                : num  17 16 15 9 18 39 31 25 80 19 ...
##  $ win_visionScore5                : num  36 22 21 19 26 19 86 70 22 10 ...
##  $ win_totalTimeCrowdControlDealt1 : num  183 33 178 134 69 310 168 279 365 15 ...
##  $ win_totalTimeCrowdControlDealt2 : num  92 291 82 61 503 45 291 493 119 62 ...
##  $ win_totalTimeCrowdControlDealt3 : num  231 31 371 332 562 133 102 287 456 126 ...
##  $ win_totalTimeCrowdControlDealt4 : num  54 235 140 274 79 78 444 501 215 209 ...
##  $ win_totalTimeCrowdControlDealt5 : num  281 407 122 163 69 73 92 193 300 168 ...
##  $ gameId                          : num  4.25e+09 4.25e+09 4.26e+09 4.26e+09 4.26e+09 ...
```

```
str(lose_stats_df)
```

```
## 'data.frame':    90500 obs. of  31 variables:
##  $ lose_kills1                     : num  3 0 3 1 4 7 0 11 3 0 ...
##  $ lose_kills2                     : num  0 2 5 6 2 1 3 5 10 1 ...
##  $ lose_kills3                     : num  4 3 1 2 4 4 4 3 7 2 ...
##  $ lose_kills4                     : num  4 3 1 1 1 5 2 7 2 2 ...
##  $ lose_kills5                     : num  4 0 1 3 5 0 4 1 6 0 ...
##  $ lose_deaths1                    : num  6 4 7 6 7 7 10 10 7 5 ...
##  $ lose_deaths2                    : num  6 6 4 3 6 9 5 4 6 2 ...
##  $ lose_deaths3                    : num  5 3 7 7 1 11 8 6 10 2 ...
##  $ lose_deaths4                    : num  7 4 5 4 7 4 9 7 10 3 ...
##  $ lose_deaths5                    : num  7 2 5 7 4 6 7 6 5 6 ...
##  $ lose_totalDamageDealtToChampions1: num  10844 4618 7096 9492 9686 ...
##  $ lose_totalDamageDealtToChampions2: num  7095 14837 17030 8557 14045 ...
##  $ lose_totalDamageDealtToChampions3: num  13458 9197 8735 6679 24086 ...
##  $ lose_totalDamageDealtToChampions4: num  9670 10035 7849 4058 2959 ...
##  $ lose_totalDamageDealtToChampions5: num  14972 5531 5815 6912 16719 ...
##  $ lose_goldEarned1                : num  6844 4524 6551 5442 7928 ...
##  $ lose_goldEarned2                : num  5205 8823 7562 7846 8042 ...
##  $ lose_goldEarned3                : num  8226 7788 5346 5092 12502 ...
##  $ lose_goldEarned4                : num  7911 9008 5004 3931 6185 ...
##  $ lose_goldEarned5                : num  8815 6993 5931 5071 10729 ...
##  $ lose_visionScore1               : num  14 30 14 1 25 11 17 46 30 13 ...
##  $ lose_visionScore2               : num  34 16 13 27 10 48 30 34 25 7 ...
##  $ lose_visionScore3               : num  8 27 40 9 29 14 38 19 39 13 ...
##  $ lose_visionScore4               : num  21 28 15 26 65 15 81 44 84 24 ...
##  $ lose_visionScore5               : num  14 10 9 5 28 13 13 68 45 8 ...
##  $ lose_totalTimeCrowdControlDealt1 : num  19 80 133 71 422 188 97 71 246 20 ...
##  $ lose_totalTimeCrowdControlDealt2 : num  38 67 249 476 151 77 36 327 98 102 ...
##  $ lose_totalTimeCrowdControlDealt3 : num  173 491 135 13 161 109 347 433 36 169 ...
##  $ lose_totalTimeCrowdControlDealt4 : num  305 50 125 52 63 204 208 44 95 47 ...
##  $ lose_totalTimeCrowdControlDealt5 : num  237 0 226 88 97 101 81 242 447 182 ...
##  $ gameId                          : num  4.25e+09 4.25e+09 4.26e+09 4.26e+09 4.26e+09 ...
```

The below functions will merge the win_stats_df and lose_stats_df together based of the gameId column and if it matches. The model will then be able to categorize our data by team 0 or team 1. This way it is easier to examine the winning teams because of the differing values.

```
# Replacing column names for rbind
colnames(win_stats_df) <- c('kill1', 'kill2', 'kill3', 'kill4', 'kill5', 'death1', 'death2', 'death3',
colnames(lose_stats_df) <- c('kill1', 'kill2', 'kill3', 'kill4', 'kill5', 'death1', 'death2', 'death3',
# Adding column based on dataset it is in
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
win_stats_df <- win_stats_df %>%
  mutate(won=1)
lose_stats_df <- lose_stats_df %>%
  mutate(won =0)

#full_stats_df <- merge(win_stats_df, lose_stats_df, by = "gameId")
full_stats_df <- rbind(win_stats_df, lose_stats_df)
drop <- c("gameId")
full_stats_df <- full_stats_df[,!(names(full_stats_df) %in% drop)]
str(full_stats_df)
```

```
## 'data.frame':    181000 obs. of  31 variables:
##  $ kill1                       : num  10 3 7 11 0 7 9 8 18 4 ...
##  $ kill2                       : num  4 7 3 4 4 1 10 10 5 2 ...
##  $ kill3                       : num  4 0 5 7 2 11 9 5 2 2 ...
##  $ kill4                       : num  6 7 9 2 11 10 9 10 2 6 ...
##  $ kill5                       : num  7 2 4 3 8 8 2 0 11 4 ...
##  $ death1                      : num  4 5 5 2 1 3 2 5 2 3 ...
##  $ death2                      : num  1 0 1 2 4 0 2 5 6 0 ...
##  $ death3                      : num  4 0 2 2 4 3 3 4 8 1 ...
##  $ death4                      : num  4 0 1 3 3 6 1 7 4 0 ...
##  $ death5                      : num  2 3 2 4 4 5 5 6 8 1 ...
##  $ totalDamageDealtToChampions1: num  17898 16662 16241 12111 10900 ...
##  $ totalDamageDealtToChampions2: num  15800 11674 7572 5510 11362 ...
##  $ totalDamageDealtToChampions3: num  10786 7498 10024 8440 14494 ...
##  $ totalDamageDealtToChampions4: num  16964 13016 15115 5373 27391 ...
##  $ totalDamageDealtToChampions5: num  11568 11393 12395 11038 22399 ...
##  $ goldEarned1                 : num  9802 8452 9029 10175 7217 ...
##  $ goldEarned2                 : num  9203 9069 6921 5552 10497 ...
##  $ goldEarned3                 : num  11127 6023 8331 7439 10323 ...
##  $ goldEarned4                 : num  9286 9868 11860 5873 13499 ...
##  $ goldEarned5                 : num  10414 7660 8589 7033 12720 ...
##  $ visionScore1                : num  28 14 11 17 42 41 19 23 72 9 ...
##  $ visionScore2                : num  16 27 35 25 38 41 46 55 50 21 ...
##  $ visionScore3                : num  23 46 25 17 30 21 25 47 29 13 ...
##  $ visionScore4                : num  17 16 15 9 18 39 31 25 80 19 ...
##  $ visionScore5                : num  36 22 21 19 26 19 86 70 22 10 ...
##  $ totalTimeCrowdControlDealt1 : num  183 33 178 134 69 310 168 279 365 15 ...
##  $ totalTimeCrowdControlDealt2 : num  92 291 82 61 503 45 291 493 119 62 ...
##  $ totalTimeCrowdControlDealt3 : num  231 31 371 332 562 133 102 287 456 126 ...
##  $ totalTimeCrowdControlDealt4 : num  54 235 140 274 79 78 444 501 215 209 ...
##  $ totalTimeCrowdControlDealt5 : num  281 407 122 163 69 73 92 193 300 168 ...
##  $ won                         : num  1 1 1 1 1 1 1 1 1 1 ...
```

Next let's randomly divide the data into train and test:

```
set.seed(1010)
i <- sample(1:nrow(full_stats_df), .75*nrow(full_stats_df), replace=FALSE)
full_stats_train <- full_stats_df[i,]
full_stats_test <- full_stats_df[-i,]
```

Dimensionality Reduction:

Principal Components Analysis:

We will now perform a PCA (Principal Components Analysis)

```
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
pca_out <- preProcess(full_stats_train[,1:31],method=c("center","scale","pca"))
pca_out
```

```
## Created from 135750 samples and 31 variables
##
## Pre-processing:
##   - centered (31)
##   - ignored (0)
##   - principal component signal extraction (31)
##   - scaled (31)
##
## PCA needed 20 components to capture 95 percent of the variance
```

We see that PCA reduced the 31 variables to 20 principal components. These twenty components capture 95% of the variance in the data.

Linear Discriminant Analysis

We will now perform a LDA (Linear Discriminant Analysis)

```
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```r
options(max.print = 10000)
lda1 <- lda(kill1~., data=full_stats_train)
summary(lda1$means)
```

```
##      kill2            kill3            kill4            kill5
##  Min.   : 1.000   Min.   : 1.000   Min.   : 1.000   Min.   : 0.000
##  1st Qu.: 4.750   1st Qu.: 4.665   1st Qu.: 4.499   1st Qu.: 4.500
##  Median : 5.467   Median : 5.466   Median : 5.598   Median : 5.468
##  Mean   : 5.091   Mean   : 5.420   Mean   : 5.339   Mean   : 5.284
##  3rd Qu.: 5.734   3rd Qu.: 5.719   3rd Qu.: 5.971   3rd Qu.: 5.804
##  Max.   :10.000   Max.   :13.000   Max.   :12.500   Max.   :11.000
##      death1           death2           death3           death4
##  Min.   : 0.000   Min.   : 0.000   Min.   : 0.000   Min.   : 0.000
##  1st Qu.: 4.583   1st Qu.: 4.501   1st Qu.: 4.489   1st Qu.: 4.596
##  Median : 4.755   Median : 5.477   Median : 5.458   Median : 5.634
##  Mean   : 4.615   Mean   : 5.185   Mean   : 5.259   Mean   : 5.666
##  3rd Qu.: 5.271   3rd Qu.: 6.315   3rd Qu.: 6.390   3rd Qu.: 6.357
##  Max.   :12.000   Max.   :11.000   Max.   :14.000   Max.   :19.000
##      death5      totalDamageDealtToChampions1 totalDamageDealtToChampions2
##  Min.   :0.000   Min.   : 4647                Min.   :  820
##  1st Qu.:4.639   1st Qu.:21988                1st Qu.:13165
##  Median :5.645   Median :33297                Median :16751
##  Mean   :5.088   Mean   :34318                Mean   :15726
##  3rd Qu.:6.485   3rd Qu.:44819                3rd Qu.:18304
##  Max.   :7.250   Max.   :78332                Max.   :41805
##  totalDamageDealtToChampions3 totalDamageDealtToChampions4
##  Min.   : 1033                Min.   :  621
##  1st Qu.:13106                1st Qu.:12672
##  Median :17440                Median :17055
##  Mean   :15687                Mean   :15442
##  3rd Qu.:18520                3rd Qu.:18656
##  Max.   :25925                Max.   :32784
##  totalDamageDealtToChampions5  goldEarned1     goldEarned2     goldEarned3
##  Min.   :  822                Min.   : 4873   Min.   : 3361   Min.   : 4054
##  1st Qu.:13130                1st Qu.:11627   1st Qu.: 9470   1st Qu.: 9761
##  Median :17140                Median :15483   Median :11075   Median :11138
##  Mean   :15693                Mean   :15042   Mean   :10374   Mean   :10275
##  3rd Qu.:18432                3rd Qu.:18600   3rd Qu.:11692   3rd Qu.:11598
##  Max.   :26541                Max.   :21462   Max.   :17144   Max.   :12928
##   goldEarned4     goldEarned5     visionScore1    visionScore2
##  Min.   : 4293   Min.   : 3611   Min.   : 0.00   Min.   : 0.00
##  1st Qu.: 9281   1st Qu.: 9755   1st Qu.:22.61   1st Qu.:26.59
##  Median :10916   Median :11141   Median :26.52   Median :32.79
##  Mean   :10192   Mean   :10277   Mean   :22.73   Mean   :29.91
##  3rd Qu.:11613   3rd Qu.:11718   3rd Qu.:28.15   3rd Qu.:36.61
##  Max.   :13232   Max.   :13595   Max.   :30.33   Max.   :70.00
##   visionScore3    visionScore4    visionScore5   totalTimeCrowdControlDealt1
##  Min.   : 0.00   Min.   : 0.00   Min.   : 0.00   Min.   :  4.0
##  1st Qu.:22.88   1st Qu.:22.31   1st Qu.:22.65   1st Qu.:200.9
##  Median :31.02   Median :30.26   Median :31.72   Median :227.5
##  Mean   :26.93   Mean   :26.25   Mean   :25.97   Mean   :217.8
##  3rd Qu.:34.97   3rd Qu.:35.15   3rd Qu.:34.53   3rd Qu.:251.2
##  Max.   :42.67   Max.   :39.38   Max.   :39.00   Max.   :313.3
```

```
##  totalTimeCrowdControlDealt2 totalTimeCrowdControlDealt3
##  Min.   :  0.0               Min.   :  0.0
##  1st Qu.:165.4               1st Qu.:173.1
##  Median :203.4               Median :211.2
##  Mean   :181.3               Mean   :190.5
##  3rd Qu.:220.6               3rd Qu.:223.0
##  Max.   :287.0               Max.   :368.2
##  totalTimeCrowdControlDealt4 totalTimeCrowdControlDealt5      won
##  Min.   :  0.0               Min.   :  0.0               Min.   :0.0000
##  1st Qu.:166.3               1st Qu.:152.3               1st Qu.:0.6953
##  Median :198.8               Median :195.4               Median :0.8063
##  Mean   :180.6               Mean   :169.5               Mean   :0.7630
##  3rd Qu.:221.5               3rd Qu.:213.0               3rd Qu.:0.9122
##  Max.   :341.3               Max.   :270.9               Max.   :1.0000
```

Logistic Regression on reduced data

Here we will perform logistic regression on the dataset to be able to compare the difference of accuracy between the PCA and data set.

```
glm_won <- glm(won~., data=full_stats_train, family = "binomial")
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(glm_won)
```

```
##
## Call:
## glm(formula = won ~ ., family = "binomial", data = full_stats_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -6.3264  -0.1052   0.0003   0.1033   5.6621
##
## Coefficients:
##                                Estimate Std. Error z value Pr(>|z|)
## (Intercept)                  -1.499e+00  4.975e-02 -30.128  < 2e-16 ***
## kill1                         3.049e-01  7.629e-03  39.970  < 2e-16 ***
## kill2                         3.166e-01  7.711e-03  41.066  < 2e-16 ***
## kill3                         3.097e-01  7.679e-03  40.334  < 2e-16 ***
## kill4                         2.936e-01  7.589e-03  38.681  < 2e-16 ***
## kill5                         3.099e-01  7.706e-03  40.218  < 2e-16 ***
## death1                       -4.803e-01  7.664e-03 -62.665  < 2e-16 ***
## death2                       -4.942e-01  7.757e-03 -63.719  < 2e-16 ***
## death3                       -4.705e-01  7.587e-03 -62.016  < 2e-16 ***
## death4                       -4.829e-01  7.719e-03 -62.557  < 2e-16 ***
## death5                       -4.963e-01  7.691e-03 -64.526  < 2e-16 ***
## totalDamageDealtToChampions1  9.772e-06  3.141e-06   3.111 0.001865 **
## totalDamageDealtToChampions2  1.199e-05  3.107e-06   3.859 0.000114 ***
## totalDamageDealtToChampions3  1.489e-05  3.109e-06   4.789 1.67e-06 ***
## totalDamageDealtToChampions4  1.769e-05  3.148e-06   5.619 1.92e-08 ***
## totalDamageDealtToChampions5  5.626e-06  3.184e-06   1.767 0.077264 .
```

```
## goldEarned1                   1.844e-04  1.315e-05  14.031  < 2e-16 ***
## goldEarned2                   1.702e-04  1.320e-05  12.894  < 2e-16 ***
## goldEarned3                   1.838e-04  1.312e-05  14.015  < 2e-16 ***
## goldEarned4                   1.845e-04  1.323e-05  13.944  < 2e-16 ***
## goldEarned5                   1.913e-04  1.313e-05  14.567  < 2e-16 ***
## visionScore1                 -2.464e-02  9.945e-04 -24.776  < 2e-16 ***
## visionScore2                 -2.507e-02  9.797e-04 -25.594  < 2e-16 ***
## visionScore3                 -2.409e-02  9.837e-04 -24.488  < 2e-16 ***
## visionScore4                 -2.368e-02  9.726e-04 -24.343  < 2e-16 ***
## visionScore5                 -2.523e-02  9.830e-04 -25.665  < 2e-16 ***
## totalTimeCrowdControlDealt1  -6.336e-04  9.615e-05  -6.590 4.41e-11 ***
## totalTimeCrowdControlDealt2  -7.739e-04  9.444e-05  -8.194 2.52e-16 ***
## totalTimeCrowdControlDealt3  -4.803e-04  9.594e-05  -5.006 5.55e-07 ***
## totalTimeCrowdControlDealt4  -7.796e-04  9.705e-05  -8.033 9.54e-16 ***
## totalTimeCrowdControlDealt5  -6.206e-04  9.666e-05  -6.421 1.35e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 188189  on 135749  degrees of freedom
## Residual deviance:  32753  on 135719  degrees of freedom
## AIC: 32815
##
## Number of Fisher Scoring iterations: 8
```

After evaluating on the dataset we can see that...

```
library(caret)
glm_probs <- predict(glm_won, newdata = full_stats_test, type = "response")
glm_pred <- ifelse(glm_probs > 0.5, 1, 0)
glm_acc <- mean(glm_pred == full_stats_test$won)
confusionMatrix(as.factor(glm_pred), reference = as.factor(full_stats_test$won))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 21820  1101
##          1   922 21407
##
##                Accuracy : 0.9553
##                  95% CI : (0.9533, 0.9572)
##     No Information Rate : 0.5026
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9106
##
##  Mcnemar's Test P-Value : 7.574e-05
##
##             Sensitivity : 0.9595
##             Specificity : 0.9511
##          Pos Pred Value : 0.9520
```

```
##          Neg Pred Value : 0.9587
##              Prevalence : 0.5026
##          Detection Rate : 0.4822
##    Detection Prevalence : 0.5065
##       Balanced Accuracy : 0.9553
##
##        'Positive' Class : 0
##
```

The logistic regression model shows an accuracy of 95.5% and the PCA also brings an accuracy of 95%. This is very hopeful. The accuracy was not lost with the data.