Data Analysis on PUBG Dataset

Mingwei Wang 101086278
Yi Zhou 101086017
Zhiying Xiao 101063142

1 Summary

There are 150 original variables of 87,898 players in the PUBG game dataset. These variables are divided into 3 different group by the participated party size, such as Solo, Duo and Squad. After data cleaning and logic data reduction, 70,318 (80%) observations of players are selected to be training set, and the rest 17,580 (20%) is test set by simple random sampling method. Three different way of data visualization, such as bar chart, box plot and correlogram, provide us different perspectives to understand the data and perform data reduction more properly. After Scaling and standardizing data, Principal Component Analysis is performed respectively in three different party size groups, for the sake of reducing dimension. The first five main components in Solo Dataset and the first six main components in both Duo and Squad Dataset are used to perform clustering, a way of unsupervised learning. Note that, there dose not contain an independent section to describe how to reduct data in our report, since data reduction is performed and mentioned in each sections before unsupervised learning and supervised learning. Continually, KillDeathRatio is classified into five different classes, for it is considered as the metric to evaluate the level of individual players' comprehensive ability in PUBG. Then support vector machine, a way of supervised learning, is performed as well as optimized in training set, and then serves to predict the distribution of KillDeathRatio class of test set.

2 Data Pre-process

2.1 Variables

The PUBG dataset of is chosen for this research program. There are 87,898 user observations and 150 original variables.

Party Size	Description	Number of Original Variables
Solo	Play the game alone	50
Duo	Play the game in a 2-person team	50
Squad	Play the game in a 4-person team	50

Table 2.1.1 Party Size of PUBG Dataset

There are over 50 original variables of different party sizes in the dataset, but we just use over half of them to perform the further analysis. For instance, 28 variables remains in dataset of Duo and Squad, while merely 24 variables remains in Solo Dataset.

Variable	Description
KillDeathRatio	A metric to define how many kills players get before they die
WinRatio	Rate of winning the game
TimeSurvived	Total survival time in the game (minutes)
RoundsPlayed	The number of rounds already participated
HeadShotKillsPg	Average frequency of successful head shot kills
LongestKill	The longest distances of kill
Heals	Total frequency of acquiring heals
Assists	Total frequency of assisting others
DamagePG	Average damage
Suicide	Total frequency of suicide
...	...

Table 2.1.2 Description of Variables in Specific Party Size

2.2 Data Cleaning

Missing values were found by using *omit.na( )* function as well as the variables holding all zero-value entries unreasonably, such as Weapon Acquired in all three party size, as well as Revives and DBNOs in Solo size.

2.3 Redundant Variable Reduction

The variables can be regarded as redundancy, since all of these variables about frequency can be equivalent to other existing variables about proportion in the form of Ratio and PerGame,

based on the known variable of rounds each player has participated.

2.4 Ordinal Categorical Variables

Ordinal means that an 'order' is implied. For example, in our case, KillDeathRatio (K/D) is a ordinal variable since we can order it by the specific value of each kind of players.Since the range of KillDeathRatio vary from 0 to 100%, a classification of KillDeathRatio was preferred in this case to divided all value of variable KillDeathRatio into five groups to replace the original detailed ratio value of each player in our research. And the output of Supervised Learning Model was designed as the KillDeathRatio Class of each player.

KillDeathRatio Class	Player Classification	METRIC VALUE RANGE
1	Beginner	Less Than 1%
2	General Player	1% -- 1.5%
3	Senior Player	1.5% -- 2%
4	Master Player	2% -- 3%
5	Super Master Player	Greater Than 3%

Table 2.4 Ordinal Classification of KillDeathRatio

2.5 Training Set and Test Set

After cleaning the data, 87898 observations can be regarded as the total dataset for our research. In this case, 80% observations were selected as training set by using simple random sampling method. In other words, the rest 20% observations were test set.

	TRAINING SET	TEST SET
Percentage	**80%**	**20%**
Number of Observation	**70318**	**17580**
Proportion of Squad K/D Class 1	**34.22%**	**34.58%**
Proportion of Squad K/D Class 2	**32.89%**	**33.01%**
Proportion of Squad K/D Class 3	**18.12%**	**17.83%**
Proportion of Squad K/D Class 4	**11.20%**	**11.40%**
Proportion of Squad K/D Class 5	**3.57%**	**3.58%**

Table 2.5 Basic Attribute of Training Set and Test Set

Based on table above, the proportions of each KillDeathRatio class in both training set and test set are approximate after setting the seed of simple random sampling equal to 1. The training set and test set are reasonably determined in the research by using *set.seed(1)*.

3 Data Visualization

3.1 Bar Chart

The average WinRatio of different game classification is measured and showed as figure below. Compared to the average WinRatio of overall game classifications approximating to 4.49%, average WinRatio of Squad (5.71%) is higher than the overall average, while average WinRatio of Duo (2.35%) and Solo (2.62%) is less.

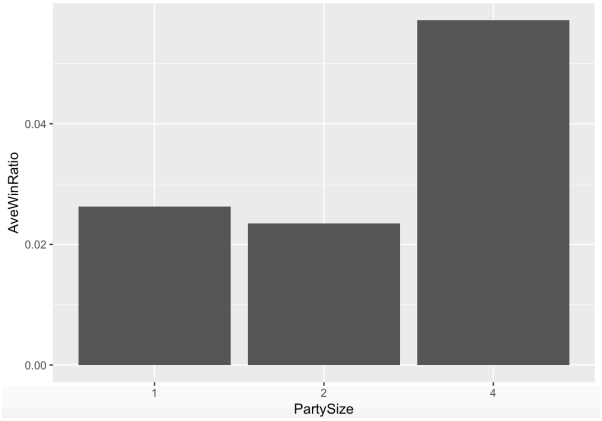

Figure 3.1 Bar Chart of WinRatio in Different Party Size

In other words, it is twice likely to win the game when players prefer to join a four-person team rather than play along or organize a two-person team. A crew of four persons can be a good strategy to raise the WinRatio in this game.

3.2 Box Plot

Box plot of each variables are performed as a convenient way of visually displaying the data distribution through their quartiles. In the Solo Party Size as Figure 3.2.1, the distribution of variable *KillDeathRatio* is skewed to right, with nearly 10,000 right-side outliers accounting for less than 13% of the total data set. This phenomenon also indicates that the majority of players belongs to the range of KillDeathRatio Class 1 and KillDeathRatio Class 2.

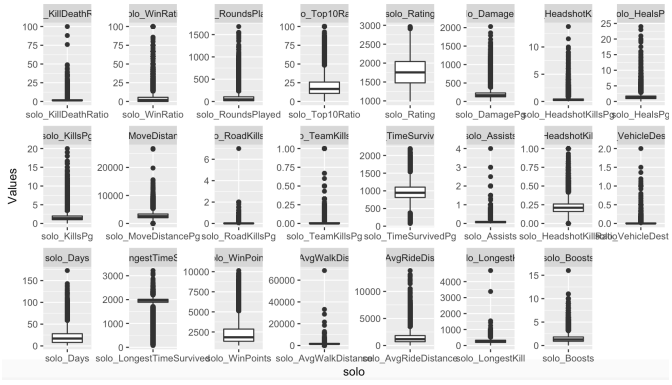

Figure 3.2.1 Box Plot of Variables of Solo Dataset

When it comes to the box plot of variables in party size of two persons as Figure 3.2.2, the distribution of variable *KillDeathRatio* is skewed to right more dramatically, with more right-side

outliers accounting for nearly 15% of the total data set. The distribution of variable Average Survival Time is symmetric, and the upper fence of outlier is 1000 minutes, with less than 5% observation belongs to outliers.

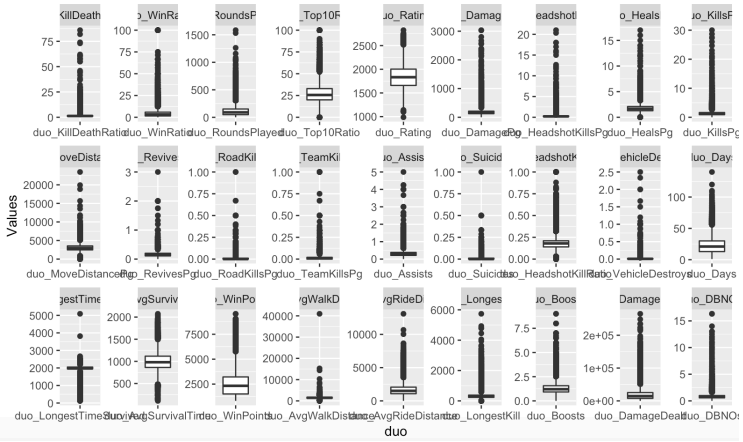

Figure 3.2.2 Box Plot of Variables of Duo Dataset

Comparing the box plot of variables in party size of two persons as Figure 3.2.3, the distribution of variable *KillDeathRatio* is skewed to right more dramatically, with more right-side outliers accounting for nearly 15% of the total data set. The distribution of variable Average Survival Time is symmetric, and the upper fence of outlier is 1200 minutes, with merely 5% observation belongs to outliers.

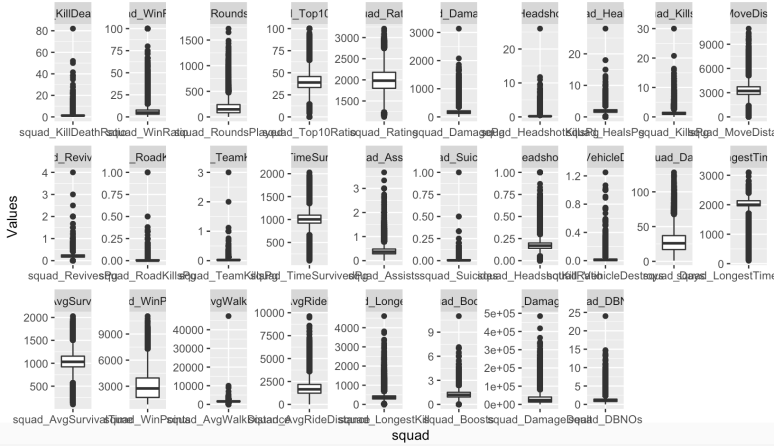

Figure 3.2.3 Box Plot of Variables of Squad Dataset

3.3 Correlogram

The correlogram of each variables in Solo Party Size can be obtain directly by using corrplot( ) in R. As shown in Figure 3.3.1, the variables RoadKillsPg, TeamKillsPg and VehicleDestroys visually illustrates the sight correlation among other variables. These two variables should be removed as a logic way of data reduction.

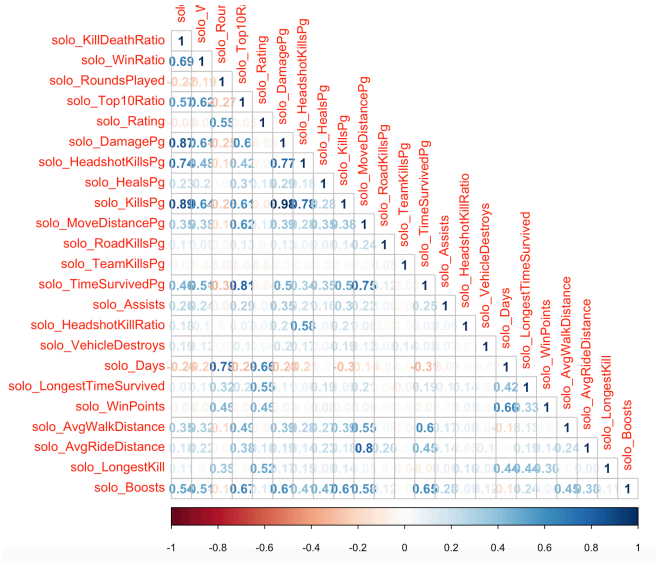

Figure 3.3.1 Correlogram of Each Variables in Solo Party Size

Based on correlogram of Duo Party Size, the correlation among RoadKillsPg, TeamKillsPg, Suicides and other variables approximate to 0. Therefore these three variables are removed and not involved in the furthermore analysis process.

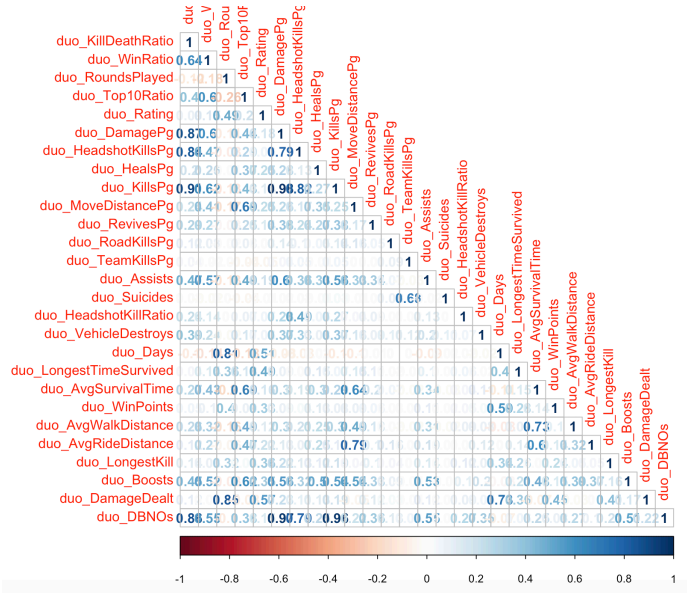

Figure 3.3.2 Correlogram of Each Variables in Duo Party Size

Similarly, the variables RoadKillsPg, TeamKillsPg, Suicides are weakly correlated to other variables, so should be removed.

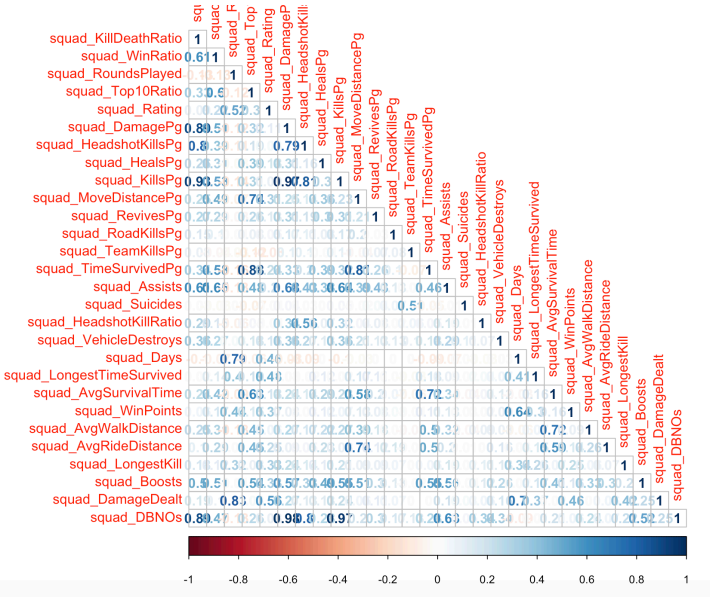

Figure 3.3.3 Correlogram of Each Variables in Squad Party Size

Meanwhile, the variables of Rating, LongestTimeSurvived and WinRatio seem less likely to correlate to KillDeathRatio in all party size, which is our target variable in supervised learning model. Therefore, these additional variables should be removed and not involved in the supervise learning model.

4 Dimension Reduction

In statistics, dimension reduction is the process of reducing the number of random variables under consideration by obtaining a set of principal variables. Feature projection transforms the data in the high-dimensional space to a space of fewer dimensions. The data transformation may be linear, as **Principal Component Analysis (PCA)** in our case.

The main linear technique for dimension reduction, principal component analysis, performs a linear mapping of the data to a lower-dimensional space in such a way that the variance of the data in the low-dimensional representation is maximized. In practice, the covariance (and sometimes the correlation) matrix of the data is constructed and the eigenvectors on this matrix are computed. The eigenvectors that correspond to the largest eigenvalues (the principal components) can now be used to reconstruct a large fraction of the variance of the original data. Moreover, the first few eigenvectors can often be interpreted in terms of the large-scale physical behaviour of the system. The original space (with dimension of the number of points) has been reduced (with data loss, but hopefully retaining the most important variance) to the space spanned by a few eigenvectors.

4.1 Scaling and Standardizing Process

Before performing principal component analysis (PCA), all the variables are require to scaled and based on the regulation of measurement translation. Since variables in different party size

are with different unit of measurement. For example, some variables are measured in minutes, while some are measured in percentage.

For principal components analysis, variables measured on different scales or on a common scale with widely differing ranges are often standardized. Based on the box plot of variables in each party size, there exist outliers with different degree of deviation. Therefore, standardization is necessary after scaling process.

4.2 Principal Component Analysis of Solo Dataset

A principal component analysis has been performed on the standardized dataset containing 87898 observations with 20 Game Performance Index variables for Solo Dataset.

```
> summary(solo.pca)
Importance of components:
                          PC1    PC2    PC3     PC4     PC5     PC6     PC7     PC8     PC9
Standard deviation     2.6123 1.8932 1.4213 1.04398 0.93613 0.92335 0.87455 0.84517 0.79948
Proportion of Variance 0.3412 0.1792 0.1010 0.05449 0.04382 0.04263 0.03824 0.03572 0.03196
Cumulative Proportion  0.3412 0.5204 0.6214 0.67591 0.71973 0.76236 0.80060 0.83632 0.86828
                          PC10    PC11    PC12    PC13    PC14    PC15    PC16    PC17    PC18
Standard deviation     0.71947 0.64248 0.62888 0.58434 0.53925 0.41038 0.38962 0.35952 0.34852
Proportion of Variance 0.02588 0.02064 0.01977 0.01707 0.01454 0.00842 0.00759 0.00646 0.00607
Cumulative Proportion  0.89416 0.91480 0.93457 0.95164 0.96618 0.97460 0.98219 0.98866 0.99473
                          PC19    PC20
Standard deviation     0.30301 0.11650
Proportion of Variance 0.00459 0.00068
Cumulative Proportion  0.99932 1.00000
```

Figure 4.2.1 Principal Component of Solo Dataset

The numerical result of PCA has indicate the first 5 principal component, which explains nearly 72% of the total variance among solo players. Through scree plot in Figure 4.2.2, it seems not much explanation of variance is gained after 5th principal component, therefore it is reasonable to select five as the number of PC to investigate.

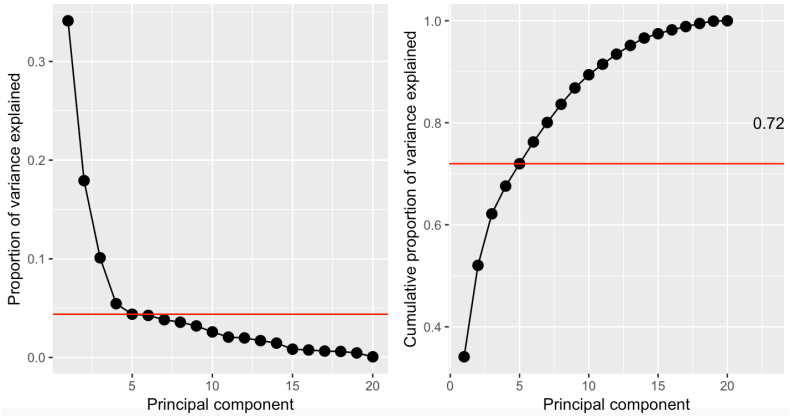

Figure 4.2.2 Scree Plot of Principal Component and Cumulative Sum of Solo Dataset

To get an overall pattern on how the 21 game index variables contribute to the first 11 principal components, which explains 91.48% of variation, a heat map has been generated as shown in

Figure 4.2.3. It is noted that the first principal component (PC1) consists of opposing loadings among specific game performance index variables.

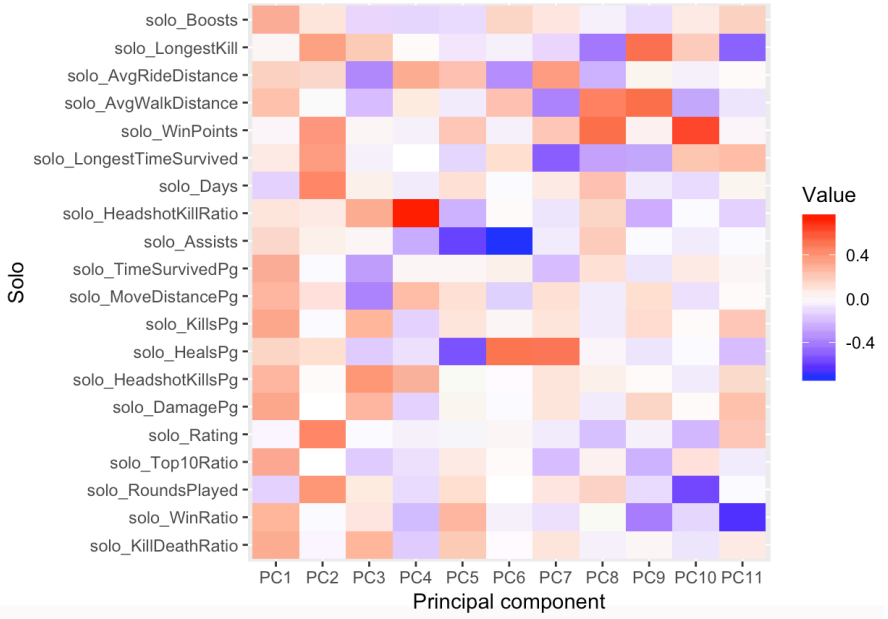

Figure 4.2.3 Heat Map of Game Index and Principal Components of Solo Dataset

A full spectrum of the variable loadings in the first 5 principal components are provided below. As noted from the above heat map, not all the first principal component is marked by same signs. The time-relative variables RoundsPlayed and Days have opposite signs with other game performance index variables.

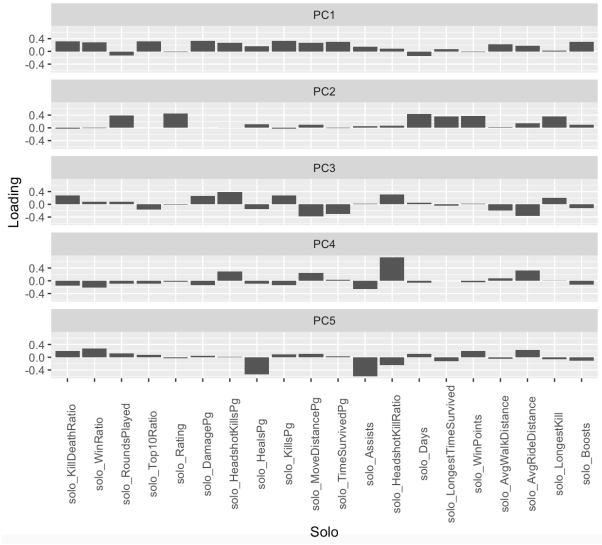

Figure 4.2.4 Full Spectrum of Variables Loadings in the first 5 PC of Solo Dataset

According to the theory of selecting performance index variables based on their contribution to the principal component. Based on the scree plot analysis, the 'variable importance' values were summed up to 5th principal component. The results have been reordered in descending order and summarized in Figure 4.2.5.

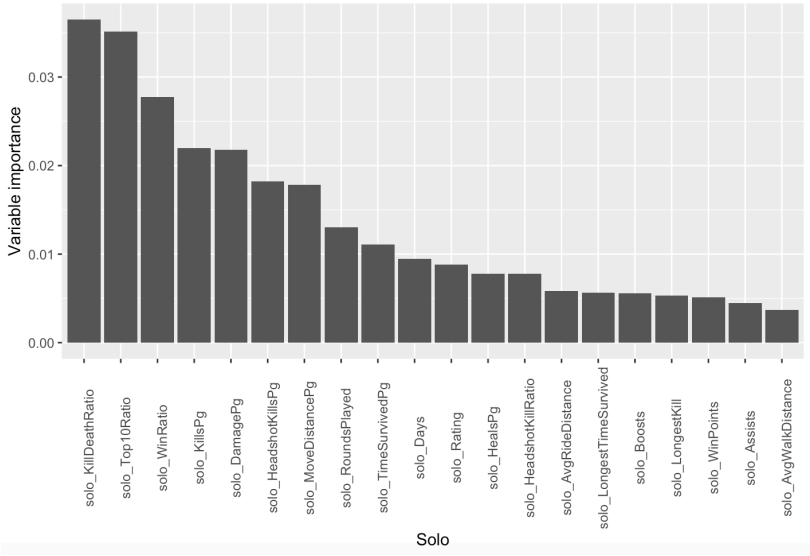

Figure 4.2.5 Rank of Variable importance for the first 5 PC of Solo Dataset

Also, the top 10 most influential variable of Solo Dataset on the first 5 principal components can be observed in the following table.

```
> (top10<-rownames(melt_NI)[order(-melt_NI$value)[1:10]])
 [1] "solo_KillDeathRatio"  "solo_Top10Ratio"      "solo_WinRatio"        "solo_KillsPg"         "solo_DamagePg"
 [6] "solo_HeadshotKillsPg" "solo_MoveDistancePg"  "solo_RoundsPlayed"    "solo_TimeSurvivedPg"  "solo_Days"
```

Table 4.2.5 Top 10 Most Influential Variables of Solo Dataset

From the above Principal Component Analysis of Solo Dataset, we can find out the top 5 game performance index variables as KillDeathRatio, Top10Ratio, WinRatio, KillsPerGame, DamagePerGame, based on their contribution to variance with the first 5 principal components, which can explain approximately 72% of the total variance. The content of these 5 game performance index variables may vary from different players in Solo Dataset.

4.3 Principal Component Analysis of Duo Dataset

A principal component analysis has been performed on the standardized dataset containing 87898 observations with 24 Game Performance Index (variables) for Duo Dataset.

```
> summary(duo.pca)
Importance of components:
                          PC1    PC2    PC3     PC4     PC5     PC6     PC7     PC8
Standard deviation     2.7490 1.9631 1.7023 1.06745 0.99785 0.93455 0.89776 0.87264
Proportion of Variance 0.3149 0.1606 0.1207 0.04748 0.04149 0.03639 0.03358 0.03173
Cumulative Proportion  0.3149 0.4755 0.5962 0.64369 0.68518 0.72157 0.75515 0.78688
                          PC9    PC10    PC11    PC12    PC13    PC14    PC15    PC16
Standard deviation     0.85751 0.83360 0.82221 0.75728 0.65575 0.64722 0.60239 0.54633
Proportion of Variance 0.03064 0.02895 0.02817 0.02389 0.01792 0.01745 0.01512 0.01244
Cumulative Proportion  0.81752 0.84647 0.87464 0.89854 0.91645 0.93391 0.94903 0.96146
                          PC17    PC18    PC19    PC20    PC21    PC22    PC23    PC24
Standard deviation     0.49498 0.45451 0.40644 0.30710 0.30362 0.26718 0.17880 0.13498
Proportion of Variance 0.01021 0.00861 0.00688 0.00393 0.00384 0.00297 0.00133 0.00076
Cumulative Proportion  0.97167 0.98028 0.98716 0.99109 0.99493 0.99791 0.99924 1.00000
```

Figure 4.3.1 Principal Component of Duo Dataset

The numerical result of PCA has indicate the first 6 principal component, which explains nearly 72.16% of the total variance among solo players. Through scree plot in Figure 4.3.2, it seems not much explanation of variance is gained after 6th principal component, therefore it is reasonable to select six as the number of PC to investigate.

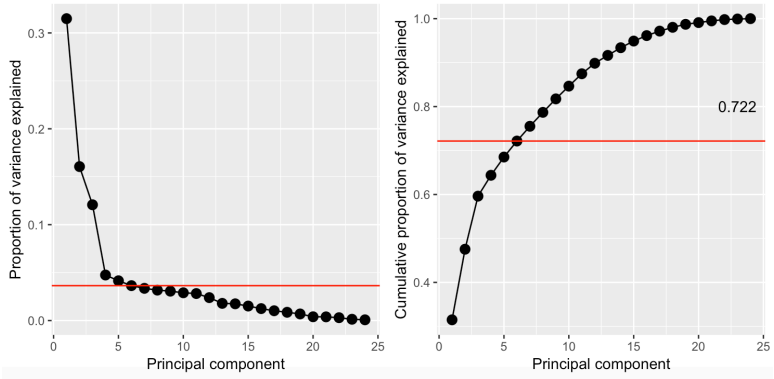

Figure 4.3.2 Scree Plot of Principal Component and Cumulative Sum of Duo Dataset

To get an overall pattern on how the 24 game index variables contribute to the first 12 principal components, which explains 89.85% of variation, a heat map has been generated.

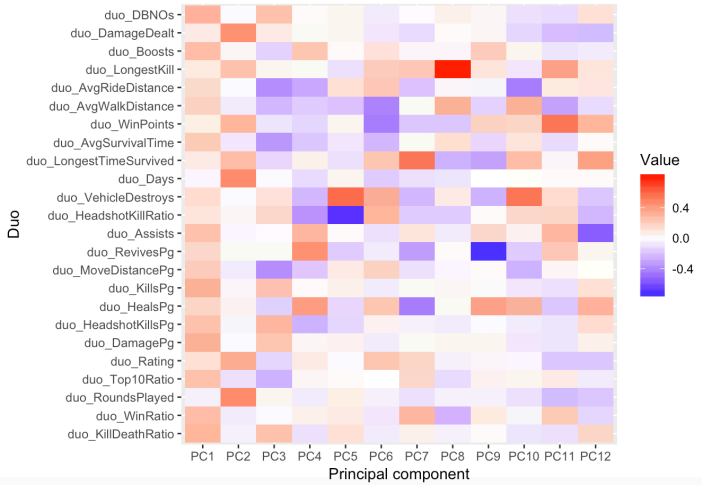

Figure 4.3.3 Heat Map of Game Index and Principal Components of Duo Dataset

A full spectrum of the variable loadings in the first 6 principal components are provided below. As noted from the above heat map, all the first principal component is marked by same signs.

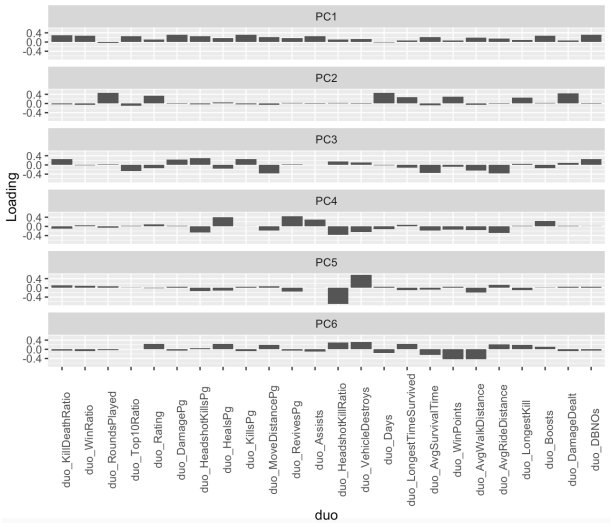

Figure 4.3.4 Full Spectrum of Variables Loadings in the first 6 PC of Duo Dataset

Based on the scree plot analysis, the 'variable importance' values were summed up to 6th principal component. The results have been reordered in descending order and summarized in Figure 4.3.5.

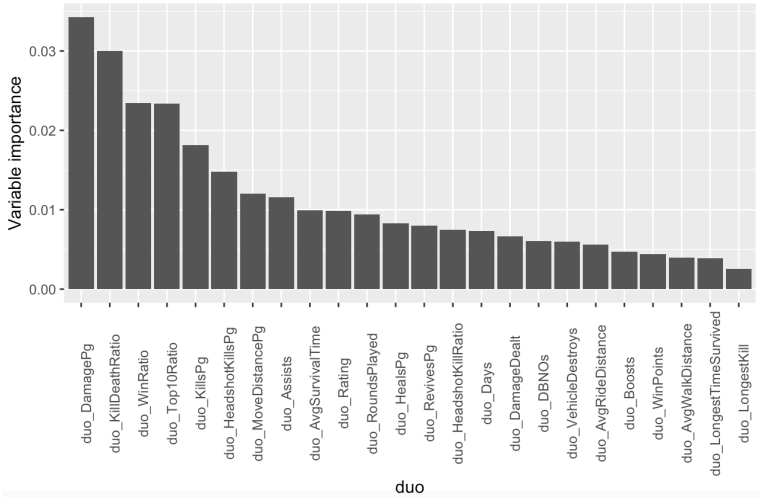

Figure 4.3.5 Rank of Variable importance for the first 6 PC of Duo Dataset

Also, the top 10 most influential variable of Duo Dataset on the first 6 principal components can be observed in the following table.

```
> (top10<-rownames(melt_NI)[order(-melt_NI$value)[1:10]])
 [1] "duo_DamagePg"        "duo_KillDeathRatio"  "duo_WinRatio"        "duo_Top10Ratio"      "duo_KillsPg"
 [6] "duo_HeadshotKillsPg" "duo_MoveDistancePg"  "duo_Assists"         "duo_AvgSurvivalTime" "duo_Rating"
```

Table 4.3.5 Top 10 Most Influential Variables of Duo Dataset

From the above Principal Component Analysis of Duo Dataset, we can find out the top 5 game performance index variables as DamagePerGame, KillDeathRatio, WinRatio, Top10Ratio, KillsPerGame, based on their contribution to variance with the first 6 principal components, which can explain approximately 72.16% of the total variance. The content of these 5 game performance index variables may vary from different players in Duo Dataset.

4.4 Principal Component Analysis of Squad Dataset

A principal component analysis has been performed on the standardized dataset containing 87898 observations with 25 Game Performance Index (variables) using prcomp( ) for Squad Dataset.

```
> summary(squad.pca)
Importance of components:
                          PC1    PC2    PC3     PC4     PC5    PC6     PC7     PC8     PC9   PC10
Standard deviation     2.8225 1.9615 1.8103 1.06675 1.00275 0.9552 0.89316 0.88232 0.87515 0.8306
Proportion of Variance 0.3186 0.1539 0.1311 0.04552 0.04022 0.0365 0.03191 0.03114 0.03064 0.0276
Cumulative Proportion  0.3186 0.4725 0.6036 0.64915 0.68937 0.7259 0.75777 0.78891 0.81955 0.8471
                          PC11    PC12    PC13    PC14    PC15    PC16    PC17    PC18    PC19
Standard deviation     0.82596 0.74919 0.6709 0.65536 0.62562 0.56486 0.49639 0.4389 0.39903
Proportion of Variance 0.02729 0.02245 0.0180 0.01718 0.01566 0.01276 0.00986 0.0077 0.00637
Cumulative Proportion  0.87443 0.89689 0.9149 0.93207 0.94772 0.96049 0.97034 0.9780 0.98442
                          PC20    PC21    PC22    PC23    PC24    PC25
Standard deviation     0.36520 0.31343 0.25641 0.22817 0.15448 0.12785
Proportion of Variance 0.00533 0.00393 0.00263 0.00208 0.00095 0.00065
Cumulative Proportion  0.98975 0.99368 0.99631 0.99839 0.99935 1.00000
```

Figure 4.4.1 Principal Component of Squad Dataset

The numerical result of PCA has indicate the first 6 principal component, which explains nearly 72.59% of the total variance among solo players. Through scree plot in Figure 4.4.2, it seems not much explanation of variance is gained after 6th principal component, therefore it is reasonable to select six as the number of PC to investigate.

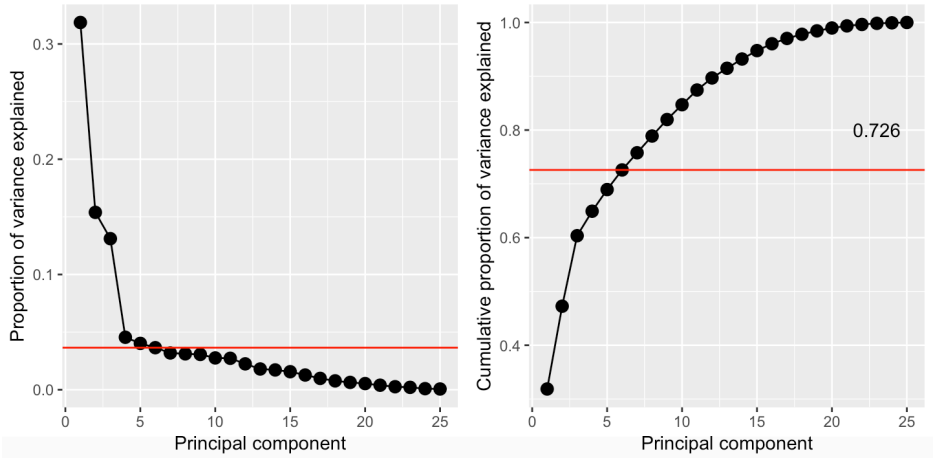

Figure 4.4.2 Scree Plot of Principal Component and Cumulative Sum of Squad Dataset

To get an overall pattern on how the 25 game index variables contribute to the first 12 principal components, which explains 89.69% of variation, a heat map has been generated.

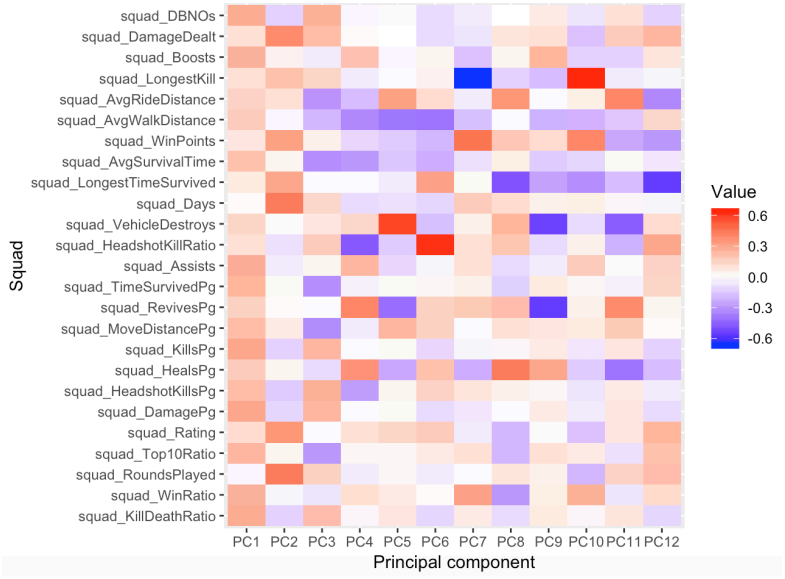

Figure 4.4.3 Heat Map of Game Index and Principal Components of Squad Dataset

A full spectrum of the variable loadings in the first 6 principal components are provided below. As noted from the above heat map, all the first principal component is marked by same signs.

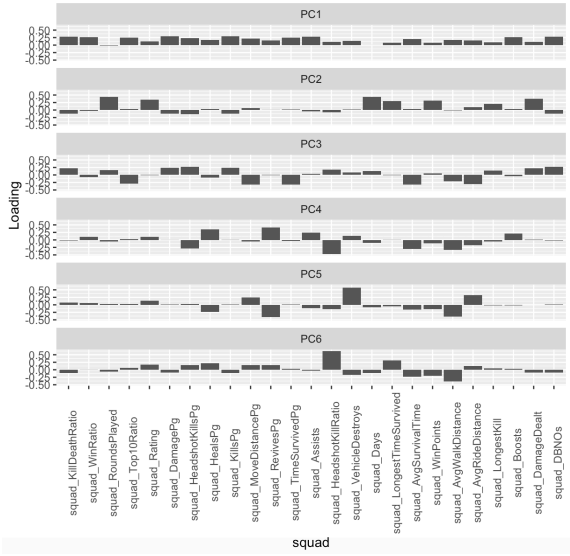

Figure 4.4.4 Full Spectrum of Variables Loadings in the first 6 PC of Squad Dataset

Based on the scree plot analysis, the 'variable importance' values were summed up to 6th principal component. The results have been reordered in descending order and summarized.

```
> (top10<-rownames(melt_NI)[order(-melt_NI$value)[1:10]])
 [1] "squad_DamagePg"        "squad_KillDeathRatio"  "squad_WinRatio"        "squad_Top10Ratio"      "squad_KillsPg"
 [6] "squad_TimeSurvivedPg"  "squad_HeadshotKillsPg" "squad_MoveDistancePg"  "squad_Assists"         "squad_Rating"
```

Table 4.4.5 Top 10 Most Influential Variables of Squad Dataset

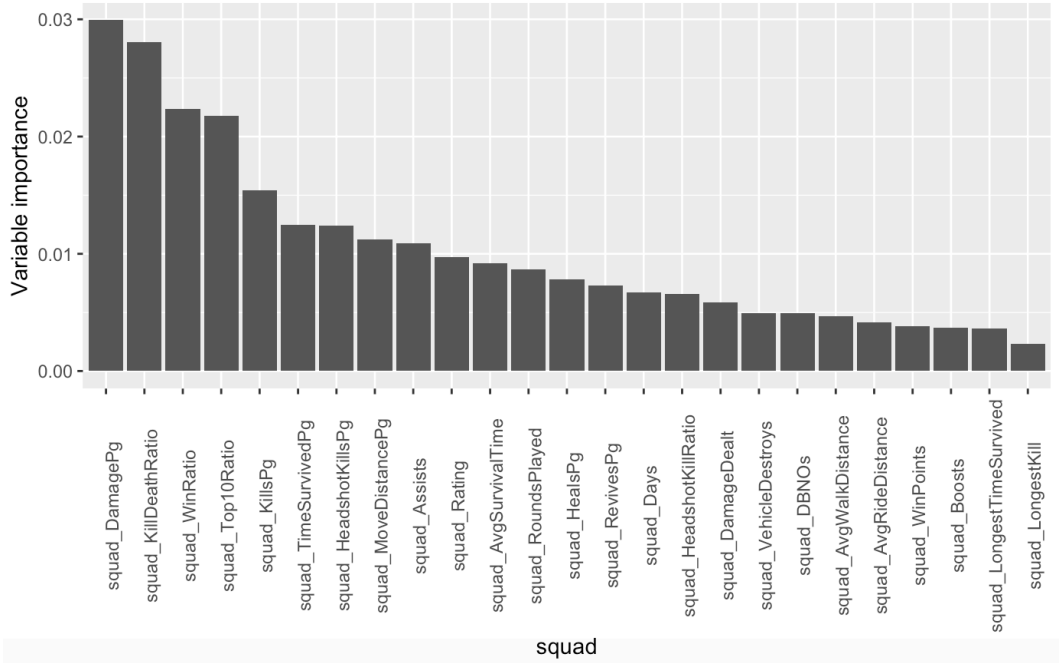

Figure 4.4.5 Rank of Variable importance for the first 6 PC of Squad Dataset

From the above Principal Component Analysis of Squad Dataset, we can find out the top 5 game performance index variables as DamagePerGame, KillDeathRatio, WinRatio, Top10Ratio, KillsPerGame, based on their contribution to variance with the first 6 principal components, which can explain approximately 72.59% of the total variance. The content of these 5 game performance index variables may vary from different players in Squad Dataset.

5 Unsupervised Learning

Clustering is the task of grouping a set of objects in such a way that objects in the same group (called 'one cluster') are more similar, in some sense, to each other than to those in other groups (clusters). This part is to tell the relationship between any two variables in each clusters based on the results of PCA.

5.1 Clustering of Solo Dataset

The figure from last parts shows that the cumulative proportion of PC5 explains 72% (over 70%). So the first 5 components would be used to do the clustering.

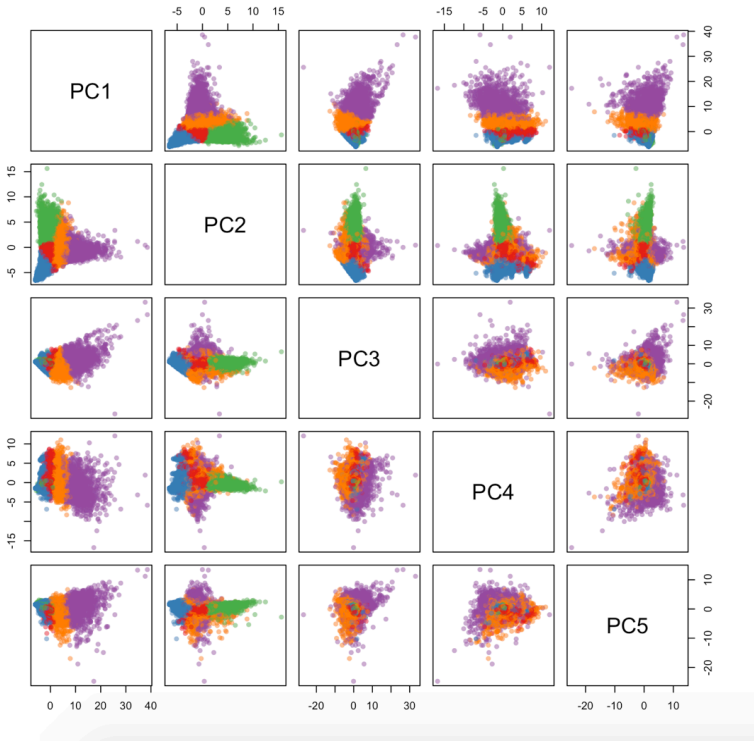

Figure 5.1.1 Clustering of Solo Dataset

The Figure 5.1.1 shows that 23 2-D projections of data which are in a 5-D space with five clusters. Each five clusters can be identified clearly when we see the relationship among PC1, PC2 and PC3. For example, five clusters with five colors are clear and there are few overlaps between PC1 and PC2. However, there are many overlaps among clusters when the relationship between PC4 and PC5 is figured.

Next a 3D visualization would be applied to analysis. The one exception to this rule is when the visualization is interactive, which allows the user to explore the space and not lose meaning due to three dimensions being collapsed into a 2D image.

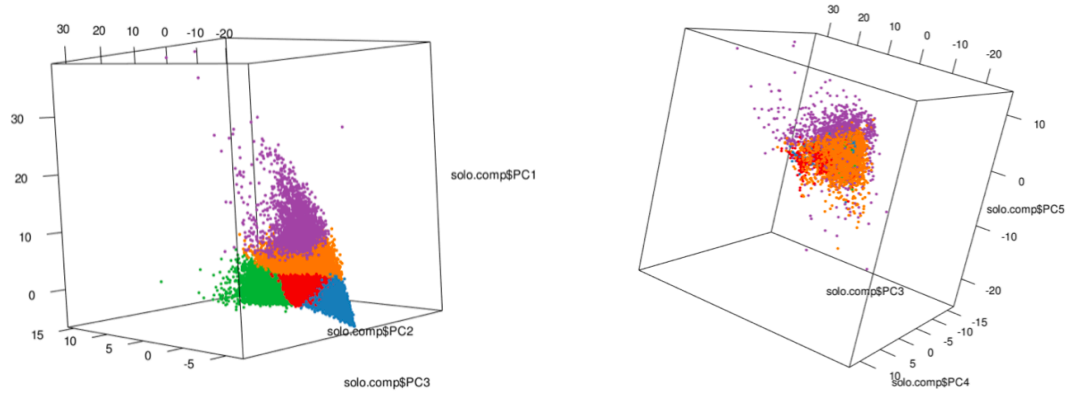

Figure 5.1.2 3D plot of clustering result on PC1~PC3 and PC3~PC5 of Solo Dataset

The figure above shows that data are confined mainly to the one plane on the left (components 1-3), with the exception of the outlier. There are five clear clusters in figure of left side. There is also bunching in the other dimensions (components 3,4,5) on right side, but clusters are not clear as the left one.

Then the relationship between each variable would be discussed based on five clusters.

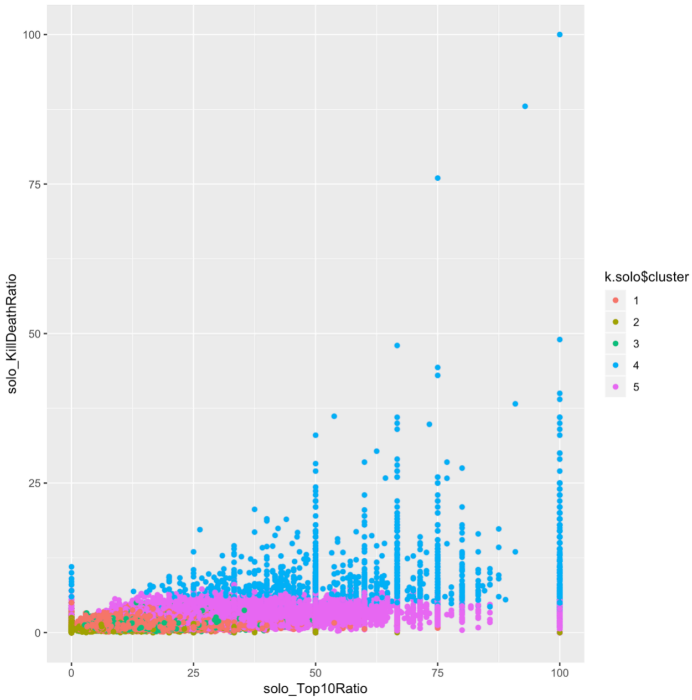

Figure 5.1.3 Distribution Between Top10Ratio and KillDeathRatio of Solo Dataset

The reason for choosing variables Top 10Ratio and Kill Death Ratio is that they belong to top10 most important variables based PCA. The figure above shows that when Top 10 ratio increases, K/D ratio keeps in a constant interval, except some outliers. For example, K/D ratio keeps from 0% to 10%, when Top 10 ratio increases from 0% to 85%(except some outliers) in purple cluster. Most people's K/D ratio are from 10% to 25%.

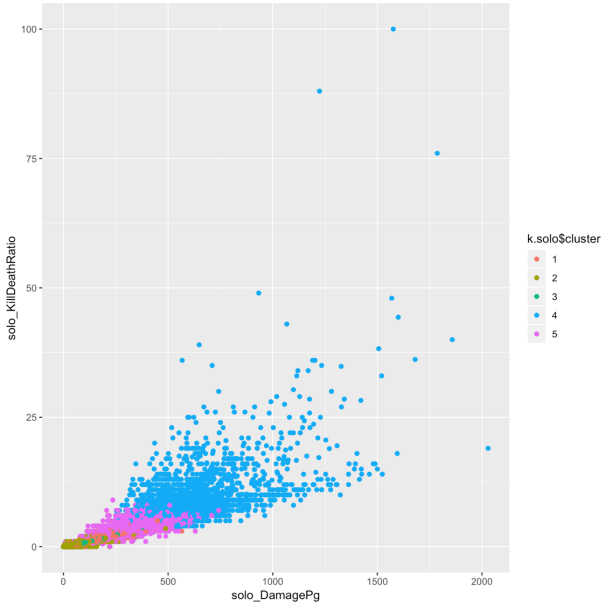

Figure 5.1.4 Distribution between DamagePerGame and KillDeathRatio of Solo Dataset

The reason for choosing these two variables is the same as above. The figure shows that when DamagePg increases, K/D ratio increases, except some outliers. For example, K/D ratio goes up f from 10% to 25%, when DamagePg increases from 250 to 1500 in blue cluster. Most people are in blue cluster.

5.2 Clustering of Duo Dataset

The figure from last parts shows that the cumulative proportion of PC6 explains 72.16%. So six components would be used to do the clustering.

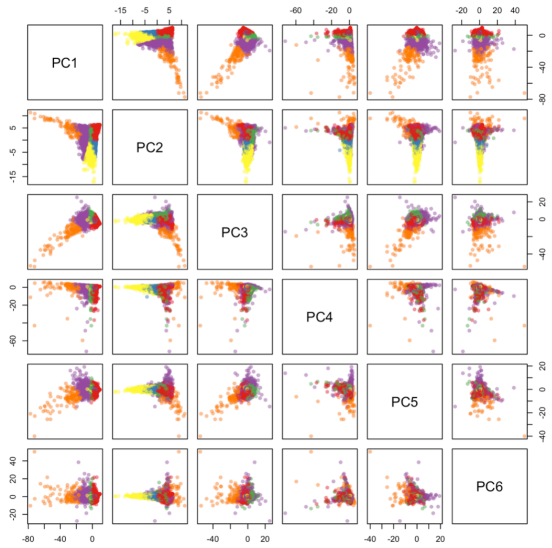

Figure 5.2.1 Clustering of Duo Dataset

The Figure 5.2.1 shows that 27 2-D projections of data which are in a 5-D space with six clusters. Each six clusters can be identified clearly when we see the relationship among PC1, PC2 and PC3. For example, six clusters with five colors are clear and there are few overlaps between PC1 and PC2. However, there are many overlaps among clusters when the relationship among PC4 ,PC5 and PC6 is figured.

Next a 3D visualization would be applied to analysis. The one exception to this rule is when the visualization is interactive, which allows the user to explore the space and not lose meaning due to three dimensions being collapsed
into a 2D image.

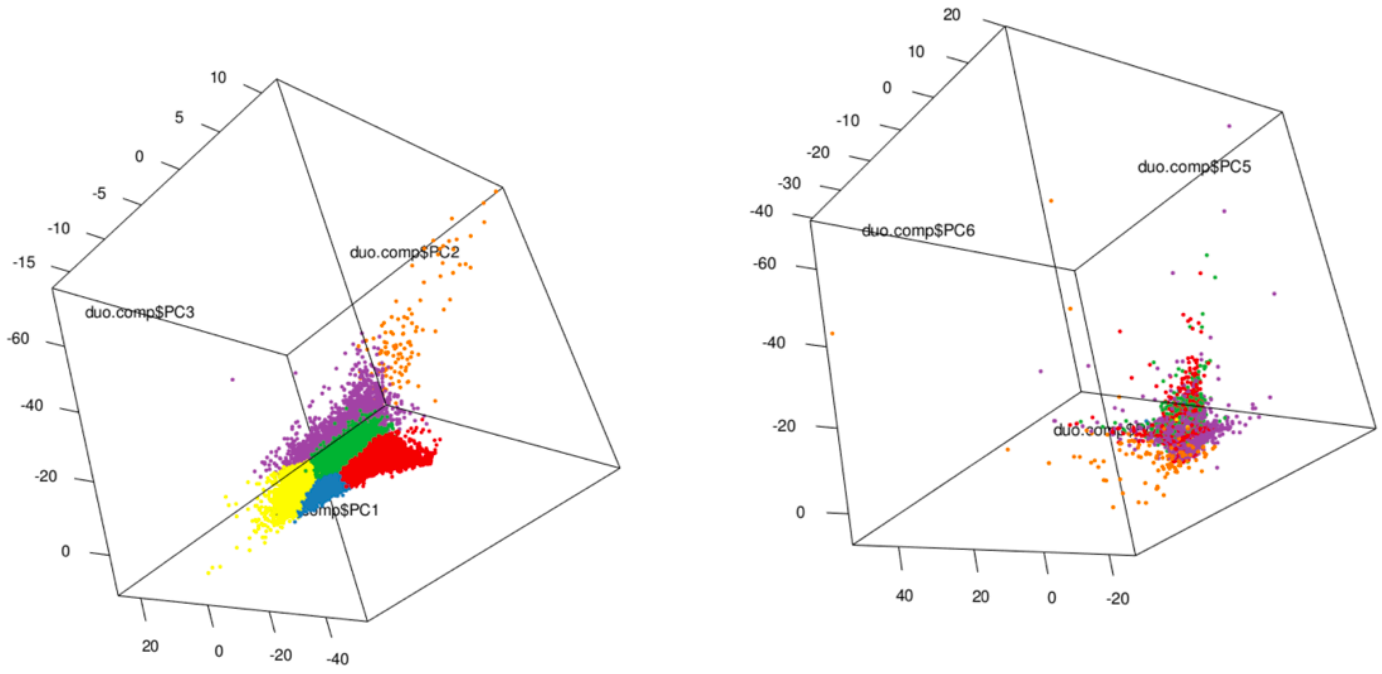

Figure 5.2.2 3D plot of clustering result on PC1~PC3 and PC4~PC6 of Duo Dataset

The figure above shows that data are confined mainly to the one plane on the left (components 1-3), with the exception of the outlier. There are clear clusters in the left figure.There is also bunching in the other dimensions (components 4,5,6) on right side. But clusters are not clear.

Then the relationship between each variable would be discussed based on six clusters.

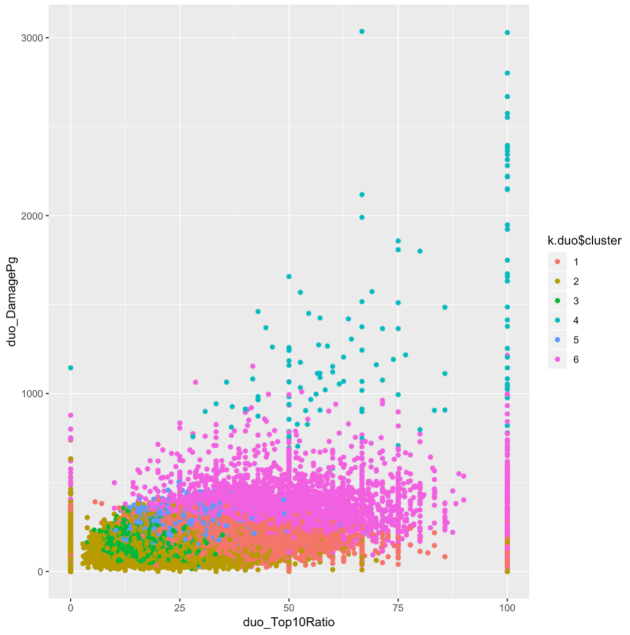

Figure 5.2.3 Distribution Between Top10Ratio and DamagePerGame of Duo Dataset

The reason for choosing variables Top 10Ratio and DamagePg is that they belong to top10 most important variables based PCA. The figure above shows that when Top 10 ratio increases, DamagePg keeps in a constant interval, except some outliers. For example, DamagePg keeps from 500 to 600, when Top 10 ratio increases from 15% to 90% (except outliers) in purple cluster. Most people's DamagePg are from 500 to 600.

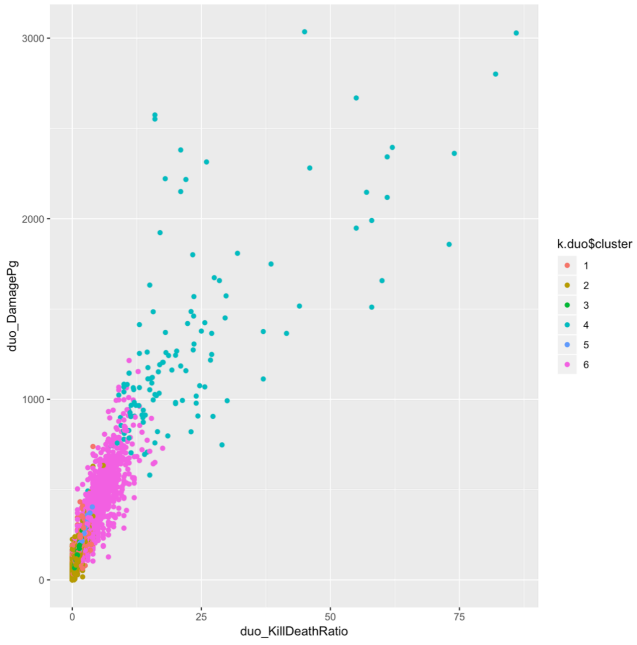

Figure 5.2.4 Distribution Between DamagePerGame and KillDeathRatio of Duo Dataset

The reason for choosing these two variables is the same as above. The figure shows that when DamagePg increases, K/D ratio increases, except some outliers. For example, K/D ratio goes up f from 2% to 12.5%, when DamagePg increases from 100 to 1000 in blue cluster. Most people are in purple cluster.

5.3 Clustering of Squad Dataset

The figure from last parts shows that the cumulative proportion of PC6 explains 72.59%. So six components would be used to do the clustering.

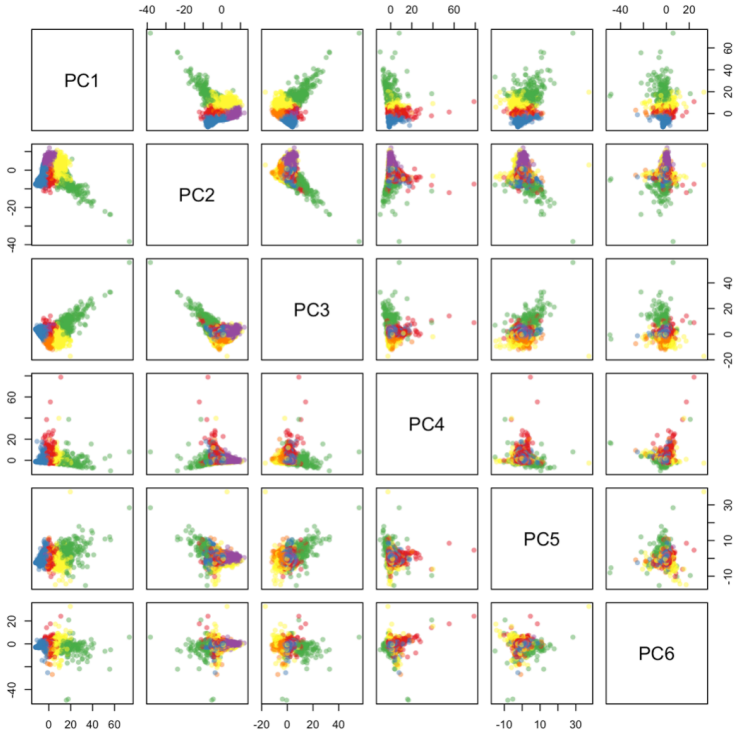

Figure 5.3.1 Clustering of Squad Dataset

The Figure 5.3.1 shows that 27 2-D projections of data which are in a 5-D space with six clusters. Each six clusters can be identified clearly when we see the relationship among PC1, PC2 and PC3. For example, six clusters with five colors are clear and there are few overlaps between PC1 and PC2. However, there are many overlaps among clusters when the relationship among PC4 ,PC5 and PC6 is figured.

Next a 3D visualization would be applied to analysis. The one exception to this rule is when the visualization is interactive, which allows the user to explore the space and not lose meaning due to three dimensions being collapsed into a 2D image.

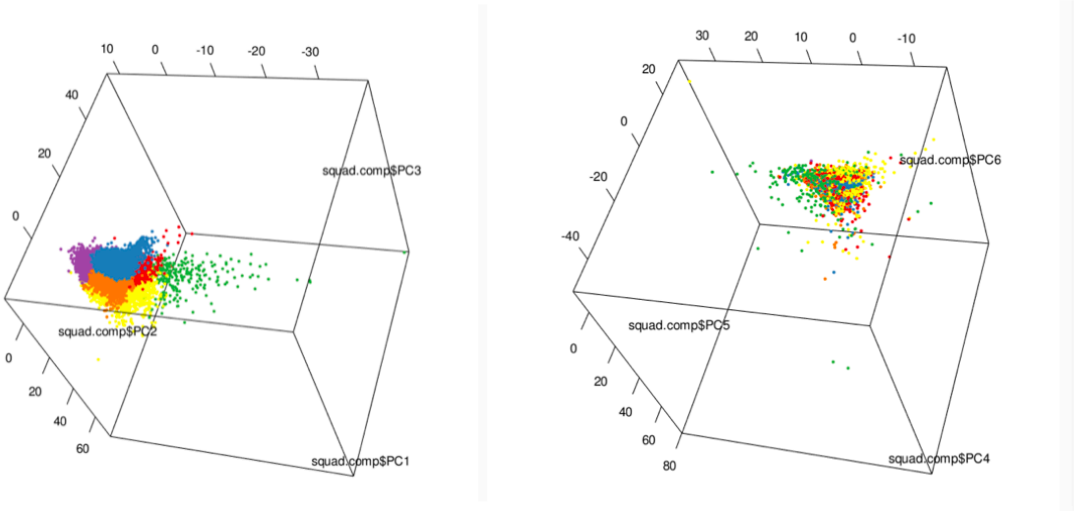

Figure 5.3.2 3D plot of clustering result on PC1~PC3 and PC4~PC6 of Squad Dataset

The figure above shows that data are confined mainly to the one plane on the left (components 1-3), with the exception of the outlier. There are clear clusters in the left figure.There is also bunching in the other dimensions (components 4,5,6) on right side. But clusters are not clear.

Then the relationship between each variable would be discussed based on six clusters.

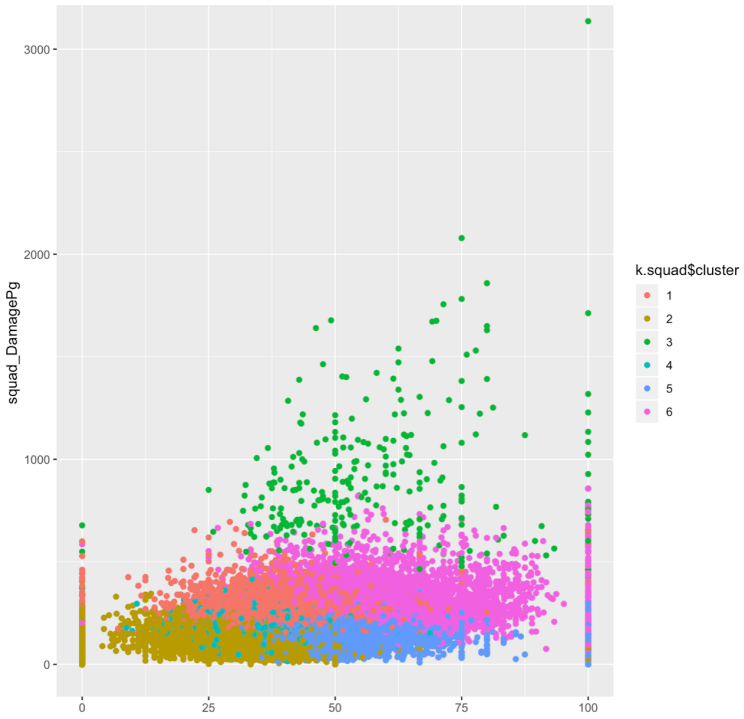

Figure 5.3.3 Distribution Between Top10Ratio and DamagePerGame of Squad Dataset

The reason for choosing variables Top 10Ratio and DamagePg is that they belong to top10 most important variables based PCA. The figure above shows that when Top 10 ratio increases, DamagePg keeps in a constant interval, except some outliers. For example, DamagePg keeps from 200 to 600, when Top 10 ratio increases from 15% to 90% (except outliers) in purple cluster. Most people's DamagePg are from 200 to 600.

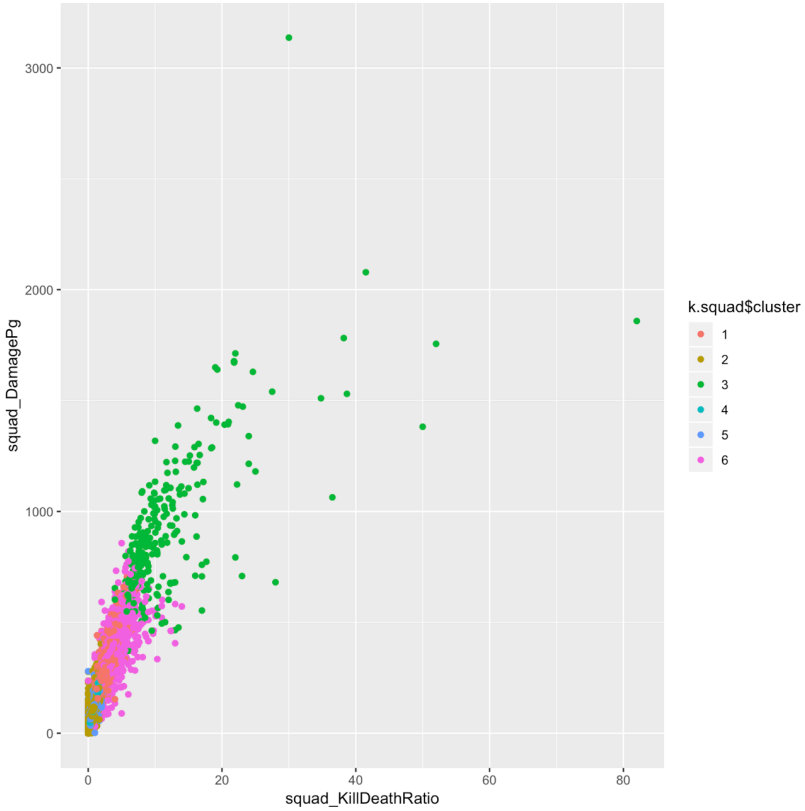

Figure 5.3.4 Distribution Between DamagePerGame and KillDeathRatio of Squad Dataset

The reason for choosing these two variables is the same as above. The figure shows that when DamagePg increases, K/D ratio increases, except some outliers. For example, K/D ratio goes up f from 2% to 10%, when DamagePg increases from 100 to 600 in blue cluster. Most people are in purple cluster.

5.4 Conclusion on Unsupervised Learning

Only solo has five clusters. Both duo and squad have six clusters. The results of duo and squad are very similar. The little difference is that the K/D ratio and DamagePg of players with duo are higher than those of players with squad.

6 Supervised Learning

6.1 Data Preparation

6.1.1 Classification Introduction

We will classify the players in terms of the variable KillDeathRatio, which is the most important value to evaluate the level of one's comprehensive ability in PUBG. The following formula is about how to compute it.

$$K/D = \frac{kill}{death}$$

According to the value of K/D, the level of the players and the number of classification in this research are shown in table below.

Range	Level	Classification
K/D<1	Beginners	1
1<K/D<1.5	General Players	2
1.5<K/D<2	Senior Players	3
2<K/D<3	Master Players	4
K/D>3	Super Master Players	5

Table 6.1.1 K/D Classification

6.1.2 Correlation analysis

To classify the PUBG players by K/D. Deleted the variables that have extremely low correlation coefficients, whose absolute values are smaller than 0.1, with K/D by correlograms below. For the data of all three group size. we deleted the same 3 variables: Rating, LongestSurviveTime and WinPoints. Take Solo as an example, from figure below we can observe that the correlation coefficients between K/D and Rating, K/D and LongestSurviveTime, K/D and WinPoints is smaller than 0.1.

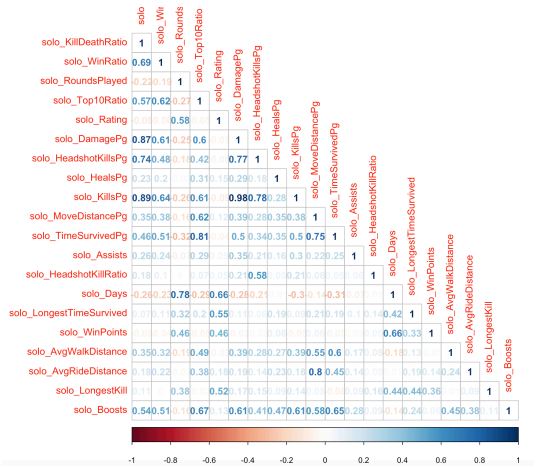

Figure 6.1.2 Correlogram of Solo Dataset

6.1.3 Data Slicing

Split each of the 3 datasets (Solo, Duo and Squad) into a training set with 80% observations and a test set with the rest 20% observations by using simple random sampling. The following three tables show the summary of the distribution of K/D classification in training set and test set.

Classification	Training Set	Test Set
1	16614	4109
2	20095	5078
3	13959	3520
4	11884	2959
5	7766	1914
Total	70318	17580

Table 6.1.3.1 K/D Ratio Summary of Solo Dataset

Classification	Training Set	Test Set
1	22066	5510
2	23508	6006
3	13382	3367
4	8393	1981
5	2969	716
Total	70318	17580

Table 6.1.3.2 K/D Ratio Summary of Duo Dataset

Classification	Training Set	Test Set
1	24066	6048
2	23125	5783
3	12739	3124
4	7876	1998
5	2512	627
Total	70318	17580

Table 6.1.3.3 K/D Ratio Summary of Squad Dataset

6.2 Training SVM Classifier using Non-Linear Kernel

Build models using Non-Linear Kernel with the Radial Basis Function. In Radial kernel, it needs to select proper value of Cost “C” parameter and “Gamma” parameter to obtain the best Support Vector Machine (SVM) model.

6.2.1 SVM Model of Solo Dataset

Using the default parameters C=1 and Gamma=0.0625, we obtained the following Confused matrix in Table 6.2.1.1 with an high accuracy 97.75%.

Solo	**Truth**				
Predict	1	2	3	4	5
1	**16335**	277	5	0	0
2	277	**19600**	315	0	0
3	2	217	**13520**	199	0
4	0	1	118	**11641**	124
5	0	0	1	44	**7642**

Table 6.2.1.1 Confused Matrix by Default of Solo Dataset

After tuning our classifier with different values of C and Gamma, we found that the best parameters are C=5 and Gamma=0.5. The result of the classification is in table below with an accuracy 99.82%. It is clear that the accuracy of this prediction after tuning is higher than before.

Solo	**Truth**				
Predict	1	2	3	4	5
1	**16577**	30	0	0	0
2	37	**20044**	25	0	0
3	0	21	**13926**	5	0
4	0	0	8	**11879**	0
5	0	0	0	0	**7766**

Table 6.2.1.2 Confused Matrix after Tuning of Solo Dataset

6.2.2 SVM Model of Duo Dataset

Using the default parameters C=1 and Gamma=0.0625, we got the following Confused matrix in table below with an high accuracy 97.66%.

Duo	Truth				
Predict	1	2	3	4	5
1	**21719**	331	2	0	0
2	347	**22970**	305	0	0
3	0	206	**12949**	198	0
4	0	1	126	**8153**	88
5	0	0	0	42	**2881**

Table 6.2.2.1 Confused Matrix by Default of Duo Dataset

Then, after tuning our classifier with different values of C and Gamma, we found that the best parameters are C=5 and Gamma=0.5. The result of the classification is in Table 6.2.2.2 with an accuracy 99.98%. Also, the accuracy of this prediction after tuning is higher than before.

Duo	Truth				
Predict	1	2	3	4	5
1	**22063**	11	0	0	0
2	3	**23496**	1	0	0
3	0	1	**13381**	0	0
4	0	1	0	**8393**	0
5	0	0	0	0	**2969**

Table 6.2.2.2 Confused Matrix after Tuning of Duo Dataset

6.2.3 SVM Model of Squad Dataset

Using the default parameters C=1 and Gamma=0.0625, obtain the following Confused matrix in Table 6.2.3.1 with an accuracy 97.77%.

Squad	Truth				
Predict	1	2	3	4	5
1	**23743**	320	2	0	0
2	323	**22586**	276	0	0
3	0	218	**12352**	215	0
4	0	1	109	**7640**	81
5	0	0	0	21	**2431**

Table 6.2.3.1 Confused Matrix by Default of Squad Dataset

After tuning our classifier with different values of C and Gamma, we found that the best parameters are C=10 and Gamma=0.5. The result of the classification is in table below with an accuracy 100%, which is perfect.

Squad	**Truth**				
Predict	1	2	3	4	5
1	**24066**	0	0	0	0
2	0	**23125**	0	0	0
3	0	0	**12739**	0	0
4	0	0	0	**7876**	0
5	0	0	0	0	**2512**

Table 6.2.3.2 Confused Matrix after Tuning of Squad Dataset

6.3 Prediction of Test Set

6.3.1 Prediction of Solo Test Set

Use the SVM trained model with C=5 and Gamma=0.5 that we had tuned above to predict the test set, it's giving an accuracy of 91.18%. It is clear that it is a good result.

Moreover, consider the accuracy for each classification. For the test set, the number of the observations for the 5 classifications are 4109, 5078, 3520, 2959 and 1914. Compute the values of the prediction accuracy for these 5 classifications, we obtained 93.79%, 91.81%, 87.84%, 87.10% and 96.39% respectively.

Solo	**Truth**				
Predict	1	2	3	4	5
1	**3854**	157	3	0	0
2	156	**4662**	190	0	0
3	1	189	**3092**	139	0
4	0	0	132	**2577**	69
5	98	70	103	243	**1845**

Table 6.3.1 Confused Matrix for Prediction of Solo Test Set

6.3.2 Prediction of Duo Test Set

Similarly, use the SVM trained model with C=5 and Gamma=0.5 that we had tuned above to predict the test set, it's giving an accuracy of 89.28%. Compare with that of Solo Test Set, this result is smaller than 90%, which is not so good.

Consider the accuracy for each classification. For the test set, the number of the observations for the 5 classifications are 5510, 6006, 3367, 1981 and 716. Compute the values of the prediction accuracy for these 5 classifications, we obtained 93.65%, 91.77%, 85.86%, 89.55% and 50.14% respectively.

Duo	**Truth**				
Predict	1	2	3	4	5
1	**5160**	218	1	0	0
2	274	**5512**	243	1	0
3	3	217	**2891**	176	0
4	73	59	232	**1774**	357
5	0	0	0	30	**359**

Table 6.3.2 Confused Matrix for Prediction of Duo Test Set

6.3.3 Prediction of Squad Test Set

Use the SVM trained model with C=10 and Gamma=0.5 that we had tuned above to predict the test set, it's giving an accuracy of 87.83%.

Furthermore, for the test set, the number of the observations for the 4 classifications are 6048, 5783, 3124, 1998 and 627. Similarly, compute the values of the prediction accuracy for these 5 classifications, we obtained 93.62%, 93.39%, 83.16%, 89.84% and 25.20% respectively.

Squad	**Truth**				
Predict	1	2	3	4	5
1	**5662**	238	1	0	0
2	296	**5227**	292	0	0
3	2	251	**2598**	199	0
4	88	67	233	**1795**	469
5	0	0	0	4	**158**

Table 6.3.3 Confused Matrix for Prediction of Squad Test Set

6.4 Conclusion on Supervised Learning

Combine the SVM prediction results of Solo, Duo and Squad Dataset into following table.

Comparison		Classification					
		1	2	3	4	5	Total
Solo	Accuracy	93.97%	91.81%	87.84%	87.10%	96.39%	91.18%
	Number of Obs	4109	5078	3520	2959	1914	17580
Duo	Accuracy	93.65%	91.77%	85.86%	89.55%	50.14%	89.28%
	Number of Obs	5510	6006	3367	1981	716	17580
Squad	Accuracy	93.62%	93.62%	83.16%	89.84%	25.20%	87.83%
	Number of Obs	6048	5783	3124	1998	627	17580

Table 6.4 Comparison of Prediction Accuracy

Firstly, compare the total accuracy of these 3 group sizes. Because of the same number of observations, we can conclude that if the group size is bigger, the accuracy will be lower. In other words, it is more and more difficult to predict the K/D level for a PUBG player as the team size going up.

Moreover, compare the 5 classifications' accuracy of these 3 group sizes. We can observe that for all of 3 group sizes, The K/D classification 1 and 2 with higher number of observations is easier to predict, maybe because most of the players are in the first 2 K/D classification, who are the Beginners and General players.

The K/D classification 3 and 4 with fewer observations have lower accuracies compared with classification 1 and 2, but they still have a good result.

Clearly, the K/D classification 5 is the most difficult to predict. It was dramatically effected by the group size. One of the reason is that the proportion of Super Master Players is actually very small, which will reduce the accuracy. Furthermore, we found that the group size is bigger, the prediction will need more observations in classification 5 in order to obtain a higher accuracy.

Consequently, the accuracy of predicting higher level of players will go down, especially when the size of group increases.

7 Appendix

R Code for Final Project

```
# Installing required packages
install.packages('ggplot2')
install.packages('reshape2')
install.packages('corrplot')
install.packages('gridExtra')
install.packages('NbClust')
install.packages('factoextra')
install.packages('tidyverse')
install.packages('RColorBrewer')
install.packages('scales')
install.packages('rgl')
install.packages('e1071')

# Calling necessary libraries
library(ggplot2)
library(reshape2)
library(corrplot)
library(gridExtra)
library(NbClust)
library(factoextra)
library(tidyverse)
library(RColorBrewer)
library(scales)
library(rgl)
library(e1071)

#-----Import the data------#
chiji.origin<-read.csv("/Users/mingweiwang/Desktop/
PUBG_Player_Statistics.csv",header=TRUE)

#------Clean the data--------#
chiji.clean<-na.omit(chiji.origin)

#--------Data Preparation--------#
# Solo
solo<-chiji.clean[,2:52]
solo<-solo[,-c(4,6,7,8,10,12,18,22,23,25,26,27,30:35,38,39,41:43,47,48,50,51)]
solo$solo_Assists<-solo$solo_Assists/solo$solo_RoundsPlayed
solo$solo_VehicleDestroys<-solo$solo_VehicleDestroys/solo$solo_RoundsPlayed
solo$solo_Boosts<-solo$solo_Boosts/solo$solo_RoundsPlayed

# Duo
duo<-chiji.clean[,c(2,53:102)]
duo<-duo[,-c(4,6,7,8,10,12,21,22,23,26,27,30:35,38,41:43,47,48)]
duo$duo_Assists<-duo$duo_Assists/duo$duo_RoundsPlayed
duo$duo_Suicides<-duo$duo_Suicides/duo$duo_RoundsPlayed
duo$duo_VehicleDestroys<-duo$duo_VehicleDestroys/duo$duo_RoundsPlayed
duo$duo_Boosts<-duo$duo_Boosts/duo$duo_RoundsPlayed
duo$duo_DBNOs<-duo$duo_DBNOs/duo$duo_RoundsPlayed
```

```
# Squad
squad<-chiji.clean[,c(2,103:152)]
squad<-squad[,-c(4,6,7,8,10,12,22,23,26,27,30:35,38,41:43,47,48)]
squad$squad_Assists<-squad$squad_Assists/squad$squad_RoundsPlayed
squad$squad_Suicides<-squad$squad_Suicides/squad$squad_RoundsPlayed
squad$squad_VehicleDestroys<-squad$squad_VehicleDestroys/squad$squad_RoundsPlayed
squad$squad_Boosts<-squad$squad_Boosts/squad$squad_RoundsPlayed
squad$squad_DBNOs<-squad$squad_DBNOs/squad$squad_RoundsPlayed



#------------Part1. Visualization----------------#
#-------Bar Chart-------#
(AveWinRatio.solo<-sum(chiji.clean$solo_Wins)/sum(chiji.clean$solo_RoundsPlayed))
(AveWinRatio.duo<-sum(chiji.clean$duo_Wins)/sum(chiji.clean$squad_RoundsPlayed))
(AveWinRatio.squad<-sum(chiji.clean$squad_Wins)/sum(chiji.clean$squad_RoundsPlayed))
(AveWinRatio.all<-(sum(chiji.clean$squad_Wins)+sum(chiji.clean$squad_Wins)
+sum(chiji.clean$squad_Wins))/(sum(chiji.clean$squad_RoundsPlayed)
+sum(chiji.clean$squad_RoundsPlayed)+sum(chiji.clean$squad_RoundsPlayed)))

PartySize<-c('1','2','4')

AveWinRatio<-c(AveWinRatio.solo,AveWinRatio.duo,AveWinRatio.squad)
mydata <-data.frame(PartySize, AveWinRatio)

p <-ggplot(mydata, aes(PartySize, AveWinRatio))
p +geom_bar(stat = "identity")


#----------Box plot---------#
# Solo
melt_solo<-melt(solo,id.vars = "tracker_id")
ggplot(data=melt_solo) +
  geom_boxplot(aes(x=variable, y=value)) +
  facet_wrap(~variable, scale="free", ncol=5) +
  labs(x="solo", y="Values")

# Duo
melt_duo<-melt(duo,id.vars = "tracker_id")
ggplot(data=melt_duo) +
  geom_boxplot(aes(x=variable, y=value)) +
  facet_wrap(~variable, scale="free", ncol=5) +
  labs(x="duo", y="Values")

# Squad
melt_squad<-melt(squad,id.vars = "tracker_id")
ggplot(data=melt_squad) +
  geom_boxplot(aes(x=variable, y=value)) +
  facet_wrap(~variable, scale="free", ncol=5) +
  labs(x="squad", y="Values")
```

```
#----------Correlegram---------#
# Solo
corr_matrix.solo <- cor(solo[,-1])
corrplot(corr_matrix.solo)
corrplot(corr_matrix.solo,
         method = 'number',
         type = "lower")
solo<-solo[,-c(12,13,17)] # Delete the irrelevant variables

# Duo
corr_matrix.duo <- cor(duo[,-1])
corrplot(corr_matrix.duo)
corrplot(corr_matrix.duo,
         method = 'number',
         type = "lower")
duo<-duo[,-c(13,14,16)]

# Squad
corr_matrix.squad <- cor(squad[,-1])
corrplot(corr_matrix.squad)
corrplot(corr_matrix.squad,
         method = 'number',
         type = "lower")
squad<-squad[,-c(13,14,17)]



#----------Part 2. Dimension Reduction: PCA------#

# Solo
#-------Scaling---------#
solo.scale<-solo
solo.scale[,-1]<-data.frame(scale(solo.scale[,-1]))

# Function we need for standardization
f.data.std <- function(data) {
  data <- as.matrix(data)
  bar <- apply(data, 2, mean)
  s <- apply(data, 2, sd)
  t((t(data) - bar)/s)
}

#-------Standardize---------#
solo.standard<-solo.scale
solo.standard[,-1]<-f.data.std (solo.standard[,-1])
solo.pca <- prcomp(solo.standard[,-1], center = TRUE,scale. = TRUE)
summary(solo.pca)
solo.pca$x <- -solo.pca$x
solo.pca$rotation <- -solo.pca$rotation
solo.pve <- solo.pca$sdev^2 / length(solo.pca$sdev)# Proportion of variance explained
```

```
# Duo
#-------Scaling---------#
duo.scale<-duo
duo.scale[,-1]<-data.frame(scale(duo.scale[,-1]))
#-------Standardize---------#
duo.standard<-duo.scale
duo.standard[,-1]<-f.data.std (duo.standard[,-1])
duo.pca <- prcomp(duo.standard[,-1], center = TRUE,scale. = TRUE)
summary(duo.pca)
duo.pca$x <- -duo.pca$x
duo.pca$rotation <- -duo.pca$rotation
duo.pve <- duo.pca$sdev^2 / length(duo.pca$sdev)# Proportion of variance explained

# Squad
#-------Scaling---------#
squad.scale<-squad
squad.scale[,-1]<-data.frame(scale(squad.scale[,-1]))
#-------Standardize---------#
squad.standard<-squad.scale
squad.standard[,-1]<-f.data.std (squad.standard[,-1])
squad.pca <- prcomp(squad.standard[,-1], center = TRUE,scale. = TRUE)
summary(squad.pca)
squad.pca$x <- -squad.pca$x
squad.pca$rotation <- -squad.pca$rotation
squad.pve <- squad.pca$sdev^2 / length(squad.pca$sdev)# Proportion of variance explained

##----- Scree plot and cumulative proportion of variance explained------##
# Solo
solo.p1 = ggplot() +
  geom_line(aes(x=c(1:length(solo.pca$sdev)), y=solo.pve)) +
  geom_point(aes(x=c(1:length(solo.pca$sdev)), y=solo.pve), size=3) +
  geom_hline(yintercept=solo.pve[5], color='red') +
  labs(x="Principal component", y="Proportion of variance explained")
solo.p2 = ggplot() +
  geom_line(aes(x=c(1:length(solo.pca$sdev)), y=cumsum(solo.pve))) +
  geom_point(aes(x=c(1:length(solo.pca$sdev)), y=cumsum(solo.pve)), size=3) +
  geom_hline(yintercept=cumsum(solo.pve)[5], color='red') +
  annotate("text", x=23, y=0.80, label=paste(round(cumsum(solo.pve)[5],3))) +
  labs(x="Principal component", y="Cumulative proportion of variance explained")
grid.arrange(solo.p1, solo.p2, ncol=2)

# Duo
duo.p1 = ggplot() +
  geom_line(aes(x=c(1:length(duo.pca$sdev)), y=duo.pve)) +
  geom_point(aes(x=c(1:length(duo.pca$sdev)), y=duo.pve), size=3) +
  geom_hline(yintercept=duo.pve[6], color='red') +
  labs(x="Principal component", y="Proportion of variance explained")
duo.p2 = ggplot() +
  geom_line(aes(x=c(1:length(duo.pca$sdev)), y=cumsum(duo.pve))) +
  geom_point(aes(x=c(1:length(duo.pca$sdev)), y=cumsum(duo.pve)), size=3) +
  geom_hline(yintercept=cumsum(duo.pve)[6], color='red') +
  annotate("text", x=23, y=0.80, label=paste(round(cumsum(duo.pve)[6],3))) +
  labs(x="Principal component", y="Cumulative proportion of variance explained")
grid.arrange(duo.p1, duo.p2, ncol=2)
```

```
# Squad
squad.p1 = ggplot() +
  geom_line(aes(x=c(1:length(squad.pca$sdev)), y=squad.pve)) +
  geom_point(aes(x=c(1:length(squad.pca$sdev)), y=squad.pve), size=3) +
  geom_hline(yintercept=squad.pve[6],color='red') +
  labs(x="Principal component", y="Proportion of variance explained")
squad.p2 = ggplot() +
  geom_line(aes(x=c(1:length(squad.pca$sdev)), y=cumsum(squad.pve))) +
  geom_point(aes(x=c(1:length(squad.pca$sdev)), y=cumsum(squad.pve)), size=3) +
  geom_hline(yintercept=cumsum(squad.pve)[6], color='red') +
  annotate("text", x=23, y=0.80, label=paste(round(cumsum(squad.pve)[6],3))) +
  labs(x="Principal component", y="Cumulative proportion of variance explained")
grid.arrange(squad.p1, squad.p2, ncol=2)


##-----------Heatmap of variables and main principal components------------##
# Solo
melt_solo.pca <- melt(solo.pca$rotation[,c(1:11)])
colnames(melt_solo.pca) <- c("Solo", "PC", "Value")
ggplot(data=melt_solo.pca) +
  geom_tile(aes(x=PC, y=Solo, fill=Value)) +
  scale_fill_gradient2(low='blue', mid='white', high='red', midpoint=0) +
  labs(x="Principal component")
# Duo
melt_duo.pca <- melt(duo.pca$rotation[,c(1:12)])
colnames(melt_duo.pca) <- c("Duo", "PC", "Value")
ggplot(data=melt_duo.pca) +
  geom_tile(aes(x=PC, y=Duo, fill=Value)) +
  scale_fill_gradient2(low='blue', mid='white', high='red', midpoint=0) +
  labs(x="Principal component")
# Squad
melt_squad.pca <- melt(squad.pca$rotation[,c(1:12)])
colnames(melt_squad.pca) <- c("Squad", "PC", "Value")
ggplot(data=melt_squad.pca) +
  geom_tile(aes(x=PC, y=Squad, fill=Value)) +
  scale_fill_gradient2(low='blue', mid='white', high='red', midpoint=0) +
  labs(x="Principal component")

# ------- Full spectrum———
# Solo
melt_solo.pca1<- melt(solo.pca$rotation[,c(1:5)])
colnames(melt_solo.pca1) = c("Solo", "PC", "Value")
ggplot(data=melt_solo.pca1) +
  geom_col(aes(x=Solo, y=Value)) +
  facet_wrap(~PC, ncol=1) +
  theme(axis.text.x = element_text(angle = 90)) +labs(x="Solo", y="Loading")

# Duo
melt_duo.pca1<- melt(duo.pca$rotation[,c(1:6)])
colnames(melt_duo.pca1) = c("Duo", "PC", "Value")
ggplot(data=melt_duo.pca1) +
  geom_col(aes(x=Duo, y=Value)) +
  facet_wrap(~PC, ncol=1) +
  theme(axis.text.x = element_text(angle = 90)) +labs(x="duo", y="Loading")
```

```
# Squad
melt_squad.pca1<- melt(squad.pca$rotation[,c(1:6)])
colnames(melt_squad.pca1) = c("Squad", "PC", "Value")
ggplot(data=melt_squad.pca1) +
  geom_col(aes(x=Squad, y=Value)) +
  facet_wrap(~PC, ncol=1) +
  theme(axis.text.x = element_text(angle = 90)) +labs(x="squad", y="Loading")

##----------------Selected Variables----------##
### Selection of variables

# Solo
pve.mat <- matrix(rep(solo.pve, each = 5), nrow=length(solo.pve))
solo.impact <- apply(solo.pca$rotation[,c(1:5)]^2 * pve.mat, 1, sum)
melt_NI <- melt(solo.impact)
ggplot(data=melt_NI) +
  geom_col(aes(x=reorder(rownames(melt_NI), -value), y=value)) +
  theme(axis.text.x = element_text(angle = 90)) +
  labs(x="Solo", y="Variable importance")
(top10<-rownames(melt_NI)[order(-melt_NI$value)[1:10]])

# Duo
pve.mat <- matrix(rep(duo.pve, each = 6), nrow=length(duo.pve))
duo.impact <- apply(duo.pca$rotation[,c(1:6)]^2 * pve.mat, 1, sum)
melt_NI <- melt(duo.impact)
ggplot(data=melt_NI) +
  geom_col(aes(x=reorder(rownames(melt_NI), -value), y=value)) +
  theme(axis.text.x = element_text(angle = 90)) +
  labs(x="duo", y="Variable importance")
(top10<-rownames(melt_NI)[order(-melt_NI$value)[1:10]])

# Squad
pve.mat <- matrix(rep(squad.pve, each = 6), nrow=length(squad.pve))
squad.impact <- apply(squad.pca$rotation[,c(1:6)]^2 * pve.mat, 1, sum)
melt_NI <- melt(squad.impact)
ggplot(data=melt_NI) +
  geom_col(aes(x=reorder(rownames(melt_NI), -value), y=value)) +
  theme(axis.text.x = element_text(angle = 90)) +
  labs(x="squad", y="Variable importance")
(top10<-rownames(melt_NI)[order(-melt_NI$value)[1:10]])



#-------------Part3. Unsupervised learning (Clustering)-----------------#
#------------KMeans Cluster:solo----------------#
solo.comp<-data.frame(solo.pca$x[,1:5])
k.solo<- kmeans(solo.comp, 5)
palette(alpha(brewer.pal(9,'Set1'), 0.5))
plot(solo.comp, col=k.solo$clust, pch=16)
k.solo$cluster
k.solo$size
# 3D plot
plot3d(solo.comp$PC1,solo.comp$PC2,solo.comp$PC3,col=k.solo$clust)
plot3d(solo.comp$PC3,solo.comp$PC4,solo.comp$PC5,col=k.solo$clust)
```

```
#--------------Disribution of Variable:solo--------------#
k.solo$cluster <- as.factor(k.solo$cluster)
ggplot(solo, aes(solo_Top10Ratio,solo_KillDeathRatio, color = k.solo$cluster)) + geom_point()
ggplot(solo, aes(solo_DamagePg,solo_KillDeathRatio, color = k.solo$cluster)) + geom_point()

#------------KMeans Cluster: Duo-----------------#
duo.comp<-data.frame(duo.pca$x[,1:6])
k.duo<- kmeans(duo.comp, 6)
palette(alpha(brewer.pal(9,'Set1'), 0.5))
plot(duo.comp, col=k.duo$clust, pch=16)
k.duo$cluster
k.duo$size
# 3D plot
plot3d(duo.comp$PC1,duo.comp$PC2,duo.comp$PC3,col=k.duo$clust)
plot3d(duo.comp$PC4,duo.comp$PC5,duo.comp$PC6,col=k.duo$clust)
#--------------Disribution of Variable:Duo--------------#
k.duo$cluster <- as.factor(k.duo$cluster)
ggplot(duo, aes(duo_Top10Ratio,duo_DamagePg, color = k.duo$cluster)) + geom_point()
ggplot(duo, aes(duo_KillDeathRatio,duo_DamagePg, color = k.duo$cluster)) + geom_point()

#------------KMeans Cluster: Squad-----------------#
squad.comp<-data.frame(squad.pca$x[,1:6])
k.squad<- kmeans(squad.comp, 6)
palette(alpha(brewer.pal(9,'Set1'), 0.5))
plot(squad.comp, col=k.squad$clust, pch=16)
k.squad$cluster
k.squad$size
# 3D plot
plot3d(squad.comp$PC1,squad.comp$PC2,squad.comp$PC3,col=k.squad$clust)
plot3d(squad.comp$PC4,squad.comp$PC5,squad.comp$PC6,col=k.squad$clust)
#--------------Disribution of Variable:Squad--------------#
k.squad$cluster <- as.factor(k.squad$cluster)
ggplot(squad, aes(squad_Top10Ratio,squad_DamagePg, color = k.squad$cluster)) +
geom_point()
ggplot(squad, aes(squad_KillDeathRatio,squad_DamagePg, color = k.squad$cluster)) +
geom_point()



#-------------Part4. Supervised learning (Classification)----------------#
#-----------Subset data according to KD-----------#
# Solo
KDClass.solo<-solo$solo_KillDeathRatio
solo<-cbind(solo,KDClass.solo)
solo[[ "KDClass.solo"]]<-ordered(cut(as.numeric(solo[[ "KDClass.solo"]]), c(-1,1,1.5,2,3,101)),
                                 labels = c(as.numeric(1),as.numeric(2), as.numeric(3),
as.numeric(4),as.numeric(5)))

# Duo
KDClass.duo<-duo$duo_KillDeathRatio
duo<-cbind(duo,KDClass.duo)
duo[[ "KDClass.duo"]]<-ordered(cut(as.numeric(duo[[ "KDClass.duo"]]), c(-1,1,1.5,2,3,101)),
                               labels = c(as.numeric(1),as.numeric(2), as.numeric(3),
as.numeric(4),as.numeric(5)))
```

```
# Squad
KDClass.squad<-squad$squad_KillDeathRatio
squad<-cbind(squad,KDClass.squad)
squad[[ "KDClass.squad"]]<-ordered(cut(as.numeric(squad[[ "KDClass.squad"]]),
c(-1,1,1.5,2,3,101)),
                                   labels = c(as.numeric(1),as.numeric(2), as.numeric(3),
as.numeric(4),as.numeric(5)))

#------Delete the variabeles that have no correlation ship with KD-------#
# Solo
solo.svm<-solo[,-c(2,6,16,17)]
# Duo
duo.svm<-duo[,-c(2,6,17,19)]
# Squad
squad.svm<-squad[,-c(2,6,18,20)]


#------Data Slicing-solo------#
(n<-nrow(chiji.clean)*0.8)
set.seed(1)
train.solo.ind<-sample(nrow(solo),n, replace = FALSE, prob = NULL)
training.solo<-solo.svm[train.solo.ind,]
test.solo<-solo.svm[-train.solo.ind,]
dim(training.solo)
summary(training.solo$KDClass.solo)
dim(test.solo)
summary(test.solo$KDClass.solo)

#------Data Slicing-duo------#
train.duo.ind<-sample(nrow(duo),n, replace = FALSE, prob = NULL)
training.duo<-duo.svm[train.duo.ind,]
test.duo<-duo.svm[-train.duo.ind,]
dim(training.duo)
summary(training.duo$KDClass.duo)
dim(test.duo)
summary(test.duo$KDClass.duo)

#------Data Slicing-squad------#
train.squad.ind<-sample(nrow(squad),n, replace = FALSE, prob = NULL)
training.squad<-squad.svm[train.squad.ind,]
test.squad<-squad.svm[-train.squad.ind,]
dim(training.squad)
summary(training.squad$KDClass.squad)
dim(test.squad)
summary(test.squad$KDClass.squad)

#-------------SVM Model using Nonlinear Kernel-------------#
## Solo ##
#------Find best model for training set------#
set.seed(1)
train0.ind.solo<-sample(nrow(training.solo),2000, replace = FALSE, prob = NULL)
training0.solo<-training.solo[train0.ind.solo,]
training0.solo<-training0.solo[,-1]
```

```
x0.solo <- subset(training0.solo, select=-KDClass.solo)
y0.solo <- training0.solo$KDClass.solo
dat0.solo <- data.frame(x=x0.solo,y=as.factor(y0.solo))
svmfit0.solo <- svm(y~., data = dat0.solo)
tune.out0.solo <- tune(svm, y~., data = dat0.solo, kernel = "radial",
                       ranges = list(cost = c(0.1,0.5,1,5,10),
                                     gamma = c(0.5,1,1.5,2)))
tune.out0.solo$best.model

training.solo<-training.solo[,-1]
x.solo <- subset(training.solo, select=-KDClass.solo)
y.solo <- training.solo$KDClass.solo
dat.solo <- data.frame(x=x.solo,y=as.factor(y.solo))
svmfit.solo <- svm(y~., data = dat.solo) # By Defult parameters
svmfit.solo <- svm(y~., data = dat.solo, kernel = "radial", cost=5,gamma=0.5) # After tuning
system.time(ypred.solo<-predict(svmfit.solo,dat.solo))
#------Confused Matrix------#
(misclass.solo <- table(predict.solo = ypred.solo, truth = dat.solo$y))
(acc.solo<-
(misclass.solo[1,1]+misclass.solo[2,2]+misclass.solo[3,3]+misclass.solo[4,4]+misclass.solo[5,5]
)/70318)


## Duo ##
#------Find best model for training set------#
set.seed(1)
train0.ind.duo<-sample(nrow(training.duo),2000, replace = FALSE, prob = NULL)
training0.duo<-training.duo[train0.ind.duo,]
training0.duo<-training0.duo[,-1]

x0.duo <- subset(training0.duo, select=-KDClass.duo)
y0.duo <- training0.duo$KDClass.duo
dat0.duo <- data.frame(x=x0.duo,y=as.factor(y0.duo))
svmfit0.duo <- svm(y~., data = dat0.duo)
tune.out0.duo <- tune(svm, y~., data = dat0.duo, kernel = "radial",
                      ranges = list(cost = c(0.1,0.5,1,5,10),
                                    gamma = c(0.5,1,1.5,2)))
tune.out0.duo$best.model

training.duo<-training.duo[,-1]
x.duo <- subset(training.duo, select=-KDClass.duo)
y.duo <- training.duo$KDClass.duo
dat.duo <- data.frame(x=x.duo,y=as.factor(y.duo))
svmfit.duo <- svm(y~., data = dat.duo) # By Defult parameters
svmfit.duo <- svm(y~., data = dat.duo, kernel = "radial", cost=5,gamma=0.5) # After tuning
system.time(ypred.duo<-predict(svmfit.duo,dat.duo))
#------Confused Matrix------#
(misclass.duo <- table(predict.duo = ypred.duo, truth = dat.duo$y))
(acc.duo<-
(misclass.duo[1,1]+misclass.duo[2,2]+misclass.duo[3,3]+misclass.duo[4,4]+misclass.duo[5,5])/
70318)
```

```
## Squad ##
#------Find best model for training set------#
set.seed(1)
train0.ind.squad<-sample(nrow(training.squad),2000, replace = FALSE, prob = NULL)
training0.squad<-training.squad[train0.ind.squad,]
training0.squad<-training0.squad[,-1]

x0.squad <- subset(training0.squad, select=-KDClass.squad)
y0.squad <- training0.squad$KDClass.squad
dat0.squad <- data.frame(x=x0.squad,y=as.factor(y0.squad))
svmfit0.squad <- svm(y~., data = dat0.squad)
tune.out0.squad <- tune(svm, y~., data = dat0.squad, kernel = "radial",
                        ranges = list(cost = c(0.1,0.5,1,5,10),
                                      gamma = c(0.5,1,1.5,2)))
tune.out0.squad$best.model

training.squad<-training.squad[,-1]
x.squad <- subset(training.squad, select=-KDClass.squad)
y.squad <- training.squad$KDClass.squad
dat.squad <- data.frame(x=x.squad,y=as.factor(y.squad))
svmfit.squad <- svm(y~., data = dat.squad) # By Defult parameters
svmfit.squad <- svm(y~., data = dat.squad, kernel = "radial", cost=10,gamma=0.5) # After
tuning
system.time(ypred.squad<-predict(svmfit.squad,dat.squad))
#------Confused Matrix------#
(misclass.squad <- table(predict.squad = ypred.squad, truth = dat.squad$y))
(acc.squad<-
(misclass.squad[1,1]+misclass.squad[2,2]+misclass.squad[3,3]+misclass.squad[4,4]+misclass.
squad[5,5])/70318)



#----------------Test Set Prediction-------------------#

## Solo ##
x.test.solo <- subset(test.solo, select=-KDClass.solo)
y.test.solo <- test.solo$KDClass.solo
dat.test.solo <- data.frame(x=x.test.solo,y=as.factor(y.test.solo))
ypred.test.solo <- predict(svmfit.solo, dat.test.solo)
(misclass.test.solo <- table(predict = ypred.test.solo, truth = dat.test.solo$y))
(t.acc.solo<-
(misclass.test.solo[1,1]+misclass.test.solo[2,2]+misclass.test.solo[3,3]+misclass.test.solo[4,4]+
misclass.test.solo[5,5])/17580)


## Duo ##
x.test.duo <- subset(test.duo, select=-KDClass.duo)
y.test.duo <- test.duo$KDClass.duo
dat.test.duo <- data.frame(x=x.test.duo,y=as.factor(y.test.duo))
ypred.test.duo <- predict(svmfit.duo, dat.test.duo)
(misclass.test.duo <- table(predict = ypred.test.duo, truth = dat.test.duo$y))
(t.acc.duo<-
(misclass.test.duo[1,1]+misclass.test.duo[2,2]+misclass.test.duo[3,3]+misclass.test.duo[4,4]+m
isclass.test.duo[5,5])/17580)
```

```
## Squad ##
x.test.squad <- subset(test.squad, select=-KDClass.squad)
y.test.squad <- test.squad$KDClass.squad
dat.test.squad <- data.frame(x=x.test.squad,y=as.factor(y.test.squad))
ypred.test.squad <- predict(svmfit.squad, dat.test.squad)
(misclass.test.squad <- table(predict = ypred.test.squad, truth = dat.test.squad$y))
(t.acc.squad<-
(misclass.test.squad[1,1]+misclass.test.squad[2,2]+misclass.test.squad[3,3]+misclass.test.squ
ad[4,4]+misclass.test.squad[5,5])/17580)
```

Note: All the content and codes of each sections have been fully discussed and repeatedly verified by all three members of our team during the whole research procedure, therefore the statement outlining responsibility for each sections is not necessary and fair to present in this case. Thanks for your comprehension.