I Tianpei Liao student ID # 215632375 acknowledge that I have contributed at least 30% of time and effort to the preparation of this report and work discussed herein.

Tianpei Liao

I Jiahao Li student ID # 216263949 acknowledge that I have contributed at least 30% of time and effort to the preparation of this report and work discussed herein.

Jiahao Li

I Wenxuan Li student ID # 216374324 acknowledge that I have contributed at least 30% of time and effort to the preparation of this report and work discussed herein.

Wenxuan Li

EECS 4412

Project Part 01

Instructor: Habib-ur Rehman

Datasets Background

In our dataset, it has all the data about the 2019 League of Legends World Championship. There are total 3 tables:

1. **wc_champions.csv:** This entity describes the information of champions that the players used.
2. **wc_matches.csv:** This entity describes the information of each match from different teams in the World Championship.
3. **wc_players.csv:** This entity shows each player's statistics.

1. Task 1

1.1 wc_champions.csv:

In the 'wc_champions.csv', we choose 5 dimensions:

"champion": The name of champions.

"sum_total": The total games that this champion has in.

"win_total": Total wins on both sides.

"matches_less_25_min": Total matches with less than 25 min length.

"winrate_less_25_min": Win rate in matches with less than 25 min length.

The dimension "champion" is a nominal type of data and it represents the dataset's recorded meaning. So we use it as the x-axis in our graph.

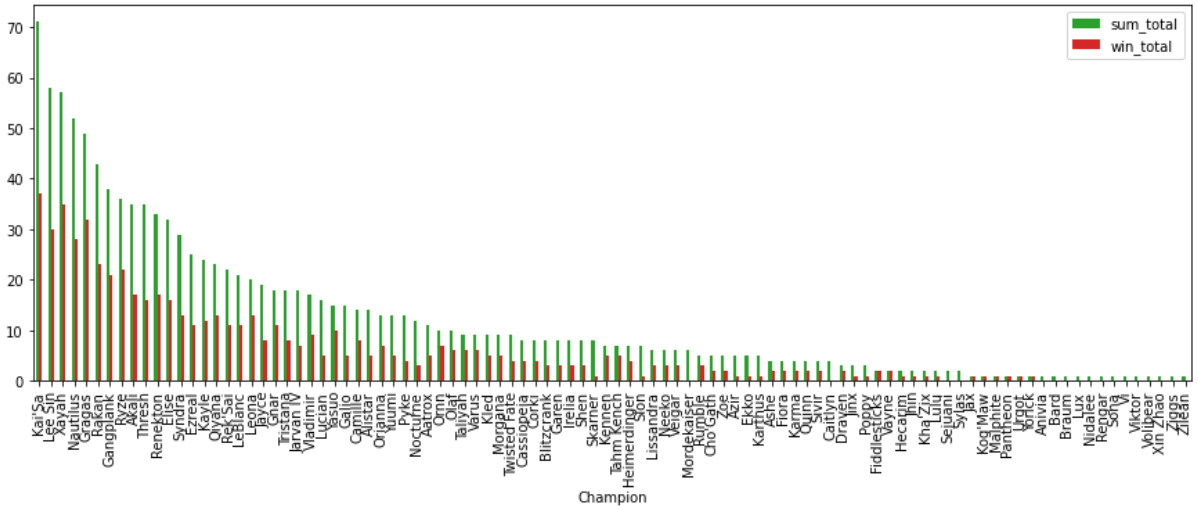

figure 1.1

The total number of champions and the total win number are put together. Because we want to see how popular the champion is and how many win rounds in total. The bar chart is most suitable for this task.

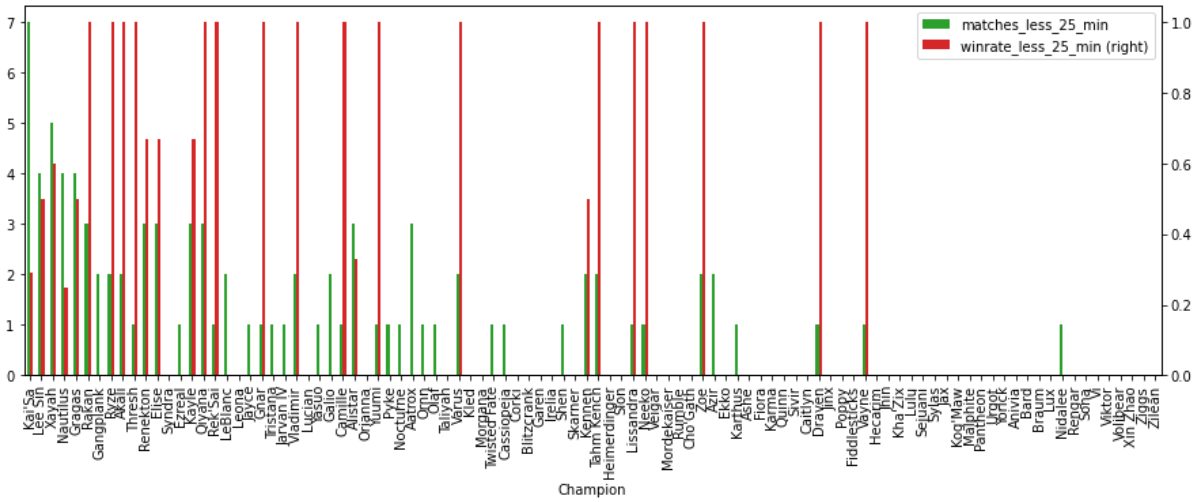

figure 1.2

Here we want to show the matches end early and see how the champion is involved in the early stage. So we focus on the game in less than 25 mins and their corresponding win rate.

1.2 wc_matches.csv

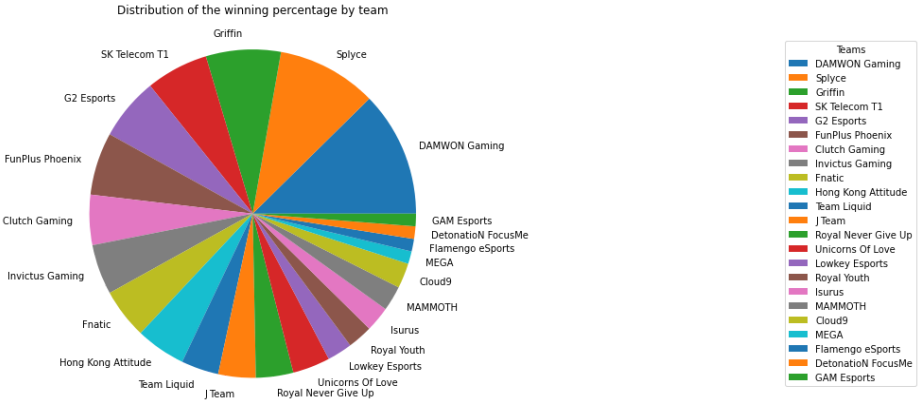

Figure 1.3

Figure 1.3 shows the pie chart of the winning percentage by different teams. It clearly shows that teams like Splyce and DAMWON Gaming have the nearly highest winning percentage.

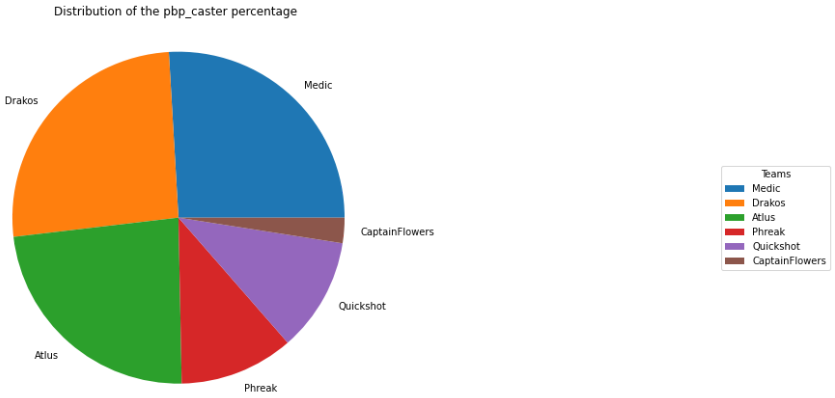

Figure 1.4

Figure 1.4 shows the pie chart of the pbp_caster percentage by different persons.

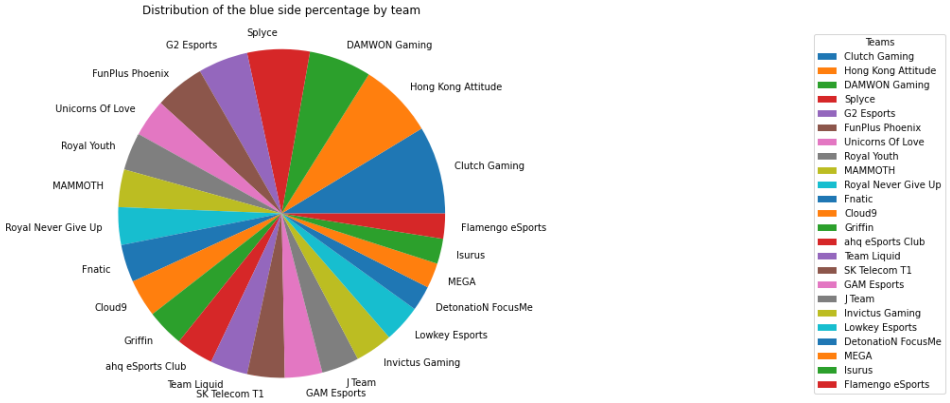

Figure 1.5

Figure 1.5 shows the pie chart of the blue side percentage by different teams.

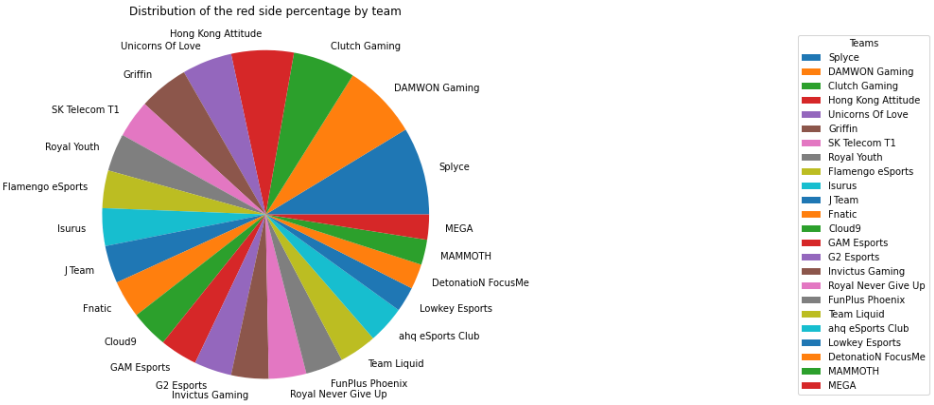

Figure 1.6

Figure 1.6 shows the pie chart of the red side percentage by different teams.

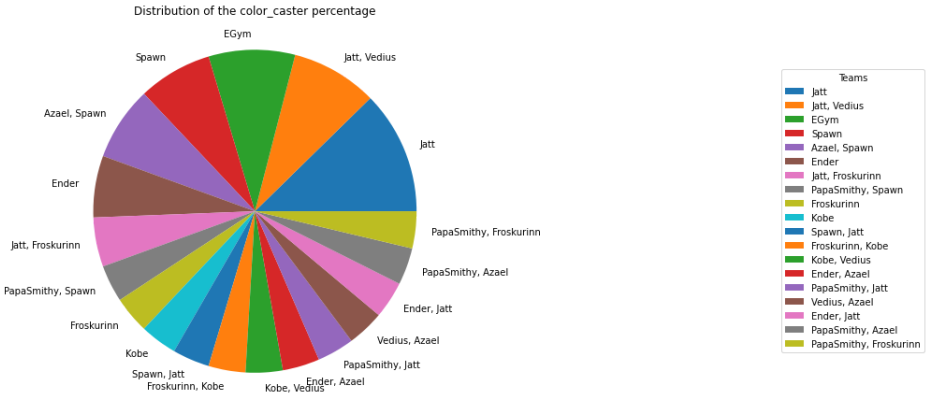

Figure 1.7

Figure 1.7 shows the pie chart of the color_caster percentage by different people.

1.3 wc_players.csv

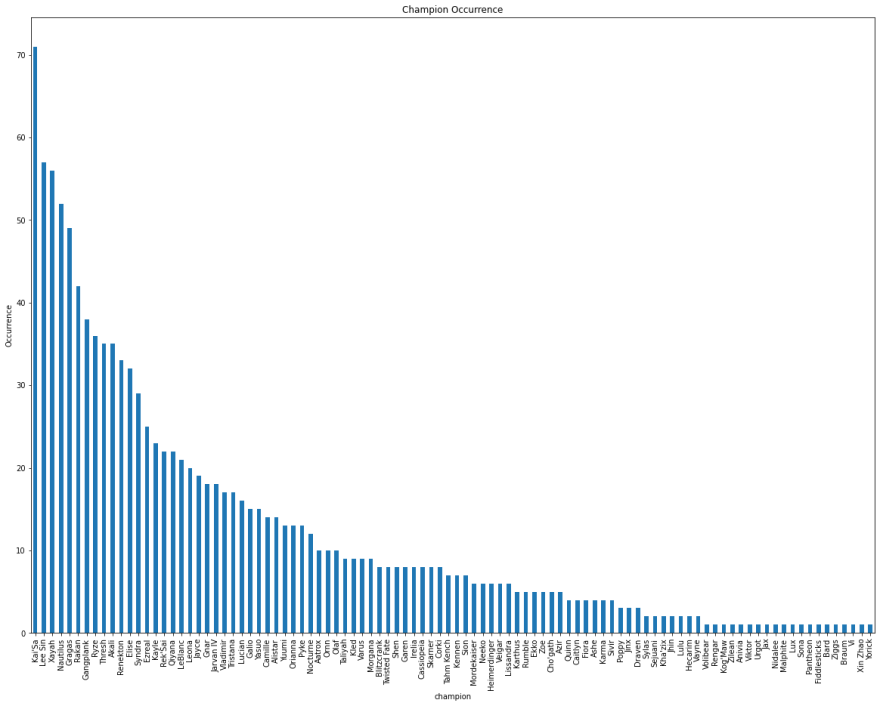

Figure 1.8

Figure 1.8 shows the bar chart of the champion occurrence.

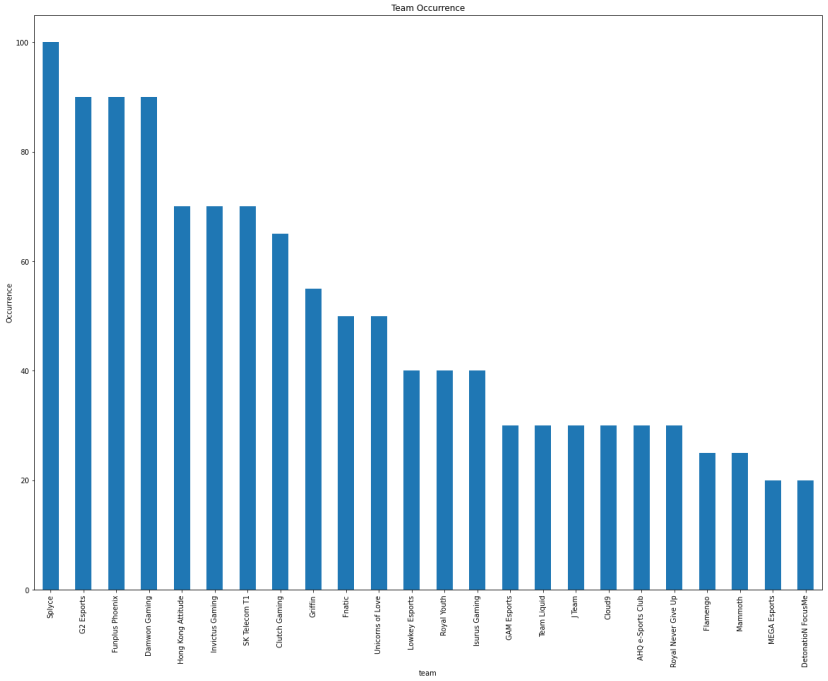

Figure 1.9

Figure 1.9 shows the bar chart of the team occurrence.

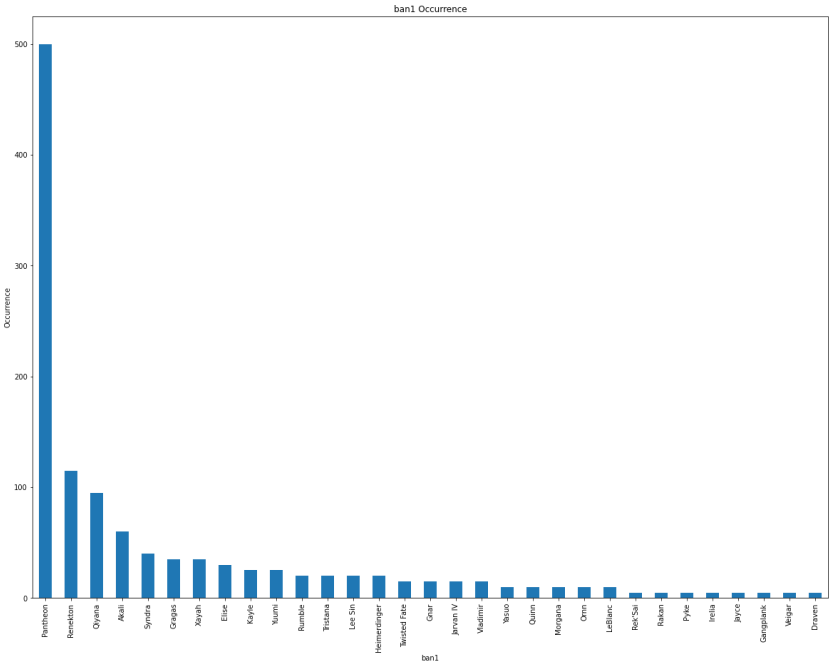

Figure 1.10

Figure 1.10 shows the bar chart of the ban1 occurrence.

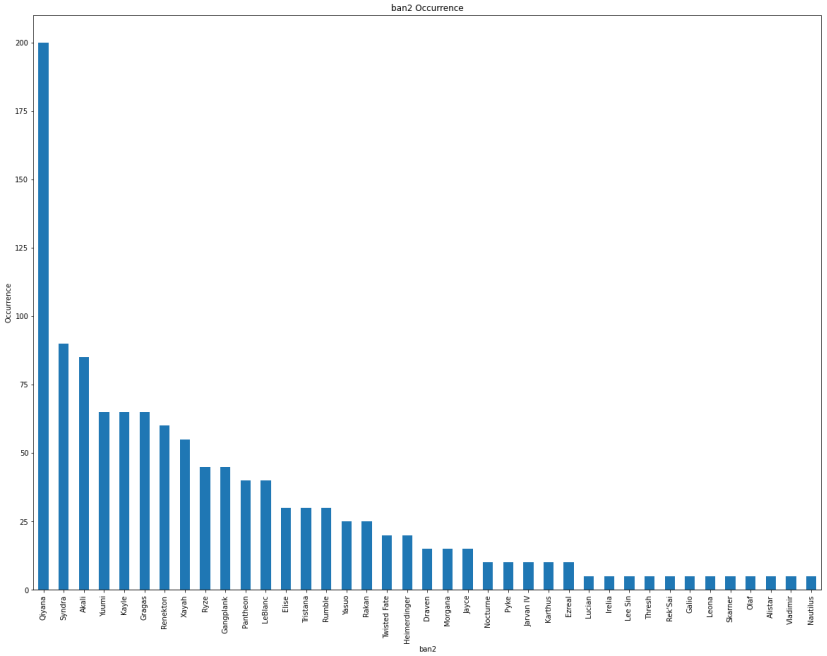

Figure 1.11

Figure 1.11 shows the bar chart of the ban2 occurrence.

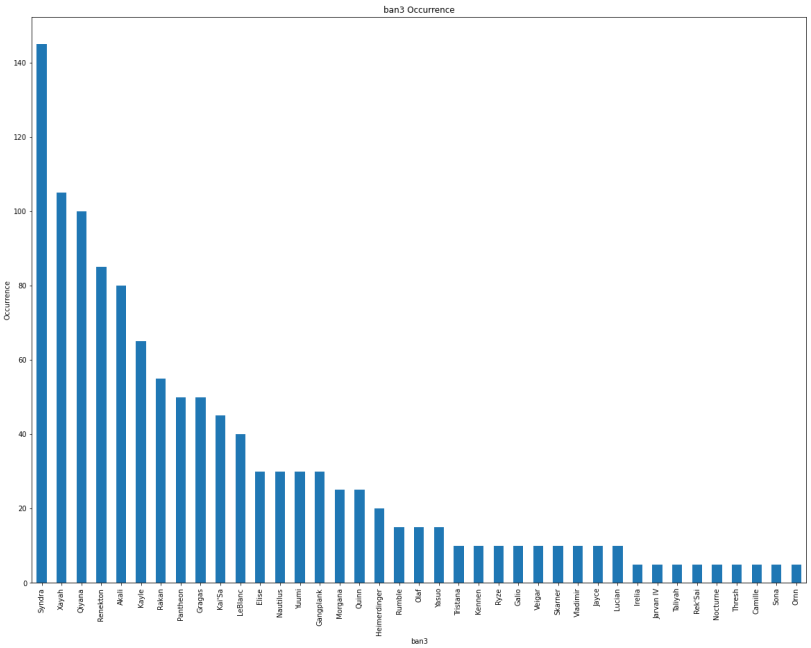

Figure 1.12

Figure 1.12 shows the bar chart of the ban3 occurrence.

2. Task 2

2.1 Nominal:

We use the ‘winner’ dimension in the wc_matches.csv. It is a Nominal data type so we want to know which team wins the most games in the matches. The **value_counts()** attribute of pandas can count these instances of the winning team. The team that wins the most games is the **max value** in the value_counts.
So “DAMN Gaming” has the most winning number. The result of the statistic is the figure 2.1.

```
DAMWON Gaming          10
Splyce                  8
Griffin                 6
SK Telecom T1           5
G2 Esports              5
FunPlus Phoenix         5
Clutch Gaming           4
Invictus Gaming         4
Fnatic                  4
Hong Kong Attitude      4
Team Liquid             3
J Team                  3
Royal Never Give Up     3
Unicorns Of Love        3
Lowkey Esports          2
Royal Youth             2
Isurus                  2
MAMMOTH                 2
Cloud9                  2
MEGA                    1
Flamengo eSports        1
DetonatioN FocusMe      1
GAM Esports             1
Name: winner, dtype: int64
```

figure 2.1

2.2 Ordinal:

Since we do not have any ordinal data type in our dataset. We create a data dimension in the dataset called “strength” by applying the defined function. This function outputs the “op”(over power) of a given champion. The type of “op” has 5 levels: “T0”, “T1”, “T2”, “T3”, “normal”. The generation of this dimension is based on the total number of this champion and its winrate. (figure 2.2)

```
# first create a new ordinal column called: "strength"
# based on the op (over power)
def strength (row):
    if row['sum_total'] <= 20 :
        return 'normal'
    if row['winrate_total'] >= 0.6:
        return 'T0'
    if row['winrate_total'] >= 0.55:
        return 'T1'
    if row['winrate_total'] >= 0.52:
        return 'T2'
    if row['winrate_total'] >= 0.50:
        return 'T3'
    return 'normal'

data_task2_1['strength'] = data_task2_1.apply (lambda row: strength(row), axis=1)
```

figure 2.2

Then perform **value_counts()** on it (figure 2.3). We want to see how many champions can be rated as “T0” in this criteria. This is helpful in analyzing which champion is most popular and valuable in the game. And the type with the **maximum** number is “normal”. This shows that most champions are not qualified to be called “op”.

```
normal    84
T2         6
T0         3
T3         3
T1         2
Name: strength, dtype: int64
```

figure 2.3

2.3 Interval:

We use the “sum_total” dimension in the wc_champions. We care about **the mean** and **the standard deviation.** The method describe() is powerful which can provide those statistical properties. The maximum value tells the average time of every champion in the game. The standard deviation value is large so that it means the popularity of the champion is not on the same level. Some champions are preferred to be used during the game.

```
csv_task2['sum_total'].describe()

count    98.000000
mean     12.244898
std      14.474336
min       1.000000
25%       2.000000
50%       7.000000
75%      15.750000
max      71.000000
Name: sum_total, dtype: float64
```

figure 2.4

2.4 Ratio:

We use the “winrate_total” dimension in the wc_champions. We wonder how **many winrate of these champions are more than 50%** and **the mean**. The mean value can be accessed by method describe(). Then we define a method to return 1 if the rate is greater than 50%, 0 otherwise. Then sum it will give the number of the champions that have more than 50% win rate.

```
count    98.000000
mean      0.438776
std       0.271769
min       0.000000
25%       0.315000
50%       0.500000
75%       0.567500
max       1.000000
Name: winrate_total, dtype: float64
total number of champion that has more than 50% winrate:
36
```

figure 2.5

3. Task 3

We select the “**winrate_total**” and “**sum_total**” in **wc_champions** dataset.

3.1 “**winrate_total**” in wc_champions.

3.1.1 Z-score standardization of “winrate_total”

First, we print original data using some same printing method (figure 3.1)

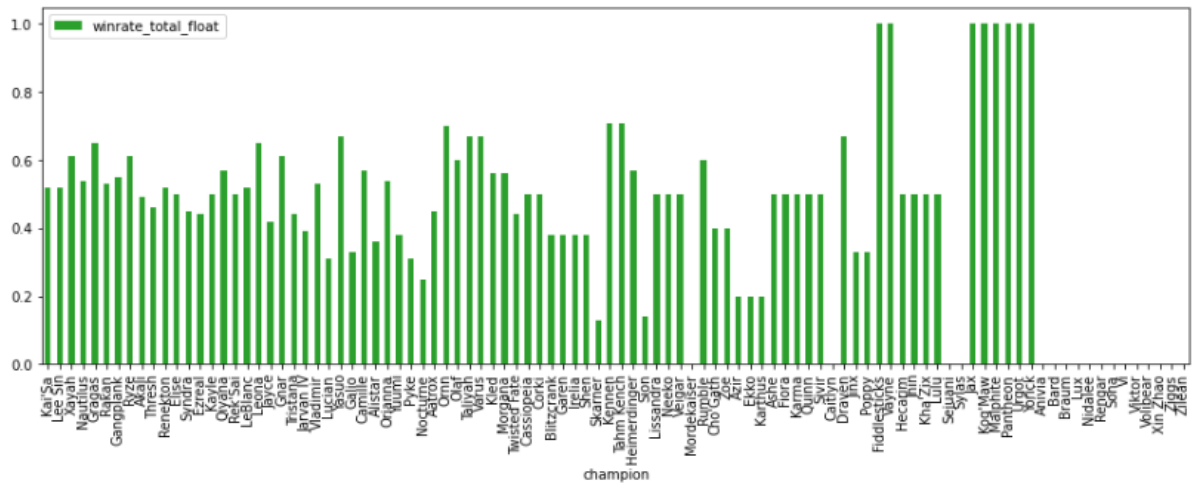

figure 3.1

and then is the data after Z-score standardization (figure 3.2)

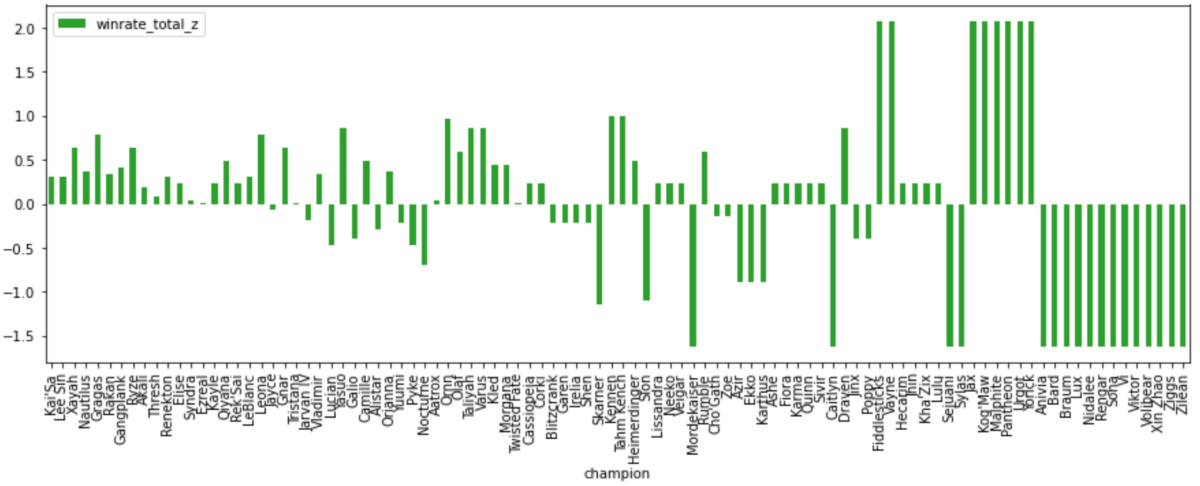

figure 3.2

We can first notice that the y domain is changed, because after the Z-Score processing the data, the result may not be exactly the same scale as the original data scale. After Z-score standardization, we can clearly see the data point distribution and easier to find out the outliers. For example, some of the champions' win rate is very high,far exceeding other normal values.

3.1.2 Min-Max normalization of “winrate_total”

We compare the original data (figure 3.3)

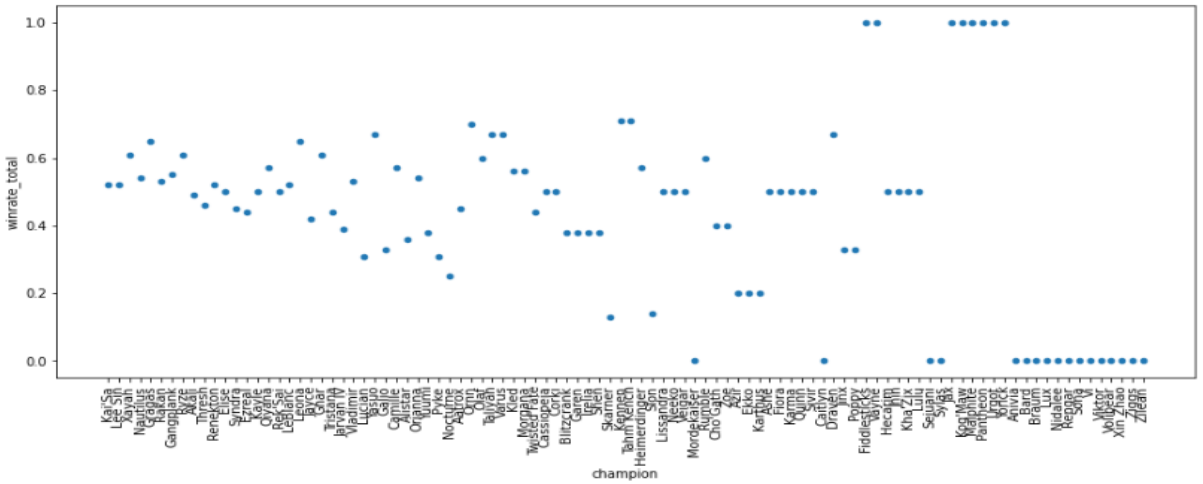

figure 3.3

and after Min-Max normalization processing data (figure 3.4)

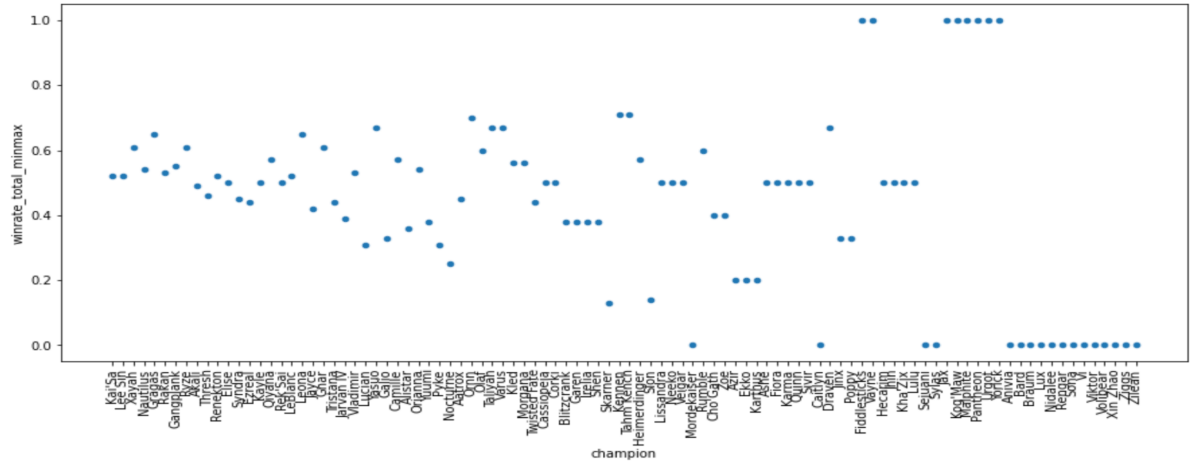

figure 3.4

This time the two plots are the same, because the winrate_total data is ratio data, so the data will be between 0 to 1, and min-max normalization is to process the data so that the value after processing is between 0 and 1. That is why two plot scales are the same, but the min-max normalization will maintain the scale of the data, that is, the processed data will be between 0 and 1, but the shape and characteristics of the data will be consistent with the original data.

3.2 “**sum_total**” in **wc_champions**

3.2.1 Z-score standardization of “sum_total”

We compare **sum_total** data (figure 3.5)

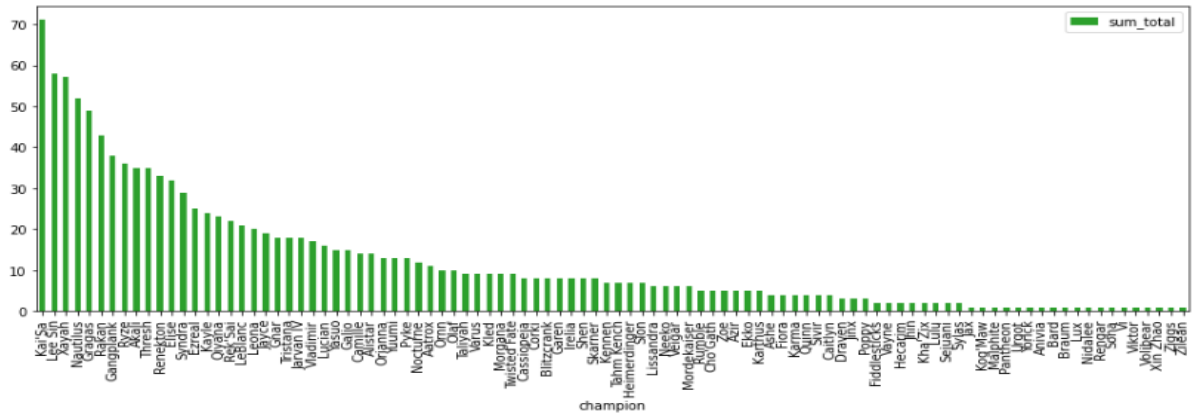

figure 3.5

and it Z-score standardization data (figure 3.6)

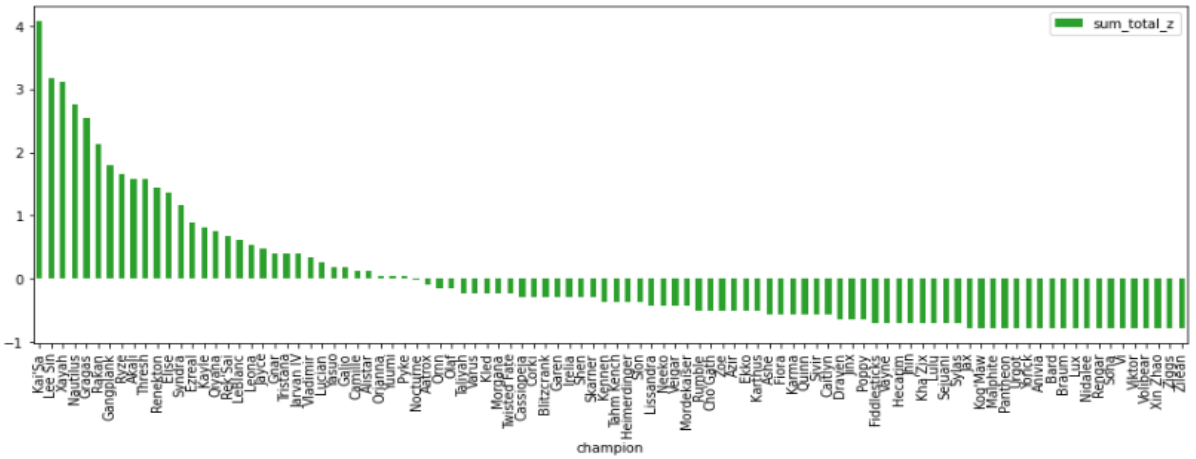

figure 3.6

We can see the same characteristic in winrate_total dimension

3.2.2 Min-Max normalization of “sum_total”

We compare the original **sum_total** (figure 3.7)

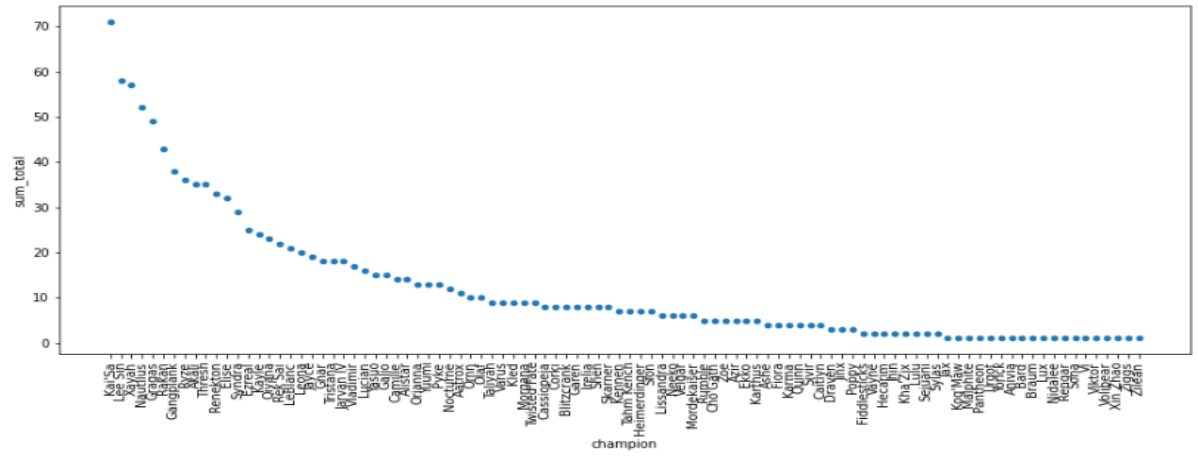

figure 3.7

and the data after Min-Max normalization (figure 3.8)

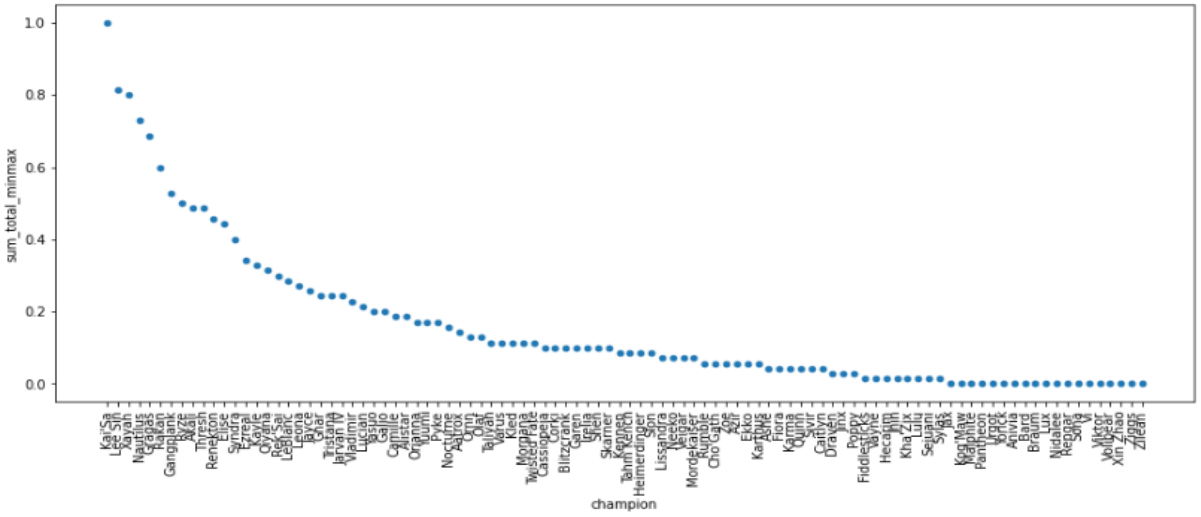

figure 3.8

Because the sum_total data is not ratio data but interval data, the characteristics of Min-Max normalization this time are revealed. We observe that the shape of the data does not change a bit, but the domain changes from 0 to 70 to 0 to 1.

3.4 Comment

Both technologies have their own advantages, so neither one is better than the other. Z-score standardization can better deal with outliers, but it will change the scale of the data. Min-Max normalization will relatively keep all features as the original scale, and the data characteristics will also be maintained, but the disadvantage of this is that it is not easy to find and process outliers.

4. Task 4

In our dataset, the most dimensions is the player object, it totally has 91 features. In this principal component analysis, we choose 10 dimensions respectively kill, death, asist, team kills, team deaths, first blood time, kill per mins, game length, first dead time, team drag kills.

	k	d	a	teamkills	teamdeaths	fbtime	fdtime	kpm	gamelength	teamdragkills
0	0	4	4	7	22	3.879983	9.717567	0.000000	26.533333	1
1	1	4	5	7	22	3.879983	9.717567	0.037688	26.533333	1
2	1	5	6	7	22	3.879983	9.717567	0.037688	26.533333	1
3	3	4	1	7	22	3.879983	9.717567	0.113065	26.533333	1
4	2	5	2	7	22	3.879983	9.717567	0.075377	26.533333	1

After selecting 10 dimensions, we start to PCA processing the data.

	0	1	2	...	1187	1188	1189
PCs_1	-2.347315	-1.872859	-1.875595	...	-1.572361	-0.483346	-1.572361
PCs_2	0.890257	0.873345	1.231215	...	-1.816698	-2.358062	-1.816698
PCs_3	0.242599	0.584920	0.515075	...	-0.859954	0.479129	-0.859954
PCs_4	0.644679	0.537906	0.475400	...	-2.109134	-2.474312	-2.109134
PCs_5	-0.433026	-0.448100	-0.518818	...	1.290621	1.382547	1.290621
PCs_6	-0.431132	-0.590534	-0.843725	...	-0.595225	-0.435555	-0.595225
PCs_7	-0.275590	-0.189095	-0.446916	...	-0.175816	0.125140	-0.175816
PCs_8	-0.779711	-0.714473	-0.461456	...	-0.125817	-0.229011	-0.125817
PCs_9	-0.086818	-0.390901	-0.569526	...	0.327420	0.047769	0.327420
PCs_10	-0.091469	-0.040116	-0.046648	...	-0.017842	0.029857	-0.017842

Then we use base on ten principal components to calculate their variance

[10 rows x 1190 columns]

PCs_1	3.060561
PCs_2	2.105294
PCs_3	1.521345
PCs_4	1.092919
PCs_5	0.826730
PCs_6	0.674309
PCs_7	0.379409
PCs_8	0.202399
PCs_9	0.128578
PCs_10	0.016867

dtype: float64

We can see that among the variance of 10 principal components, the variance of PCs_1 and PCs_2 is the largest, so we choose these two principal components. And we plot these two principal components using the scatter graph (figure 4.1).

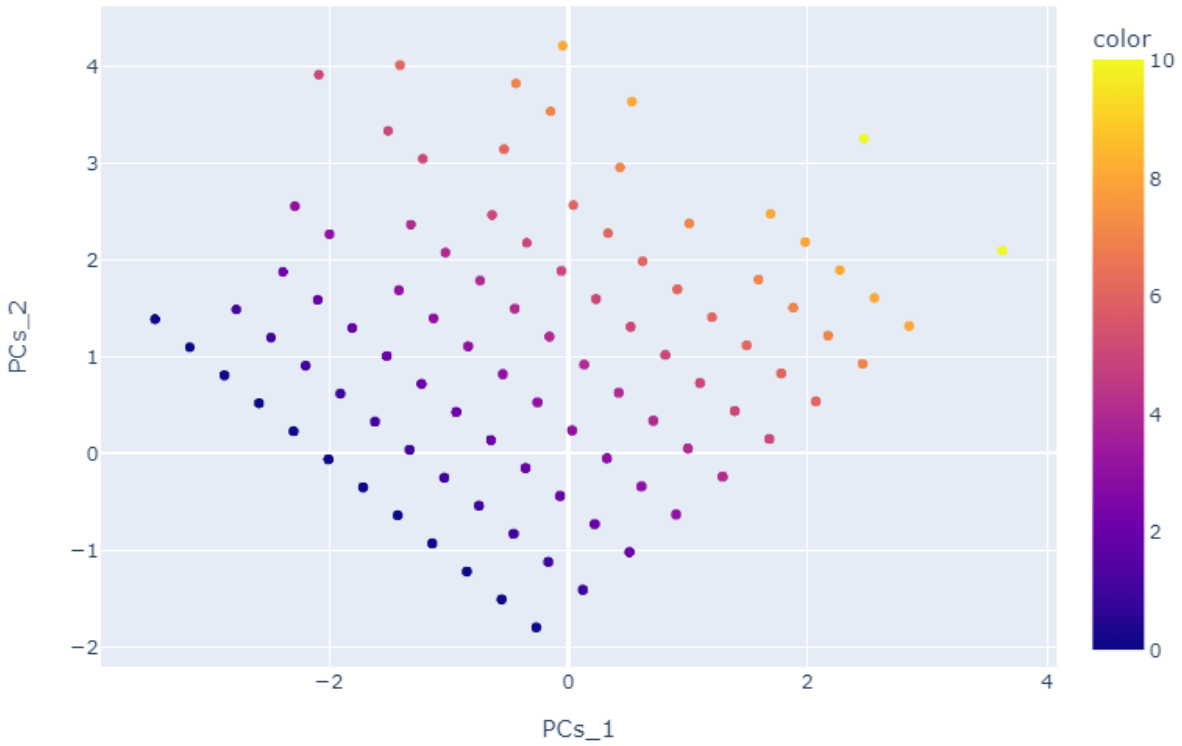

figure 4.1

Before we do the PCA on our data, we do the normalization to the data. If the original data have high standard deviation then it will make the weight match more than low standard deviation data, so to avoid this situation the data after normalization will have the same standard deviation and weight.

5. Task 5

	eu1	eu2	eu3	eu4	eu5	eu6	eu7	eu8	eu9	eu10
eu1	0	1.4	1.7	3.3	5.1	4.1	5.5	7.3	8.3	8.8
eu2	1.4	0	1	3	4.2	3.3	5.1	6.2	7.3	7.8
eu3	1.7	1	0	3.5	4.6	3.7	6.1	6.1	7.9	8.6
eu4	3.3	3	3.5	0	2.2	1.4	4.6	5.4	5.8	7.4
eu5	5.1	4.2	4.6	2.2	0	1	5.1	3.5	4.1	6.4
eu6	4.1	3.3	3.7	1.4	1	0	4.6	4.1	4.7	6.5
eu7	5.5	5.1	6.1	4.6	5.1	4.6	0	7.9	5.2	4.4
eu8	7.3	6.2	6.1	5.4	3.5	4.1	7.9	0	5	7.6
eu9	8.3	7.3	7.9	5.8	4.1	4.7	5.2	5	0	3.5
eu10	8.8	7.8	8.6	7.4	6.4	6.5	4.4	7.6	3.5	0

Figure 5.1

Figure 5.1 shows the Euclidean distance of 10 randomly selected objects from each other in a table

	cos1	cos2	cos3	cos4	cos5	cos6	cos7	cos8	cos9	cos10
cos1	1	1	0.9	1	0.8	0.9	0.7	0.6	0.6	0.5
cos2	1	1	1	1	1	1	0.8	0.8	0.8	0.6
cos3	0.9	1	1	1	0.9	0.9	0.6	0.8	0.7	0.5
cos4	1	1	1	1	1	1	0.8	0.8	0.8	0.7
cos5	0.8	1	0.9	1	1	1	0.8	0.9	0.9	0.8
cos6	0.9	1	0.9	1	1	1	0.8	0.9	0.9	0.8
cos7	0.7	0.8	0.6	0.8	0.8	0.8	1	0.6	0.9	0.9
cos8	0.6	0.8	0.8	0.8	0.9	0.9	0.6	1	0.9	0.7
cos9	0.6	0.8	0.7	0.8	0.9	0.9	0.9	0.9	1	0.9
cos10	0.5	0.6	0.5	0.7	0.8	0.8	0.9	0.7	0.9	1

Figure 5.2

Figure 5.2 shows the Cosine distance of 10 randomly selected objects from each other in a table

	ma1	ma2	ma3	ma4	ma5	ma6	ma7	ma8	ma9	ma10
ma1	0	3.54536	3.88854	3.27927	4.05744	2.64617	3.35897	3.68805	3.20146	3.92189
ma2	3.54536	0	3.84968	3.44792	2.76668	3.60055	3.60889	3.74461	3.63867	3.49881
ma3	3.88854	3.84968	0	3.15011	3.55611	2.85914	3.12972	3.18642	4.07942	3.46484
ma4	3.27927	3.44792	3.15011	0	1.64455	1.24966	2.12921	3.33291	2.64587	3.64845
ma5	4.05744	2.76668	3.55611	1.64455	0	2.19406	2.86415	3.0018	2.57903	3.46633
ma6	2.64617	3.60055	2.85914	1.24966	2.19406	0	2.23275	2.46728	1.94661	3.16677
ma7	3.35897	3.60889	3.12972	2.12921	2.86415	2.23275	0	4.03122	2.71989	2.37433
ma8	3.68805	3.74461	3.18642	3.33291	3.0018	2.46728	4.03122	0	2.4943	3.3525
ma9	3.20146	3.63867	4.07942	2.64587	2.57903	1.94661	2.71989	2.4943	0	2.4147
ma10	3.92189	3.49881	3.46484	3.64845	3.46633	3.16677	2.37433	3.3525	2.4147	0

Figure 5.3

Figure 5.3 shows the Mahalanobis distance of 10 randomly selected objects from each other in a table

In terms of the representation of the line graph, it is not doable to draw because the object has five dimensions. Therefore it is hard to judge which dimensions are suitable for the x,y axis.

The Euclidean distance represents the physical distance among the 10 objects
The cosine distance represents the change among the 10 objects.
The Mahalanobis distance represents the correlation among the 10 objects.
Therefore the mahalanobis distance is suitable for the similarity measurement because we need to measure the correlation among objects.

6. Task 6

6.1 label

Given the distance in task 5, Mahalanobis would be a better criteria to label the dataset since it represents the correlation of each other.
The data is complete, therefore no need to do the preprocessing.
In order to have a distance attribute, we need a standard sample to calculate the distance. So we take the mean vector for all the objects in our target table.
First we calculate the distance between all the objects and the standard sample. We check the distribution of the ‘distance’. Because this table is the statistical data of the game.
Then, the label can be the performance of a player in a particular game. We know that in 1 game there are 10 players but only a few of them have good performance.

So the distance of the good performance has a bigger distance from the standard sample.
Finally, we define the performance from “C” to “A” with respect to the distance from small to large (figure 6.1). The values of different label are based on the distribution of the distance.

```
def label_class (row):
    if row['distance'] <= 1.836:
        return 'C'
    if row['distance'] <= 2.4106:
        return 'B'
    return 'A'
```

figure 6.1

6.2 Decision Tree Classifier

Tool: Python library - tree in sklearn.

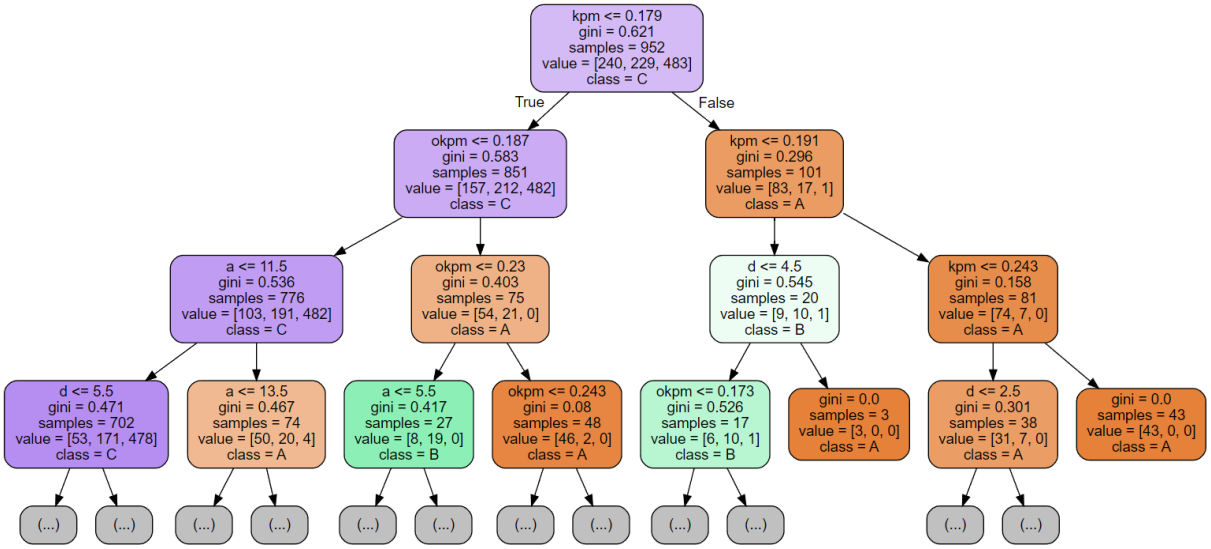

figure 6.2

Performance and Accuracy:

```
accuracy:0.8361344537815126
depth of the tree:20
the number of leaves:165
```

Figure 6.3

The performance of the decision tree is good. When training the model, we didn’t constrain the depth and number of leaves in the configuration. And we use the “gini” criteria inside of “entropy” to prevent the overfitting. The accuracy can be varied from 0.75 - 0.84.

6.3 Rule Based Classifier

Tool: Python
Since there is no mature rule based classifier so we build one.
Steps of the classifier:

1. First, separate the train data set by classes. The size of the dataset is from ‘A’ to ‘B’ then to ‘C’.

```
Xtrain_A = Xtrain.loc[Ytrain == 'A']
Xtrain_B = Xtrain.loc[Ytrain == 'B']
Xtrain_C = Xtrain.loc[Ytrain == 'C']
print(Xtrain_A.shape)
print(Xtrain_B.shape)
print(Xtrain_C.shape)
```

```
(233, 5)
(241, 5)
(478, 5)
```

2. The rule will be built from the smallest class to the largest class. Based on the size of the class, we first build the rule for ‘A’. Then, we build the rule for ‘B’. At last we build the size for ‘C’. All rules are based on the value of columns.
3. Apply the rules on the dataset in the sequence: rule of A, rule of B, rule of C. Then compare the prediction results with its classes to get the accuracy.

```
#apply the rule
apply_rule(Xtest, ruleA, ruleB, ruleC)
#calculate the accuracy
accuracy_number = len(Xtest.loc[Ytest==Xtest['performance']])
accuracy = accuracy_number / len(Xtest)
print(accuracy)
```

```
0.7352941176470589
```

The performance of the decision tree is not good enough compared with the Decision Tree classifier. The accuracy of the rule based classifier is lower than the decision tree classifier by 10% on average.The rule based classifier doesn’t have a mature tool in the machine learning library. Because all the rules are conditional dependency with the dataset. Thus, the complexity and the sequence of the rules matter. The design of the rule will also influence the performance.

6.4 Comparison

Compared with the decision tree classifier and the rule based classifier, the accuracy of the decision tree classifier is better than the rule based classifier. Because of the dataset we have 5 pure numerical attributes.
In the decision tree classifier, it is easier to narrow down the scope of a label. But the model can be overfitting and the depth of the tree may be too large for some large dataset. Luckly, our dataset only has about 1200 objects, so we don't need to worry about overfitting in our case. And this problem can be avoided by configuring the parameter in the classifier.

However, in a rule based classifier, the rules will be a larger rule set if we want to get a good training accuracy. But this will lead to low accuracy in the test set. So this may cause underfitting. The rule based classifier will have a better performance if the attributes are categorical. But the attributes here are all numerical. On the other hand, the sequence of the rules matters. It needs much more time to check the best sequence of the rules. So the rule based classifier is not suitable for our dataset.

7. Task 7

No, our initial five questions were not addressed. There are three of our initial five questions that included champion strength issues, in our project, we saw the total number of hero appearances and win rate total. Some champion's win rate total can even reach 100%, but this is not enough to prove that this champion has advantages in winning the game. Because when we look at these champions with particularly high win rate totals, we see that they have very few appearances, so this unilateral information does not help us answer any of the initial five questions about the champion. This issue is also reflected in our remaining two initial questions.

New question:

1. How to rank the champion strength?.
2. What is the best champion for every player?
3. In the MVP criteria, which columns have the biggest influence?
4. Which attribute influences the game result most?
5. What is the counter relation of the champions?

Tool of selection

All the tasks are done by Python on google colab.