

**EECS 4412**

**Project Part 4: Phase 2**

**Instructor: Habib-ur Rehman**

[**1. Task 1 Association Analysis**](#_uax0i5w0eijz) **3**

[1.1 Table selection](#_lrpacw4rw4yh) 3

[1.2 Discretization](#_a8di5lpcx4wv) 3

[1.2.1 dimension ‘k’: Equal frequency approach](#_hxof3f3muo35) 3

[1.2.2 dimension ‘d’: Equal Interval approach](#_4633u1pud5q3) 3

[1.2.3 dimension ‘a’: K-means approach](#_89x2czj72ao9) 4

[1.3.4 a tabular form for the first 10 rows](#_i6jvdqtrht99) 4

[1.3 Itemsets](#_5kcd9j6bmesr) 4

[1.3.1 10 most frequent item sets](#_tzje35fnakpy) 4

[1.3.2 10 association rules](#_b65cmjc9otqi) 5

[1.3.3 Five rules with the most confidence](#_fnqq58g24yn3) 5

[1.3.4 Five rules with the least confidence](#_lgytu5v719xv) 5

[1.3.5 Measures of the 5 rules with the most confidence](#_b22nujumnq47) 6

[1.3.6 Z - statistic](#_lq61y99drel8) 6

[**2. Task 2 Clustering Analysis**](#_fpuknfx56fxh) **7**

[2.1 Preprocessing](#_pt9n1khxod65) 7

[2.2 Clustering](#_7sz00s8nb0l0) 9

[2.2.1 K-mean](#_czlrpg66kk1p) 9

[2.2.2 Classing attribute](#_fjjn4ugxtaty) 11

[2.2.3 Best k value](#_3n23m71dxi6z) 11

[**3. Task 3 Classification Revisited**](#_cfn7x6r4jacl) **12**

[3.1 Naive bayes classifier](#_2glzbysf897p) 12

[3.2 3-fold cross validation](#_pesup1kn9c3w) 12

[3.2.1 The measures for the first fold as test data:](#_oj8x4l4sju) 13

[3.2.2 The measures for the second fold as test data:](#_dx98zzp9wc9f) 13

[3.2.3 The measures for the third fold as test data:](#_t91aqzhfcn32) 14

[3.3 ROC curve](#_9pwnrs3wqbb6) 14

[3.3.1 The first fold as test data:](#_k9jlc1y0vo0u) 15

[3.3.2 The second fold as test data](#_plvuk4y3nfcl) 15

[3.3.3 The third fold as test data](#_kffi45fv3w7t) 16

[3.3.4 Analysis of the ROC curves:](#_xfjiz7ar5rr0) 16

[**4. Task 4 Progress on your Objectives**](#_n57gl1o7la9p) **16**

[**APPENDIX**](#_hyx9clkdv9ga) **18**

# 

# 1. Task 1 Association Analysis

## 1.1 Table selection

Because we still want to explore the performance of the players. At the same time, the table “players” are mainly made of continuous values. We choose simply just “kill”, “death”, “assistant” dimensions. However, before we select 1000 rows, we have to drop the rows with outliers and also drop those rows that have missing values.

## 1.2 Discretization

The discretization can be done by the “KBinsDiscretizer” in sklearn library. For the parameters in the discretization process:

We set the encode as “ordinal”, which will return the bin identifier encoded as an integer value. Because the values in our selected datasets are pure integers.

For different approaches, we will change the “strategy” parameter to achieve different discretization methods.

### 1.2.1 dimension ‘k’: Equal frequency approach

For the equal frequency approach, the **number of items** in every bin should be equal to each other. Set the “strategy” parameter to “quantile” (by default it is “quantile”). Because the kill is the most important for the analysis, we need 4 bins to collect all the items.

The edges for this bin:



**figure 1.1**

| **bin** | Least KILL | Less KILL | More KILL | Most KILL |
| --- | --- | --- | --- | --- |
| **range** | [0,1] | (1,2] | (2,4] | (4,9] |

### 1.2.2 dimension ‘d’: Equal Interval approach

For the equal Interval approach, the **range interval** in every bin should be equal to each other. Set the “strategy” parameter to “uniform”. We separate the assistant value into 3 bins.

The edges for this bin:



**figure 1.2**

| **bin** | Least DEATH | Medium DEATH | Most DEATH |
| --- | --- | --- | --- |
| **range** | [0,2] | (2,5] | (5,8] |

### 1.2.3 dimension ‘a’: K-means approach

For the equal frequency approach, the bin size should be assigned by **k-means**. Set the “strategy” parameter to “kmeans”. We separate the assistant value into 3 bins.

The edges for this bin:



**figure 1.3**

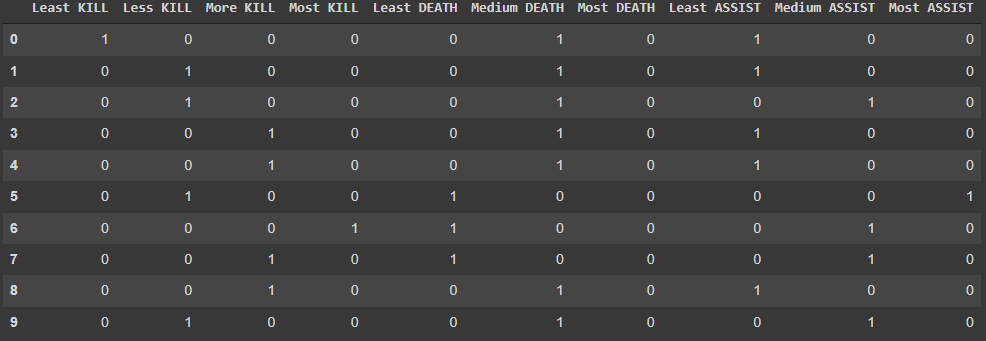
| **bin** | Least ASSIST | Medium ASSIST | Most ASSIST |
| --- | --- | --- | --- |
| **range** | [0,5] | (5,10] | (10,17] |

### 1.3.4 a tabular form for the first 10 rows

For every column, we separate them into n columns where n is the number of bins. And in these n bins for a dimension, only 1 bin will be set to ‘1’.

For example, the kill value is 5 which is supposed to be in the last bin “Most KILL”. Then the value of “Most KILL” will be 1. The others (“Least KILL”,“Less KILL”,”More KILL”) will be set to 0.

Tabular form for the first 10 rows:



**figure 1.4**

## 1.3 Itemsets

After the discretization the possible maximum itemsets will be 3. And the 1-itemset doesn’t make sense in analysis. So here we only focus on the 2-3 itemsets in the analysis.

### 1.3.1 10 most frequent item sets

With using the **apriori** algorithm APT in the mlxtend.frequent\_patterns library. We can get the most frequent itemsets in figure 1.5. However, there are only 2-itemsets. It seems like no 3-itemsets have larger support.



**figure 1.5**

### 1.3.2 10 association rules

Tool: **association\_rules** in mlxtend.frequent\_patterns library.

However, this tool can only generate all the itemsets, but we can get all the association rules then keep the rules with support bigger than 0.126 (which is the minimal support in the 10 most frequent itemsets).

### 1.3.3 Five rules with the **most** confidence

The figure 1.6 shows the 5 rules with most confidence. The rules are:

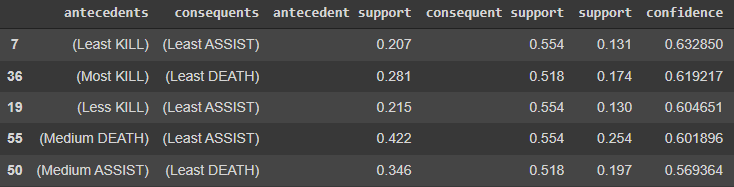
(Least KILL) -> (Least ASSIST)

(Most KILL) -> (Least DEATH)

(Less KILL) -> (Least ASSIST)

(Medium DEATH) -> (Least ASSIST)

(Medium ASSIST) -> (Least DEATH)



**figure 1.6**

### 1.3.4 Five rules with the **least** confidence

The figure 1.6 shows the 5 rules with least confidence. The rules are:

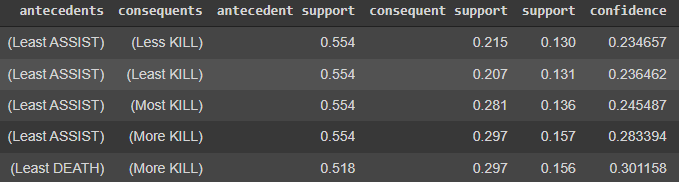
(Least ASSIST) -> (Less KILL)

(Least ASSIST) -> (Less KILL)

(Least ASSIST) -> (Most KILL)

(Least ASSIST) -> (Most KILL)

(Least DEATH) -> (Most KILL)



**figure 1.7**

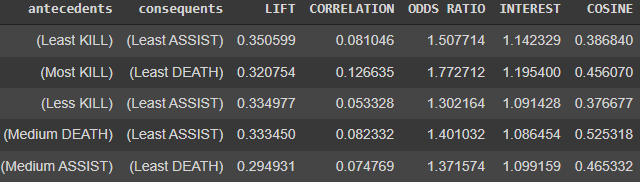
### 1.3.5 Measures of the 5 rules with the most confidence

Here we perform 5 measures:

1. lift for rule
2. correlation
3. Odds ratio
4. Interest
5. Cosine

First we calculate the f11, f10, f01, f11 for the rules by taking the antecedents as X and the consequent as Y.

Then calculate the measure based on the formula. The result is shown in figure 1.8:



**figure 1.8**

### 1.3.6 Z - statistic

It is possible, but it may not be recommended.

The Z - statistic is to predict the result based on the known dataset and its null hypothesis. But the probability is calculated in normal distribution. This is different from what we understand in this dataset. In this section, we mainly analyze the performance of the players. In every match, there is usually only 1 or 2 players that have outstanding performance. There usually is not going to be a bad performance player. So clearly, the prediction result will not be good enough.

# 2. Task 2 Clustering Analysis

## 2.1 Preprocessing

Before performing K-mean clustering on the data, it is necessary to perform certain preprocessing on the data. There will be a total of 4 types of preprocessing that require performance.

The first point is that K-means is a technique that calculates the distance between data to determine the similarity between points, so due to this reason, K-means cannot handle any type of variables except Numerical variables. If it must deal with categories or a mixture of categories and numerics, it can use techniques such as K-modes or K-prototype, but this time we only focus on K-means clustering, so we choose 7 dimensions from our wc\_player dataset and the data of these 7 dimensions are all numerical variables.

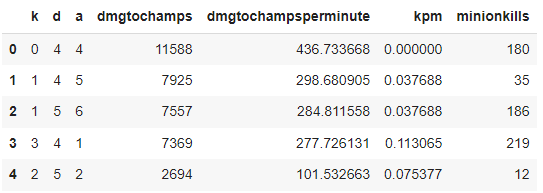


Figure1: 7 columns numerical data

After selecting the data, it is time to further process, because K-means is similar to an algorithm of the mean, and the mean is easily affected by the extreme values, and these extreme values will come from noises and outliers, so the next step is to eliminate noises, outliers and missing values ​​in our selected data. The first is to deal with outliers. In the last part of the project, we learned a very efficient way to deal with outliers, which is z-score standardization. We set the z-score threshold to 3, which means that 99.7% of the data will be saved. Then the remaining 0.3% of the outliers data will be dropped. After processing outliers we will get a relatively smooth dataset, but K-means will also be affected by missing values, so we apply the dropna() method to our selected data, which is to drop all rows with missing values ​​in the dataset. Because the above processing of these outliers and missing values may lead to a decrease in the instance because of the dropping data, so after we complete the above processing, then we limit the dataset we selected to 1000 rows.



Figure2: Handle outliers and missing values

The third point is that if you want K-means clusters to have sufficient accuracy, what you need to ensure after removing outliers is to keep all the data at the same scale. This time we used a technique that was used in the previous part of the project, which is normalization. After processing our data using normalization, the scale of our data will be unified between 0 to 1, which ensures that all data in each dimension are treated equally.

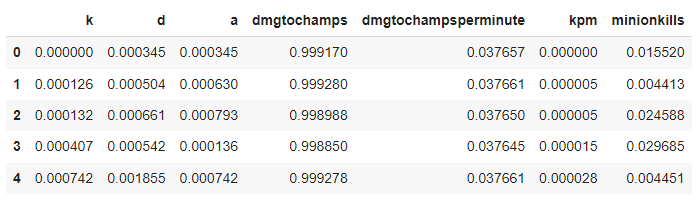


Figure3: The data after normalization

The fourth point, K-means clustering is applied to the data with 2 dimensions, and we choose 7 dimensions at this time, so we need to apply another technique used in the previous part of the project, which is to use principal component analysis(PCA) to our dataset to reduce the dimensions. After applying principal component analysis(PCA) to our dataset, we can see that k, d contains almost all the elements from the dataset in the explained variance plot, so we finally chose k and d as the data input for our K-means clustering analysis.

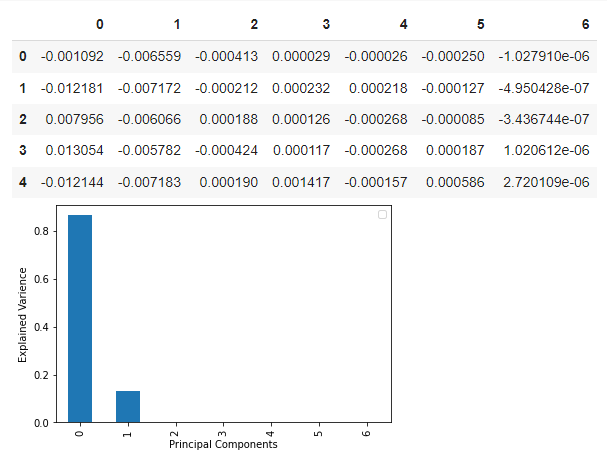


Figure3: Principal component analysis(PCA)

## 2.2 Clustering

### 2.2.1 K-mean

k-means clustering is a vector quantization method whose main purpose is to plan the observed n values into k different clusters. Each of these observed values has the closest mean to its own cluster, and these observed values are grouped together to form a cluster. In this part, we will implement K-means clustering analysis. After the preprocessing, we determined to use k(kill) as rows and d(dead) as columns to form a new table to implement K-means clustering analysis. We performed three experiments with cluster values (k) of 3, 4 and 5 in order to verify the effect of cluster value (k) on the results.

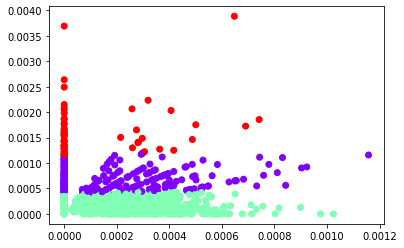


Figure4: K-mean, k = 3

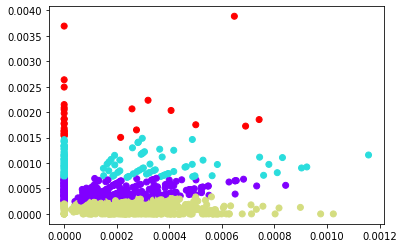


Figure5: K-mean, k = 4

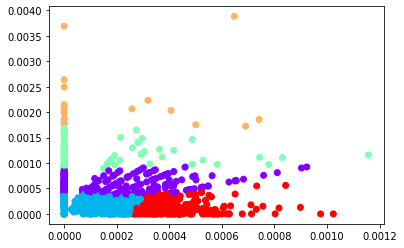


Figure6: K-mean, k = 5

### 2.2.2 Classing attribute

After performing K-means clustering on the data, we classified the original data according to the result of clustering. We classify each instance and assign the cluster category of the instance to class labels called Clusters.

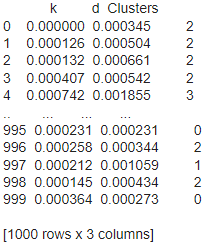
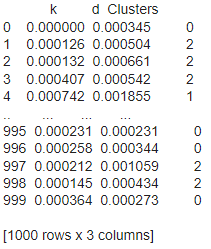


Figure7: Class attribute, when k = 3 Figure8: Class attribute, when k = 4

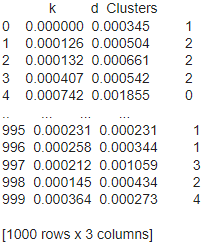


Figure9: Class attribute, whenk = 5

### 2.2.3 Best k value

Sum of squared errors(SSE) is the technique used to sum the squared differences between each observation and its group's mean, so it is perfectly determined whether the clustering performs well or not. We did three kinds of K-means clustering when the k value was 3, 4 and 5, and then we calculated the SSE values respectively.



Figure10: SSE for three k values

After we get three SSE values, through comparison, we can find that when the k value is larger, the SSE also decreases. When the k value is equal to 5, the center error of the observation value for each cluster will naturally decrease as the number of clusters increases. Through the above comparison and summary, we can decide the best k = 5.

# 3. Task 3 Classification Revisited

This task we read the csv files from the previous task. The ‘T2Mod’ is taken as the features. The ‘T2Class’ is taken as the labels. Because the k = 5 gave us the best result. So the label here will be 0, 1, 2, 3, 4.

## 3.1 Naive bayes classifier

We use the Naive Bayes classifier in the sklearn library. Because our datasets are continuous, we want to use GaussianNB for our classifier. For the training step, due to the 3-fold cross validation, the will be fitted 3 times on different training data and validation data.

## 3.2 3-fold cross validation

With the help of the “KFold” library in sklearn, we split the dataset into 3 pieces. Every time it will return the indices for the training set and the testing set.

We fit the model on the training data, and get the prediction result on the testing data. The output is the following measures for the prediction result. Because this is a multi-class classification, the measures are different here. As demonstrated in figure 3.1.

1. The **True Positives (TP)** are the same with binary classification.
2. The **False Negative (FN)** will count all the cases that are predicted as other classes, but they actually are labeled as this class.
3. The **False Positive (FP)** will count all the cases that are actually labeled as other classes, but they are predicted as this class.
4. The **True Negative (TN)** will count all the cases that are not predicted as other classes and they actually do not belong to this class.

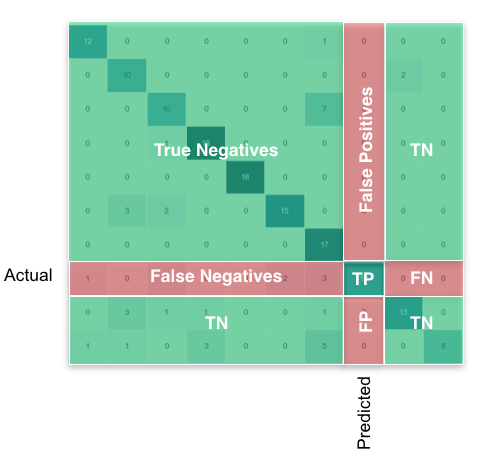


figure 3.1 confusion matrix

And these all can be calculated with the confusion matrix of the prediction and actual labels.

Then we calculate the **ACCURACY, RECALL, PRECISION, F-MEASURE**.

Code implementation for the measures as figure 3.2:

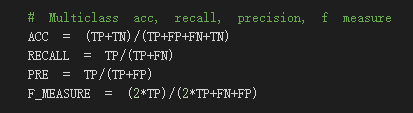


figure 3.2 Measures Calculation

### 3.2.1 The measures for the first fold as test data:

| **Class Labels** | **accuracy** | **recall** | **precision** | **f-measure** |
| --- | --- | --- | --- | --- |
| 0 | 0.970060 | 0.944444 | 0.944444 | 0.944444 |
| 1 | 0.970060 | 0.892857 | 0.986842 | 0.937500 |
| 2 | 0.994012 | 0.962963 | 0.962963 | 0.962963 |
| 3 | 0.997006 | 0.888889 | 1.000000 | 0.941176 |
| 4 | 0.955090 | 0.975806 | 0.909774 | 0.941634 |

### 3.2.2 The measures for the second fold as test data:

| **Class Labels** | **accuracy** | **recall** | **precision** | **f-measure** |
| --- | --- | --- | --- | --- |
| 0 | 0.970060 | 0.925926 | 0.949367 | 0.937500 |
| 1 | 0.958084 | 0.849462 | 1.000000 | 0.918605 |
| 2 | 0.991018 | 0.961538 | 0.925926 | 0.943396 |
| 3 | 0.997006 | 1.000000 | 0.833333 | 0.909091 |
| 4 | 0.958084 | 1.000000 | 0.902098 | 0.948529 |

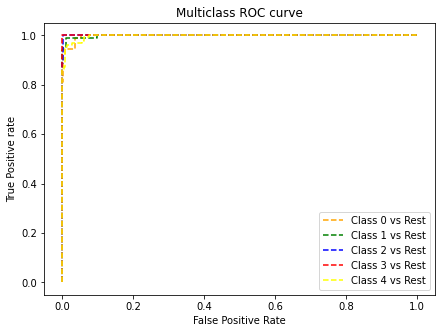
### 3.2.3 The measures for the third fold as test data:

| **Class Labels** | **accuracy** | **recall** | **precision** | **f-measure** |
| --- | --- | --- | --- | --- |
| 0 | 0.973054 | 0.977273 | 0.924731 | 0.950276 |
| 1 | 0.967066 | 0.851351 | 1.000000 | 0.919708 |
| 2 | 0.991018 | 0.968750 | 0.939394 | 0.953846 |
| 3 | 0.997006 | 1.000000 | 0.833333 | 0.909091 |
| 4 | 0.976048 | 0.985185 | 0.956835 | 0.970803 |

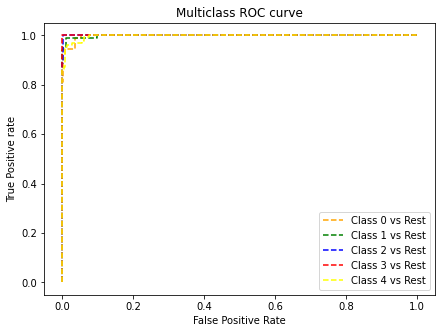
## 3.3 ROC curve

Again, because this is a multi-class classification, one graph will contain 5 ROC curve for different classes.

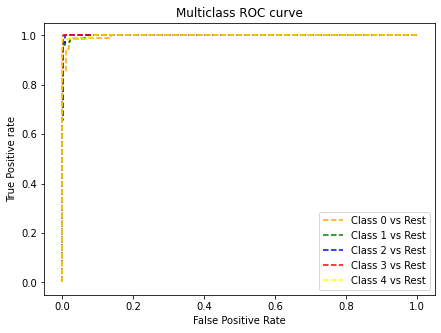
### 3.3.1 The first fold as test data:



### 3.3.2 The second fold as test data



### 3.3.3 The third fold as test data



### 3.3.4 Analysis of the ROC curves:

These training datasets are preprocessed. The missing data are opted out. The values are normalized. And we picked the columns on the PCs with the most information. So the clustered groups are very representative. So that the model built on such data will perform well. The ROC curves are all close to top-left, this shows our clustering and classification are doing well.

# 4. Task 4 Progress on your Objectives

Our original questions were:

1. What champions have advantages in winning the game?

This question might need to be adjusted, since our work is more focusing on the performance of individual players rather than the champions that players use

2. In the ban-pick phase, find out the champions that need to be banned based on the opponents.

We found out that the ban-pick phase has a specific order of ban, that is (ban1, ban2, ban3) is different from (ban2,ban1,ban3). Therefore, we discarded the analysis of ban

3. How important does the first tower matter to the win?

The first tower seems too specific to do the data mining. It is just a single column among a total ninety-one columns, thus it is trivial to find the meaning of the first tower.

4. Specify team members on the red or blue sides have a higher win rate, and what champion?

The winning rate of each individual team is a trivial task because it is already provided. Moreover, the winning rate might have bias

5. Which champion can be more powerful in the early game stage?

We might turn this question into the player’s favorite champion which adjusts the angle to the players rather than champions.

Overall, majority of our initial questions need to be adjusted, so we formulate the following questions:

1. What players have the best ratio of kills, deaths, and assists (KDA)?
2. How would the players’ performance, such as the KDA value, affect the win rate of the team?
3. How to certify a player as an MVP in each game through this method？
4. In kill, death, assistant, damage per minute, minion kill, which one can be most dominant in performance evaluation.
5. How to judge the status of each player through this method, and can the team's first round players be adjusted based on this result?

# APPENDIX

Link to our dataset: https://www.kaggle.com/datasets/ilyadziamidovich/league-of-legends-world-championship-2019