# Machine Learning Clustering and Classification Project

Source Code: https://github.com/onurgurkangultekin/MachineLearningProject

## Introduction

We are given a series of bridge games dataset. For each game we have the certain information as feature values according to the played game. Regarding to this dataset we are requested to perform machine learning algorithms such as dimensionality reduction, clustering and non-parametric estimation. For implementing this project, I will use Python programming language. Because, it has a lot of helpful libraries and well documented on the internet.

## Dataset

We have 286 rows in the dataset which are data point rows. In a data row, there are 12 certain information about the played game as features. For a game as a row we have given time, team, player, point, distribution, zone, contract, color, counter, target, result and reward. This dataset contains string and time values which are not applicable for applying machine learning algorithms. Therefore, first step will be transforming data into numeric values using labels.

## Reading and Transforming Dataset

Firstly, I have used pandas library to read excel. There are 2 sheets on the excel book but the data that we are interested in the “Games” sheet. I have read all data rows from “Time” to “Reward” columns. Here are some features of first 6 data are:

| | Zaman | Takım | Oyuncu | Puan | Dağılım | Zon | Kontrat | Renk | Kontr |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 09:35:00 | BT | B | 24 | 3 | Yok | 3 | Kozsuz | Yok |
| 1 | 09:45:00 | AL | L | 22 | 3 | Yok | 2 | Kozsuz | Yok |
| 2 | 09:55:00 | AL | L | 22 | 2 | Yok | 2 | Kupa | Yok |
| 3 | 10:07:00 | BT | B | 23 | 6 | Yok | 4 | Maça | Yok |
| 4 | 10:17:00 | BT | T | 25 | 4 | Var | 4 | Kupa | Yok |
| 5 | 10:28:00 | AB | A | 22 | 3 | Yok | 3 | Kupa | Yok |

Then I have created mapping dictionaries in order to transform labels into numeric values. For example, for players I have created a key-value pair something like player mapping = {‘A’: 1, ‘L’: 2, ‘B’: 3, ‘T’: 4} or for target target mapping = {'Partial': 1, 'Zone': 2} and replaced the data that gathered from excel with other mappings so on. After that for converting “Time” values to numeric values I have created a formula: $value = (hours - 8)\ x\ 100 + \ minutes$. Now “Time” dimension is also numeric values and ready to apply machine learning algorithms. Here are again some features of first 6 data sample:

| | Zaman | Takım | Oyuncu | Puan | Dağılım | Zon | Kontrat | Renk | Kontr | Hedef |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 135 | 0 | 3 | 24 | 3 | 0 | 3 | 1 | 1 | 2 |
| 1 | 145 | 1 | 2 | 22 | 3 | 0 | 2 | 1 | 1 | 1 |
| 2 | 155 | 1 | 2 | 22 | 2 | 0 | 2 | 2 | 1 | 2 |
| 3 | 207 | 0 | 3 | 23 | 6 | 0 | 4 | 3 | 1 | 2 |
| 4 | 217 | 0 | 4 | 25 | 4 | 1 | 4 | 2 | 1 | 2 |
| 5 | 228 | 2 | 1 | 22 | 3 | 0 | 3 | 2 | 1 | 1 |

## Normalizing Data

Now, we have converted our data to numeric values. But there is an important concern here again. We have different feature values on different unit scales. Normalization of ratings means adjusting values measured on different scales to a notionally common scale, often prior to averaging. For example target feature has a binary value. Player feature values have a range between 1-4. Reward feature has a range -800 to 2500 of value. So we should normalize the data points in order to scale different value ranges. In my application I will use min-max scale from sklearn preprocessing library which transforms features by scaling each feature to a given range [0-1]. This estimator scales and translates each feature individually such that it is in the given range on the training set, i.e. between zero and one.

$$X\_std = (X - X.min) / (X.max - X.min)$$

$$X\_scaled = X\_std * (max - min) + min$$

This transformation is often used as an alternative to zero mean, unit variance scaling (standart scaling).

## Dimensionality Reduction

Now is our data ready to apply machine learning algorithms? Not yet. Because our data set has 12 features which some of them may be correlated to each other and many of them may not be necessary to use. Maybe a few of these features are enough to understand the what is data and how to cluster it. Besides that, implementing machine learning algorithms on our data which has 12 features is time and memory consuming. Also, most of time it might be very hard to see which features are important and useful for clustering by checking data. Other then these reasons, we also have another really important reason why we should reduce dimension count. Visualizing! If we extract 2 or 3 important features then we will be able to visualize the data that we are able to see the anomalies, clusters, relationships between features and so on.

Firstly if we think about played bridge games, it can be easily said that “time” feature is not much effective as other features. This feature is something like “id” column which does not support any important idea for clustering or classifying the data. Therefore we can drop this column before we start to apply machine learning algorithms on our dataset. After dropping the time column we have 11 columns to consider applying PCA algorithm.

## Principal Component Analysis (PCA)

Our mission is to get rid of correlated features and extract new features that linearly uncorrelated named principal components. These components should be orthogonal to each other which means that have 90 degrees between each other. The number of principal components should be less than the original feature count for observation purposes. We will use PCA from sklearn decomposition library. Goal of the PCA is to create new dimensions which have variance as possible as high using by the original dimensions. After finding the first principal component which has the highest variance, we should find the second principal which has the second highest variance and so on. This algorithm is based on unsupervised learning and does not rely on output labels. Basically, the formula is $\boldsymbol{z = w^T \cdot x}$. Z is new component. w is eigenvector that used for projection on x features. x is our original data that we are willing to reduce its dimension count. Eigenvector w can be calculated by maximizing variance of z.

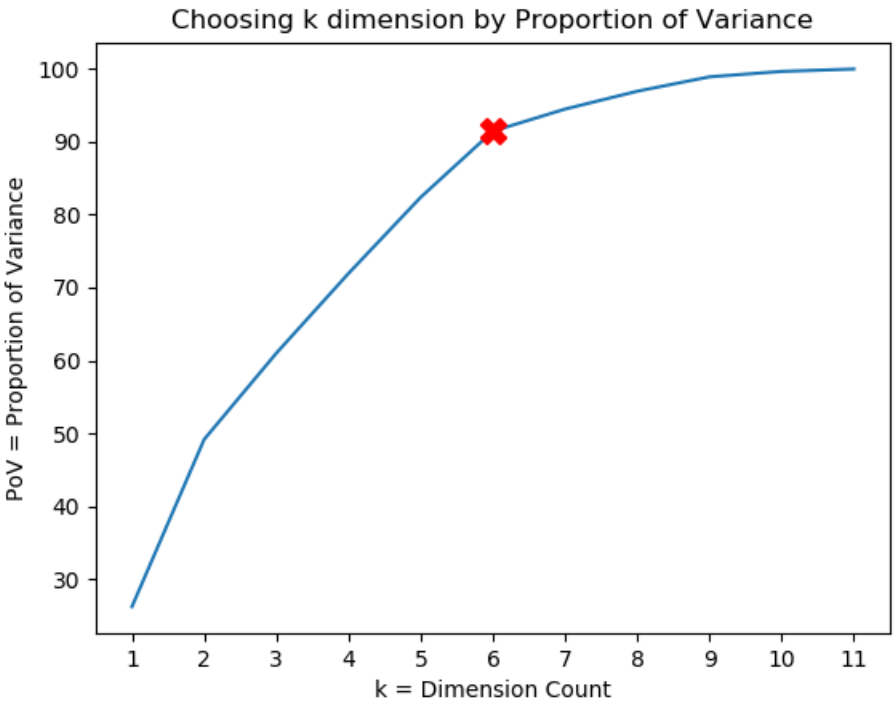
### Choosing Dimension Count k

Choosing k the dimension count is another important problem. Because it determines both complexity and accuracy of model. With original feature set we have a specific variance value. Using PCA we are trying to create new dimensions which has high variance. But how high do we want?

We will define a threshold value in order to stop finding new components. This method's name is PoV (proportion of variance). In our application I have chosen %90 variance ratio to stop extracting new dimensions. $\frac{\lambda_1 + \cdots + \lambda_k}{\lambda_1 + \cdots + \lambda_d} > 0{,}90$. First principal component x-axis gives about %26 variance ratio. Second principal gives about %23 variance ratio. As we calculate cumulative result we get around %91 variance ratio with 6 principal components. Therefore, we exactly choose k=6.

### Applying PCA (Principle Component Analysis)

As you see in the below graph we already reach eighty percent variance ratio with 5 principal component. But according to our defined variance ratio this is not enough to stop. So we calculate the dimension count as 6 principal components and get a cumulative ratio as %91. That's why we choose the dimension count of k is 6.

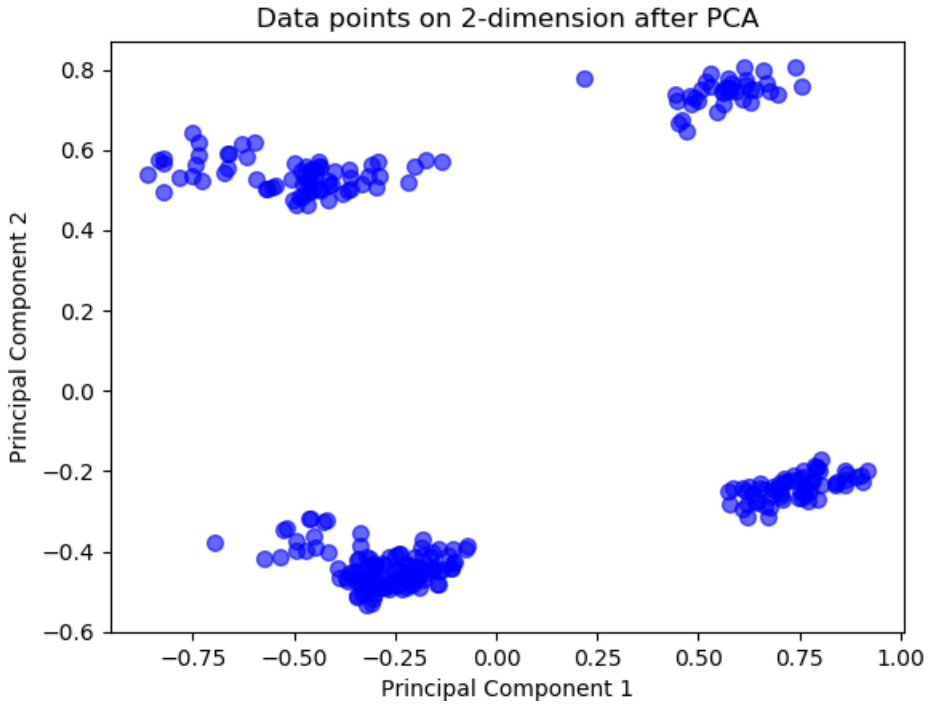



Now we can transform the original 12-featured data to 6-featured data using by new 6 principal components. After that we are able to visualize how the data seem and what should be the clusters and so on by using first 2 or 3 components. As we will see there are obvious 4 clusters separated on PC-1 and PC-2. Here are the Eigenvector values that have been created based on PCA algorithm. Eigenvector values give the weights on features respectively:

```
eigen vector 1
[ 0.10955293  0.09775272 -0.10321895 -0.04536442 -0.18036621 -0.20132344
  0.22190874 -0.27524505 -0.87588337 -0.00202889 -0.02209648]
eigen vector 2
[ 0.05246544 -0.04858499 -0.03743848  0.01224047  0.97284279 -0.01252573
  0.06825217  0.06211738 -0.19458703 -0.0186077  -0.00530346]
eigen vector 3
[-0.14677643 -0.8875441   0.01887796 -0.11495788 -0.02756132 -0.12733043
 -0.35443584 -0.13037646 -0.12782786  0.03410974  0.00718145]
eigen vector 4
[ 0.76985718 -0.35510623  0.01484838  0.05120788 -0.09860774  0.1563708
  0.37071583  0.31482217  0.03209875 -0.08053539 -0.01191441]
eigen vector 5
[-0.57528349 -0.25986171 -0.0695312   0.13743653 -0.04158781  0.20293175
  0.71956242  0.12291751  0.00649896 -0.03449328 -0.0278525 ]
eigen vector 6
[-0.18172318  0.06121916 -0.2007675  -0.00854308 -0.09091139 -0.0994589
 -0.27195448  0.83911601 -0.28017825 -0.20197196 -0.08557999]
```

If we examine the first principal component; the dominant feature is “target” with about -0.87. Second important feature is “counter” with about -0.28 because while applying the PCA operation, we consider selecting the features which have most variance. Second principle component is mostly relevant to

“zone” feature. Third one is about “player” and the forth one is relevant to “team” feature and so on. If we visualize the data on first 2 principal components, we see the graph below:



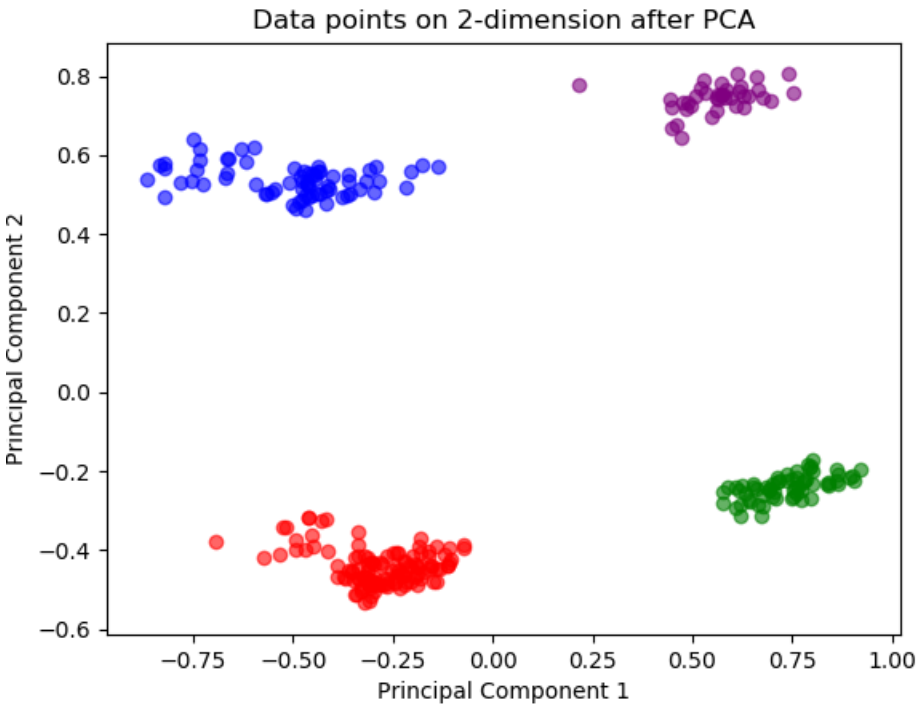

## Why Data Points Separates on Graph

If we click on the data points on the graph we can see the original feature values on the console application developed using on pick method.

```
Zaman Takim Oyncu Puan Dagilim Zon Kontrat Renk Kontr Hedef Sonuç Ödül
[datetime.time(12, 55) 'BT' 'T' 15 5 'Evet' 5 'Karo' 'Var' 'Zon' -2 -500]

Zaman Takim Oyncu Puan Dagilim Zon Kontrat Renk Kontr Hedef Sonuç Ödül
[datetime.time(11, 5) 'BL' 'B' 23 2 'Evet' 1 'Kupa' 'Yok' 'Partial' 0 30]

Zaman Takim Oyncu Puan Dagilim Zon Kontrat Renk Kontr Hedef Sonuç Ödül
[datetime.time(13, 15) 'LT' 'T' 21 2 'Hayir' 2 'Kozsuz' 'Var' 'Zon' 0 140]

Zaman Takim Oyncu Puan Dagilim Zon Kontrat Renk Kontr Hedef Sonuç Ödül
[datetime.time(9, 15) 'BT' 'T' 21 2 'Hayir' 2 'Karo' 'Yok' 'Partial' -1 -50]
```

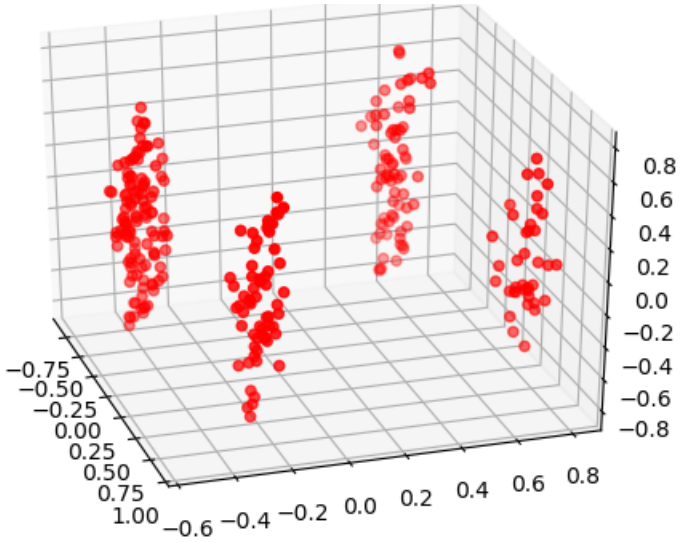
Upper left data points are whose target is zone and zone is true. Upper right data points are whose target is partial and zone is true. Because as we said earlier; on PC-1 dominant feature is “target”. So PC-1 separates the data regarding “target” values as zone and partial. Lower left data points are whose target is zone and zone is false. Lower right data points are whose target is partial and zone is false because; on PC-2 dominant feature is “zone”. So PC-2 separates the data regarding “zone” values as true and false. Let’s color the data points on the graph regarding to target and zone values. Blue colored points are which ones have target is zone and zone is true. Purple colored points are which ones have

target is partial and zone is true. Red colored points' target is zone and zone is false. Green colored points' target is zone and zone is false.




## What if we choose 3 dimensions instead of 2 dimensions to visualize?

If we try to visualize our data points in the 3-dimensional we can recognize that there are still 4 clusters in the space. Because the first principal component separates data into 2 different ranges and the second principal component separates the data another 2 different range so 4 clusters take place.



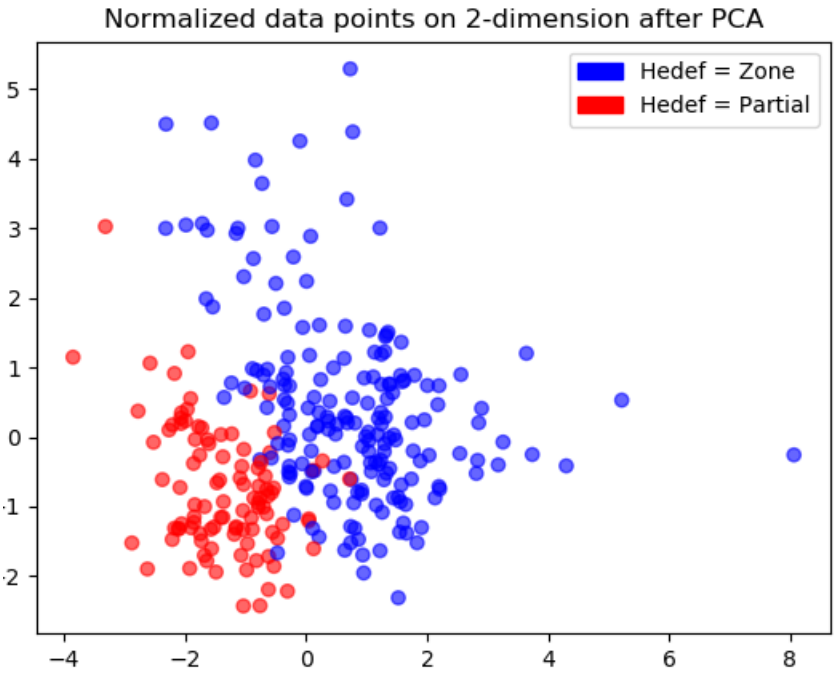

As you see the 3rd principal component does not affect the data points on aiming clustering. It is again just 2 components PC1 and PC2 which creates great variance to observe clusters. 3rd principal

component just gives us better visualizing and bring a little bit closer to original data points that we could not identify by graphics.

## Normalize data with Standard Scaling

Here is another interesting graph if we apply data normalizing by standard scaling before PCA we can inference another graph that might be attractive. Normalizing process is a method that reduces anomalies between various features. Features might be time, label, Boolean, or numeric that includes negative and positive values. Applying normalization scales data more common and near to each other. Standard Scaling makes mean = 0 and variance = 1. It allows the comparison of corresponding normalized values and also reduces the effect of anomalies.

In this graphic we firstly applied standard scaling the data and after that apply PCA transform and fit the data. It gives us much closer data points that blocks clustering. But even so it might give information when we aggregate a specific feature points with normalized and PCA applied data points. As you see on the graph on both of 2 axis higher values tend to target = zone rather than target=partial. It may give us information that these two axes correlates to Target feature highly.




## Clustering Data

Until now, we have converted the data values into numeric values. We normalized the data by scaling values between 0 and 1. We dropped unnecessary columns such as time feature. We applied PCA in order to decrease the dimension count from 12 to 6. We visualized the 2d data regarding first 2 principal components. Now we have new data points on new components that represent original data. We are ready cluster the data.

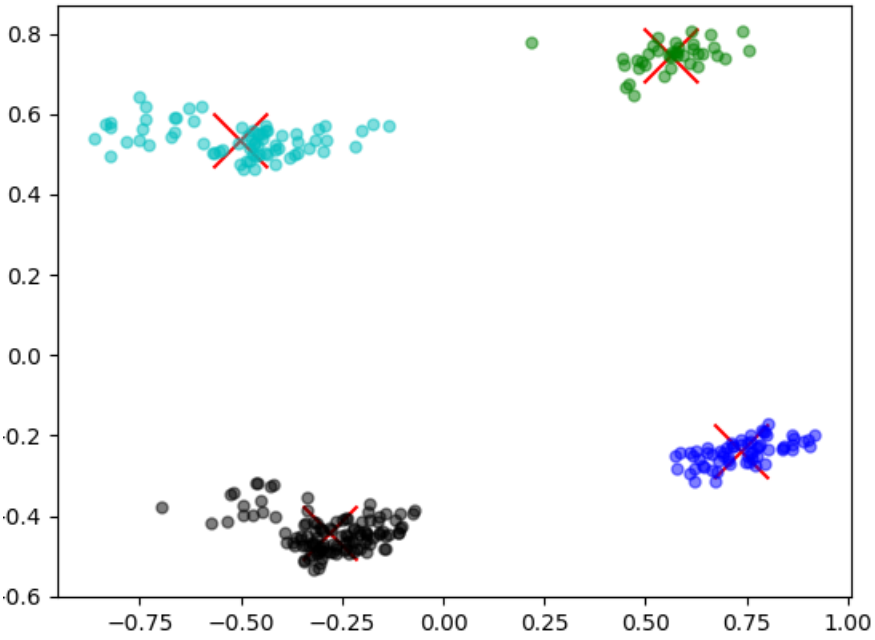
Clustering is an unsupervised learning technique deals with finding a structure in a collection of unlabeled data. We will process our new data set and find the groups that members are similar in some way. After that we will be able to see which data points are relevant to which groups and find out the labels of each data point.

## K-means

K-means is a clustering algorithm which tries to separate data into clusters with K groups of approximately equal variance by minimizing the error which is defined as sum of squares of distance between data points and mean of cluster centroids.

There are N data points, these data points have X features and we try to find k clusters. Algorithm starts with initial set of k means centroids. Then it continues iterating 2 steps. In assignment step for every data point Euclidean distance is calculated for k centroids. After that each data point is assigned to the nearest cluster. In update step regarding to new clustered data points new centroid points are calculated for each cluster. Then it iterates this 2 steps until data points are converged and assignments no longer change. An advantage is easy to implement. Disadvantages are cluster count should be given to algorithm and cluster separators are linear in space.

Here are the centroids and clustered data points:




We can observe again that these for clusters are mostly related to “target” and “zone” features because of the eigenvectors of the first 2 principal components.

## DBSCAN (Density Based Spatial Clustering Applications with Noise)

DBSCAN is another clustering algorithm given a set of points in some space; it groups together points that are closely packed together in high-density regions. It finds the outlier data points that lay alone in low-density regions whose nearest neighbors are too far away.

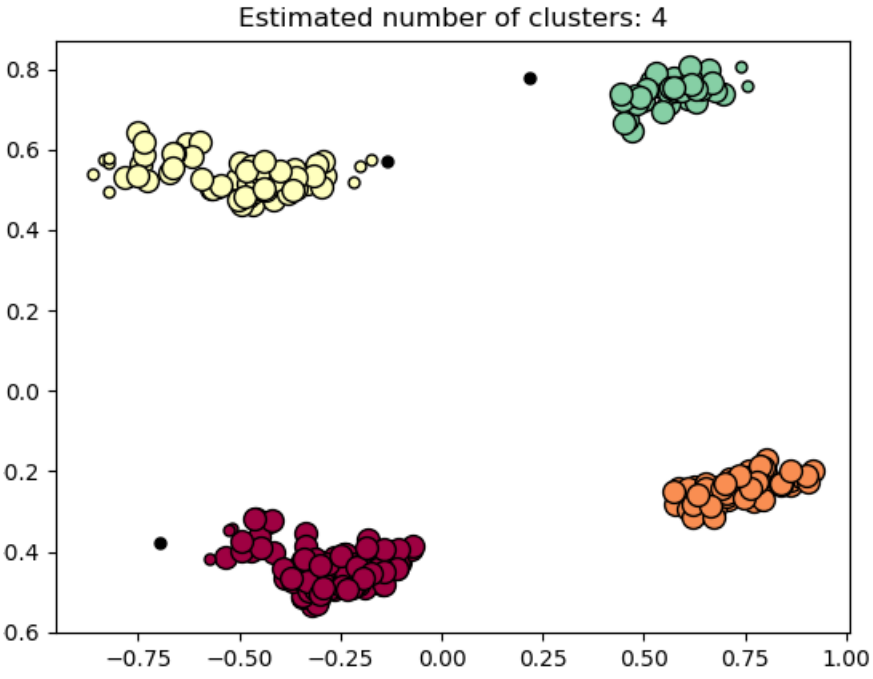
Consider a set of points to be clustered. DBSCAN clusters data points as core points, non-core points and outliers. The algorithm takes to given parameters are epsilon the distance of neighborhood and minimum point count in given radius of epsilon.

A point p is core point if at least given minimum points is within given distance $\varepsilon$ is the maximum radius of the neighborhood from p, including p). These points are directly reachable from p.

Another point q is directly reachable from a core point p within distance $\varepsilon$ is a non-core point but cannot reach points that given min point parameter within the cluster.

All other points which are not reachable from any other point are outliers.

As we see graph below there are obviously 4 clusters that found with DBSCAN algorithm. As parameters epsilon is chosen as 0.8 and minimum point in a cluster is chosen as 15.




If we carefully examine the graph we can see points in different size. Bigger colored points are the core points which can reach 15 data points in the given epsilon=0.8 boundary. Smaller colored points are the non-core points which can reach a core point in the given epsilon=0.8 boundary but cannot reach 15 points. Black small points are the outliers that cannot reach any core point in the given epsilon=0.8 boundary. It is not influential to reach a non-core point by any point. As these black points are examined

it can be found that reward is too high for the black point that left of the green group and the reward is too low for black point that left of the red group.

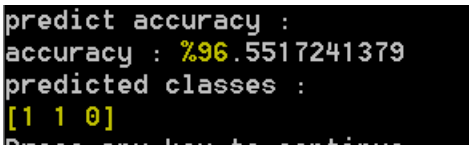
## K-Nearest Neighbors

In machine learning k-nearest neighbors algorithm is used for classification and regression. We will use this algorithm for classification. Our training examples are vectors in a multidimensional feature space, each with a class label gathered from clustering. We get the label information for each data point from DBSCAN clustering algorithm. When we get a new data sample input then we can calculate first k nearest neighbors. We will decide which class to assign by majority of class information among k nearest neighbors. When knn is applied the output is a class membership. K is the parameter that must be selected before applying the algorithm. It is selected for how many nearest neighbors will be looked up to find the most relevant class to assign to new data point. Selecting k is important concern. It depends on the data. In general, larger k values reduce the effect of noise in classification but it makes the boundaries of the classes less distinct. We select k=5 for less time and memory consuming.

The training phase of the algorithm consists only of storing the feature vectors and class labels of the training samples. We apart our data into 2 different sets. One of them is training input and output data, the other one is test input and output data. Firstly training system is used to teach the system what label comes for which specific inputs.

The test phase of the algorithm consists giving test input to the system and let it to predict the label of output for each input data point. After that it compares the given test output labels with predicted labels and calculate an accuracy level. If it is good enough regarding to selected threshold ratio it means that predicting works well. If it is lower than the selected threshold value it means that needs more training operation.

For example for our system we get the class labels from the DBSCAN clustering operation and applied the knn algorithm with k=5 and threshold accuracy ratio is %90. For cross validation we apart our data into train and test data with %90 for training, %10 for testing with 286 samples. After training and test phases, we can give new data points in the space and get the predicted labels from knn algorithm. %96.5 accuracy ratio might be enough to trust our system. Here are the results:




## Summary and Conclusion

Machine learning algorithms are applied to learn the context and learn the similarities and differences between data samples. After it learnt the model of the system from the training operation then it can predict the label of the new data using comparing features between new data and trained data.

Firstly data feature values should be converted from alphanumeric to numeric values. Because all of the machine learning steps need to use numeric values in order to calculate operations. Then a normalizing

operation should be done in order to create common comparable values. Normalization of ratings means adjusting values measured on different scales to a notionally common scale.

Applying machine learning algorithms to data directly might be a wrong decision because we have lots of features that some of them may be correlated to each other and many of them may not be necessary to use. Then dropping some columns and applying dimensionality reduction algorithms should be applied.

Now we can analyze and visualize our data more accurately using 2d or 3d because we have got rid of the unnecessary and correlated features. We have reduced the dimension count from 12 to 6. This new features give information about the model better in an easy way.

In this step clustering algorithms like DBSCAN or k means should be use in order to create separated data points as clusters. These clustering algorithms calculate density and distance to find a structure in a collection of unlabeled data. Then those algorithms process our data set and find the groups that members are similar in some way.

At last we could get the labels from the clustering in order to train and learn the system to create a model. When we deal with a new data point we can apply our knn algorithm to predict to find the class of new data. It means that when we get a new played game we can predict which class this game in.