Pokemon Clustering

CENG574 Final Report

Berke Ateş Aytekin

Middle East Technical University, 2309706

Uygar Yaşar

Middle East Technical University, 2310613

Clustering is an unsupervised learning method that is frequently used to segment the sample group under consideration. Within the scope of this research, it is tried to cluster the characters in Pokemon, a Nintendo game, according to their scores in the game. For this, the dataset was first projected in two dimensions using different methods, and then clustered with some clustering algorithms. These clustering results were compared with different clustering validation metrics and the most appropriate cluster structure was selected for the dataset. As a result of this research, there are 3 different clusters in the Pokemon dataset that can be labeled as “Weak Pokemons”, “Medium Strength Pokemons”, and “Strong Pokemons”.

Additional Keywords and Phrases: Unsupervised Learning, Data Clustering, Data Projection, Cluster Validation, Segmentation

ACM Reference Format:

B. A. Aytekin, U. Yaşar, 2022. Pokemon Clustering. CENG574 Final Report.

1 INTRODUCTION

It is difficult to visualize datasets with multidimensional feature vectors. Since we can easily understand two-dimensional representations with the human eye, we want to obtain two-dimensional projections of the samples when visualizing datasets. While we reduce the examples that we can represent with a large number of features to two dimensions, we use different linear or non-linear combinations of different features. However, no matter how detailed we try to obtain a combination, we experience some information loss when we try to reduce a large number of independent features to two dimensions. Projections in which we experience a large loss of information show a farther distribution from reality than those with a small loss of information. Thanks to these projection methods, we can witness certain groupings in the dataset before we go to the clustering stage.

Clustering is an unsupervised type of learning, that is, the labels of the data are not included in the dataset. The aim of the researcher is to cluster the data in such a way that instances in the same cluster are more similar than instances in other clusters. In this way, various inferences can be made, such as how many different sample types are in a dataset. Clustering applications have many different uses in real life. Market and customer segmentation comes first among these applications. Thanks to these applications, companies express their customers with certain clusters and thus, follow similar strategies against their customers in the same cluster. Within the scope of this research, we will perform a Pokemon segmentation by looking at the scores of the characters in the Pokemon world, which later extends to comics and anime, even though it is released as a Nintendo game. We will examine whether the values assigned to the features while creating these characters follow a certain cluster pattern.

For this research, we used the dataset named *Pokemon with stats* on the Kaggle website [1]. The dataset stores 800 Pokemons, including their name, first and second type, and basic stats: HP, Attack, Defense, Special Attack, Special Defense, and Speed. These statistics are derived from values used in Pokemon games. This dataset consists of 13 columns, 9 of which are numerical (integer), 3 are string and 1 is boolean variable. You can get an idea about the feature vector by seeing the first 5 examples in the dataset in Figure 1.

#	Name	Type1	Type2	Tot	HP	At	Def	Sp.A	Sp.D	Sp	G.	Leg.
1	Bulbasaur	Grass	Poison	318	45	49	49	65	65	45	1	False
2	Ivysaur	Grass	Poison	405	60	62	63	80	80	60	1	False
3	Venusaur	Grass	Poison	525	80	82	83	100	100	80	1	False
3	VenusaurMega	Grass	Poison	625	80	100	123	122	120	80	1	False
4	Charmander	Fire		309	39	52	43	60	50	65	1	False
5	Charmeleon	Fire		405	58	64	58	80	65	80	1	False

Figure 1: Head of the dataset

The Type 1 and Type 2 features you see in Table 1 show the non-numeric type features of the Pokemon. These features are the choices made to increase the variety of the characters in the Nintendo game. When we turned these categorical variables into a numerical variable with one-hot encoding and used them in projection and clustering methods, we could not achieve any improvement in the clustering structure. As we will talk about in the next section, no type-dependent clustering difference was observed from the cluster structures in previous studies on this dataset. Therefore, we decided to exclude Type 1, Type 2 and another non-numeric variable, Legendary, as part of this research. We will use only the numerical features of the dataset throughout the research.

Before moving on to projection and clustering studies, you can see the correlation between features in Figure 2. There is a high correlation between ID and Generation features as the samples in the dataset are ordered by their Generation. Since the Total feature is the sum of the numerical features except Generation and ID, correlation with other numerical features was expected. The biggest correlation of Total feature with Special Attack and Special Defense features shows us that these two features are the biggest shareholders of the Total value.

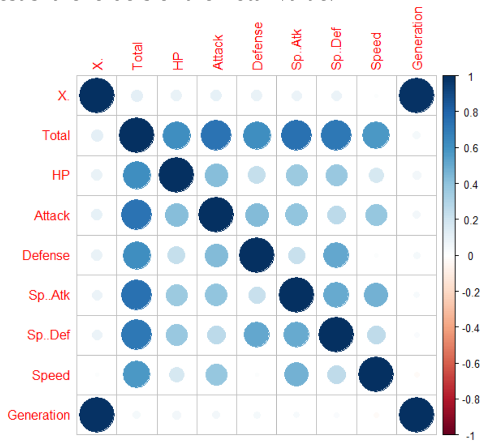


Figure 2: Correlation matrix

2 RELATED WORK

Although there has not been a detailed analysis and research on the *Pokemon with stats* dataset before, there are studies done by some users on the Kaggle platform where the dataset is located. Almost all of these studies are incomplete or very superficial studies. The only completed study aiming to cluster Pokemons among them performed this analysis using only k-means and stated that there are 5 clusters. In 2020, Bernardo named these clusters as follows: Weakest Pokemons Cluster, The Overpowering Squad, Speedy Squad, The Defensive Squad, and High HP and Slow Speed Squad [2]. Since he reached this conclusion directly without trying different cluster numbers and clustering methods, it was necessary to conduct a more comprehensive analysis on this subject.

3 PROJECTION OF DATA

In this part, we will calculate and visualize the projection of the Pokemon dataset in two dimensions. In this way, we will have an idea about how many clusters there can be in the dataset before clustering. We will find these two-dimensional projections of the dataset by using Principal Component Analysis, Principal Coordinate Analysis with two different metrics, and finally t-SNE methods.

3.1 Principal Component Analysis (PCA)

PCA is the most classical method used to reduce the dimensionality. It uses the covariance matrix of the dataset and creates its first principal component in the direction with the highest variation. It then creates its own orthonormal basis by selecting the other principal components perpendicular to the first component and visualizes the samples on these new principal coordinates.

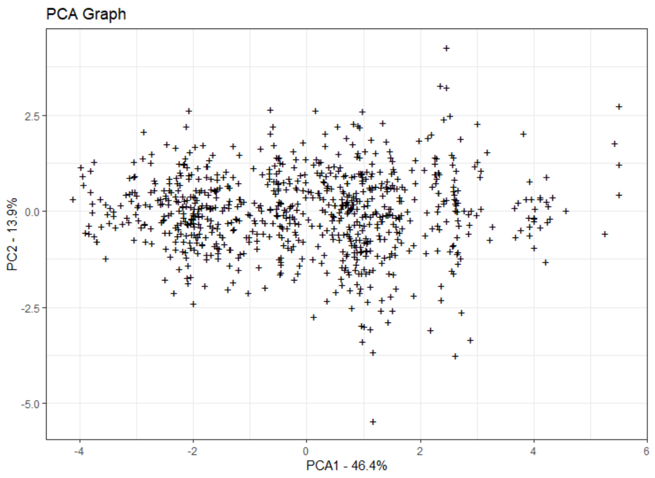


Figure 3: PCA graph

In Figure 3, you can see the projection of the Pokemon dataset in two dimensions obtained using PCA. When you look at the graph, you can see that there are too many outliers on the right, but it is possible to distinguish different clusters from the sample densities in the middle. When we look at the total variation in the first two principal components, we have a value of 60.3%. It would take a 90% variation to have an almost valid idea of the dataset. Therefore, the clusters in the dataset may be quite different from those in Figure 3.

3.2 Principal Coordinate Analysis (PCoA) with Manhattan Distance

Classical MDS, or Principal Coordinate Analysis, is actually a very similar process to PCA in terms of its underlying mathematics. Actually the only difference is that PCA uses a covariance matrix while PCoA uses a distance matrix. Because different distance measurement metrics can be used when calculating the distance between samples, PCoA may have different representations according to different distance metrics. Since using Euclidian distance will give exactly the same result as PCA, we first decided to use Manhattan distance.

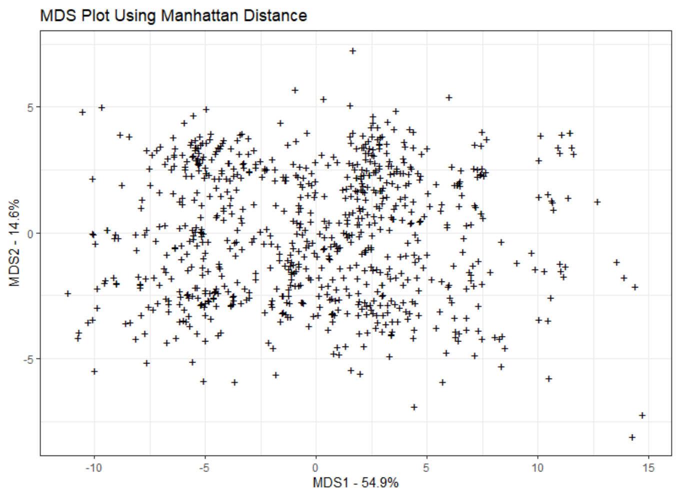


Figure 4: MDS plot using Manhattan distance

In Figure 4, you can see the results of the PCoA we applied using Manhattan distance. As you will notice, we obtained different result from the previous projection, and an easily distinguishable cluster structure is no longer observable. However, the total variation we get in the two dimensions is now 69.5%, which is higher than what we got in PCA. Therefore, we think that this graph is a more accurate representation as a two-dimensional projection of the dataset.

3.3 Principal Coordinate Analysis with Avg(logFC) Distance

When we looked at different studies previously performed on different datasets, we saw a PCoA study performed with a distance metric called avg(logFC). This new distance metric provided a noticeable increase in the total variation that could be displayed in two dimensions compared to Manhattan distance [3]. Therefore, we decided to create a new distance matrix, and first calculated the base-2 logarithm of all values in the dataset. We then averaged the absolute differences between all their features to find the distance between the two samples. Since there is no function that prepares a matrix with this method, we performed this calculation ourselves, and then we performed multidimensional scaling using this distance matrix.

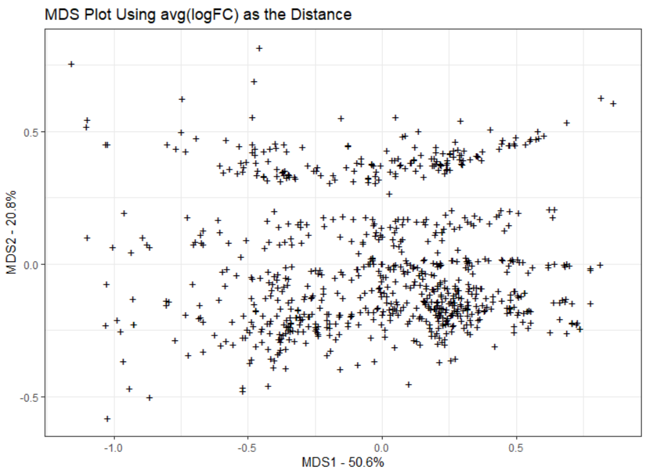


Figure 5: MDS plot using avg(logFC) distance method

In Figure 5, you can see the result of PCoA we performed with the distance metric avg(logFC). This projection in two dimensions shows certain similarities with the previous one with Manhattan distance. Clusters are a little easier to distinguish in this new projection because there are gaps between sample densities. The total variation obtained in the two dimensions is 71.4% which is larger than all other previous projection variations.

3.4 t-SNE

t-SNE is a non-linear projection method. It is used in data sets that are not suitable for dimension reduction with linear methods such as PCA or MDS. This technique, which was emerged with Maaten and Hinton's "Visualizing Data using t-SNE" article published in 2008 [4], basically tries to keep the disjoint probability distribution in the data space and the map space as close as possible. To calculate this closeness, it tries to minimize the metric called Kullback-Leibler Divergence, which is related to the locations of the points in the map. While doing this, unlike the previous SNE method, it uses the t-Student Distribution in the map space. Moreover, when visualizing in two dimensions, it offers a solution to the tendency of the samples to crowd in the center of the map, also known as the crowding problem.

The biggest advantage of t-SNE is that it is a non-linear projection method. Our projection with PCA showed a 60.3% variance, which we thought was not a good number. Therefore, we considered the possibility that our dataset might be too complex to be displayed with linear projections. Moreover, t-SNE better preserves the local and global structure, unlike PCA. On the other hand, the biggest disadvantage of the t-SNE algorithm is that it takes the probability calculation while calculating the similarities between the samples, does a lot of operations during the iterations and keeps two matrices including the distances. We can summarize the time complexity as $O(KN^2M)$, where N is the number of instances, M is the number of features, and k is the number of iterations. Space complexity is also $O(N^2)$ since it holds similarity matrices. Besides, another big disadvantage is that it contains a hyperparameter called the perplexity, unlike the projection methods we used before. To tune this hyperparameter and find the optimal value, it is a great challenge for users to run this algorithm, which is already high in complexity, over and over again. In addition to these, we can add as a disadvantage that t-SNE is not deterministic and it produces different results in different runs due to the random placement of the instances in the first iteration.

The learned parameters of the t-SNE method are the values in the created similarity matrices. These values are assigned and changed while the algorithm is running and cannot be affected by the user. One of the hyperparameters that the user can change while calling the function is the maximum iteration number, but since we will cut it at the point of convergence, we can easily handle it without the need to include it in the tuning process. The hyperparameter of the t-SNE algorithm that needs to be tuned is perplexity. This value is the number of samples expected to be near a sample, in short, the expected density. Maaten and Hinton stated in 2008 that this hyperparameter should generally range between 5 and 50 [4]. For this reason, while searching for the appropriate value, we thought to use the values of 5, 25 and 50, respectively, to see the two extreme cases and the situation in between, and then to have an idea about choosing a suitable perplexity by looking at these results. While doing this, we used the t-SNE function in the t-SNE package. We visualized the projections obtained for these perplexity values in Figure 6.

Note that since we only have 800 samples, we did not see the need to select a small subset from the dataset and try different hyperparameters on it. We directly included all the data in the entire hyperparameter optimization process.

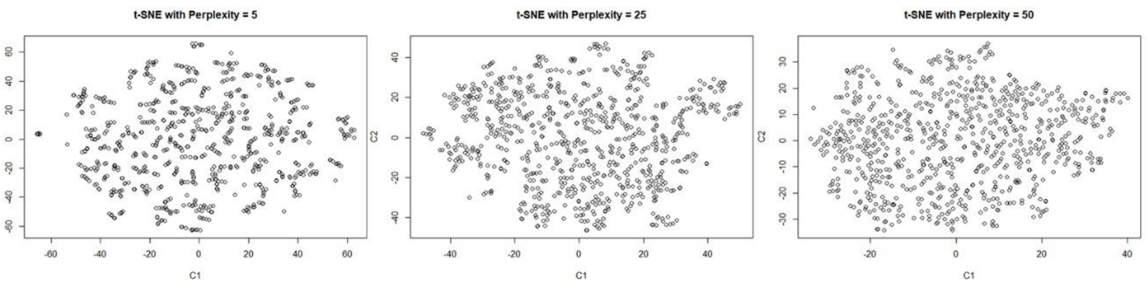


Figure 6: t-SNE with different perplexities

As seen in other projection methods, there are many elements in the dataset that we can call outlier, which is too far apart from the density centers. Ruling out any of these items shrinks the dataset too much, so we chose not to remove these values. Since the dataset is not very dense, the expected density value around the samples is also small. For this reason, small perplexity values will work better with the Pokemon dataset. You can also understand this situation from the t-SNE graphs. Unfortunately, the dataset, which gives better results with small perplexity values, has obtained a structure containing too many small clusters. If we encountered such a structure as a result of a clustering method, it would be unacceptable. Therefore, we decided not to use the non-linear projection that we obtained with t-SNE as a basis for the rest of the research

4 DATA CLUSTERING

We tried to observe possible cluster structures in the dataset with projection methods. Now, we will visualize the clusters in the dataset using direct clustering methods. While doing this, we will apply hierarchical clustering, k-means clustering and density-based clustering, DBSCAN, respectively. We will try different hyperparameters in each method and try to find the optimal cluster structure.

4.1 Hierarchical Clustering

Hierarchical clustering starts by considering each sample in the set as a separate cluster and continues until a single cluster remains by combining clusters and/or instances closest to each other. For this reason, nested cluster structures are seen. We can stop this process when we get the number of clusters we want and observe the cluster structure in the dataset. Although there are not many different methods for measuring the distance between samples, different linkages can be mentioned when measuring the distance between clusters. This linkage is actually the hyperparameter of the hierarchical clustering. We tried different linkages throughout the research. In Figure 7, you can see the dendograms we obtained as a result of the single-linkage, complete-linkage and average-linkage applications, respectively.

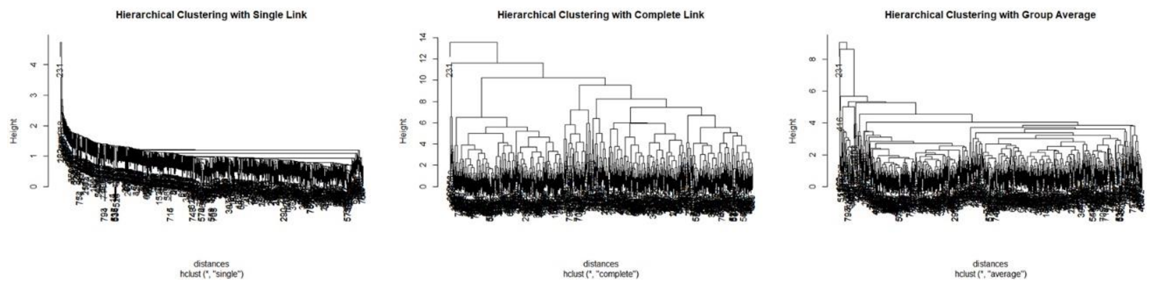


Figure 7: Hierarchical clustering dendograms with single, complete, and average links

As you can see, the dendograms above are very dense and contain clusters of single samples. When we used Ward's method as the linkage, we got a much more sparse and clearer dendogram than the previous ones. You can see the corresponding dendogram in Figure 8. Another hyperparameter of hierarchical clustering was the number of clusters, that is, where to cut the dendogram. While doing this, we returned to the projections in the previous part to make an inference about how many clusters there could be in the dataset. For this reason, we thought that there could be 3 or 5 clusters in the dataset.

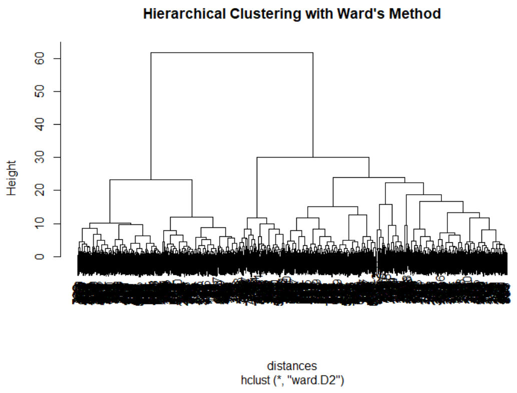


Figure 8: Hierarchical clustering dendogram with Ward’s method

When we performed hierarchical clustering with the number of 3 clusters with the Ward linkage, the clusters in the dataset were as in Figure 9.

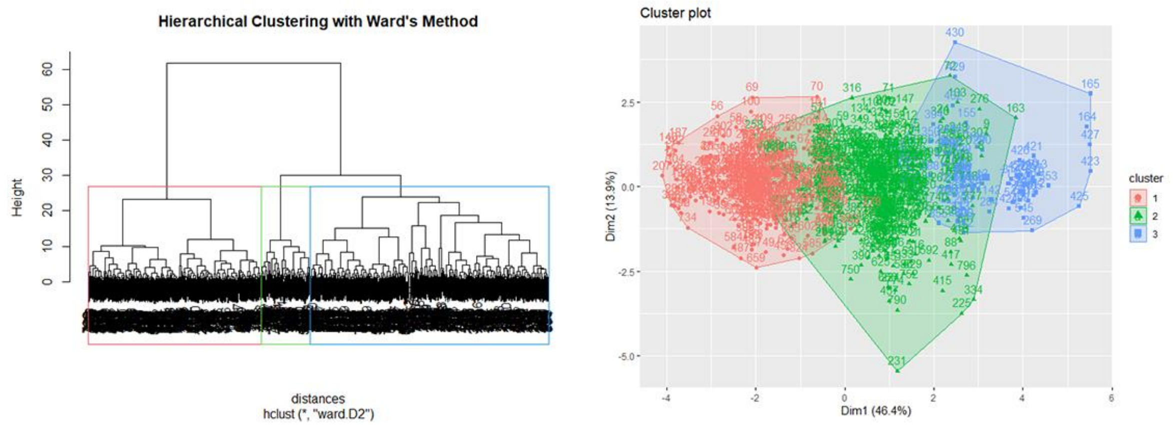


Figure 9: Hierarchical clustering with Ward’s method for 3 clusters

Assuming that there are 5 clusters in the dataset, the result of hierarchical clustering was as in Figure 10.

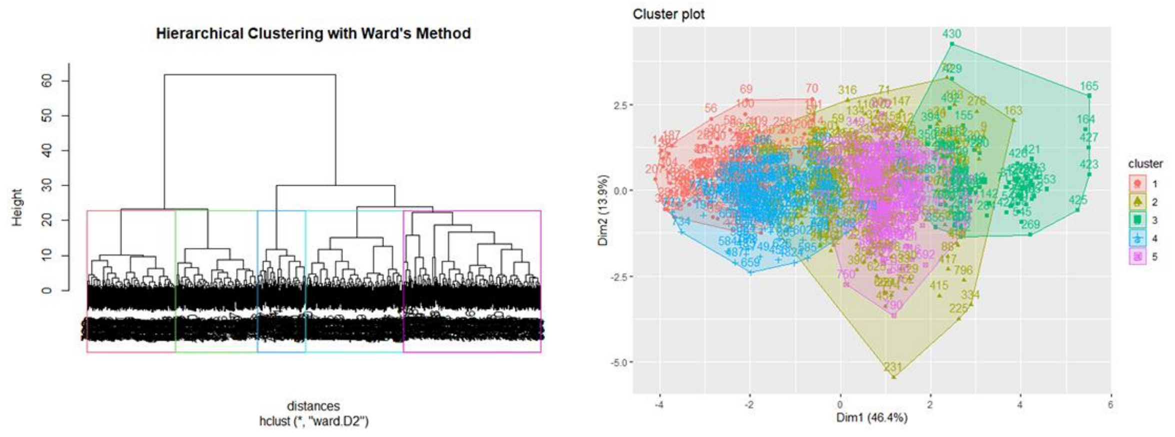


Figure 10: Hierarchical clustering with Ward’s method for 5 clusters

Although there are very large overlapping regions between clusters in both of these graphs, it should be noted that when we reduce an 8-dimensional dataset to two dimensions using PCA, we can only show a variation of 60.3%. These cluster structures are completely disjoint from each other in the original feature space.

4.2 K-Means Clustering

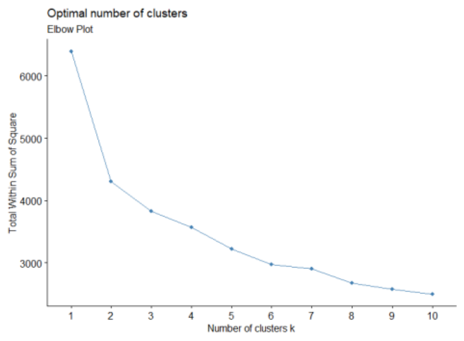


Figure 11: Elbow plot

The most important hyperparameter in the k-means clustering is the k value, that is, the number of clusters in the dataset. We can find the most appropriate value for k using the elbow method. For this, we plot different objective loss function values obtained according to different numbers of clusters. The first break point on this line graph, that is, the elbow-like point, gives the optimal number of clusters. Also, the k-means algorithm is not a deterministic algorithm because the result varies depending on the cluster centers initially assigned. There are different initializations for assigning these cluster centers, but we exclude this hyperparameter change from this paper as we observed in our study that these different initializations do not have a radical effect on the resulting clusters. When we examine the plot in Figure 11, we see that the first break occurs when the k value is 2, according to the elbow method. However, the large decrease in the loss function after k=2 caused us to take an initiative to perform clustering for different k values and see the result. Therefore, you can see the clustering structures obtained by applying k-means clustering for different values of k from 2 to 6 in Figure 12.

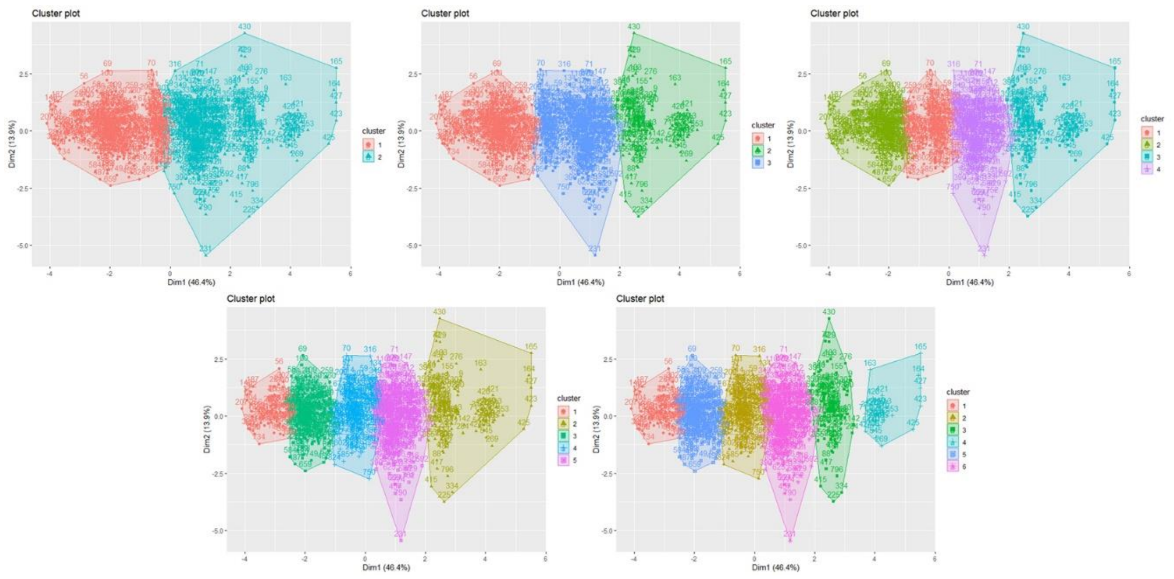


Figure 12: k-Means clustering with different k values

When we use the k-means algorithm, unlike hierarchical clustering, there was no overlapping region when visualizing in two dimensions, and so, each of the different clusterings we made from 2 clusters to 6 clusters produced visually satisfactory results.

4.3 DBSCAN

DBSCAN is a density-based clustering non-parametric algorithm. It groups close points as clusters, marking as outliers points that lie alone in low-density regions, that means, whose nearest neighbors are too far away. This algorithm emerged in Ester et al.'s 1996 article "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise" [5]. According to this algorithm, arbitrary points are selected in the dataset. Then, other points within the calculated diameter of these selected points using the hyperparameter Epsilon are determined, and if the number of these points is more than the other hyperparameter given as minPts, this point is assigned as the core point; if not, non-core point. Points that cannot be assigned as core or non-core are determined as outliers. Afterwards, any of the core points is selected and a cluster value is assigned, and the algorithm proceeds by assigning the same cluster to the core and non-core points around it within the diameter calculated by Epsilon. At the end of the algorithm, both clusters and outlier samples are observed in the dataset.

This method works better on datasets where the samples are dense. The Pokemon dataset is a sparse dataset spread over a very wide area, with density variations. In this respect, it does not seem appropriate for DBSCAN at first. However, we wanted to make a clustering using this algorithm and compare the result with our previous results, since we can distinguish the samples that may be outlier with this algorithm without outlier elimination as a pre-process. In other words, the fact that DBSCAN is outlier and noise resistant is one of the most important advantages of this method and the main reason why we want to examine this method. In addition, another advantage of DBSCAN is that we do not need to assign the cluster number as a hyperparameter beforehand. Without specifying any number of clusters, DBSCAN can show us the clusters in the dataset. Furthermore, DBSCAN can find arbitrarily-shaped clusters. It can even find a cluster completely surrounded by other cluster. When we look at the time and space complexity of the method, we can see that it produces the same or even better results than methods that do the same job. It visits each point multiple times, which is up to cluster count. Therefore, overall average runtime complexity of $O(N \log N)$; however, in the worst case, which is the case that all the points closer than Epsilon to the cluster center, the time complexity is $O(N^2)$. The space complexity is $O(N^2)$ due to the distance matrix.

When we come to the disadvantages of DBSCAN, first of all, the method is not deterministic since border points that are reachable from more than one cluster can be part of either cluster depending on the order the data are processed but this is not a frequent situation. Moreover, for high-dimensional data, this method can be useless due to the curse of dimensionality; however, since in the Pokemon dataset, the number of samples is far larger than the number of features, we cannot observe the curse of dimensionality. Lastly, DBSCAN cannot cluster data sets well with large differences in densities since choosing optimal minPts and Epsilon is impossible when the densities varies a lot between clusters. This was the main problem in the Pokemon case because possible clusters would have relatively larger density differences.

The learned parameters of DBSCAN are the distances between the samples and whether the points are core points or not. The most important hyperparameters to be given by the user are the Epsilon and the minPts value. We can use the knee method similar to the elbow method we use in k-means clustering to find the optimal Epsilon value, but we will need to perform clustering with different values to find the optimal value of the minPts hyperparameter. As mentioned before, since the dataset is not very large, a method like hold-out or cross validation will not be used and the entire dataset will be included in this hyperparameter optimization process. We will use the dbscan and factoextra packages while performing these operations.

In order to find the optimal Epsilon value using the knee method, it is necessary to draw a different graph for each minPts value. During the hyperparameter optimization process, it was observed that the breaking point occurred at Epsilon=55 in all of the different graphs created for different minPts values wanted to try. Although we do not visualize each graph separately, you can see the graph obtained when the minPts value is 2 in the Figure 13.

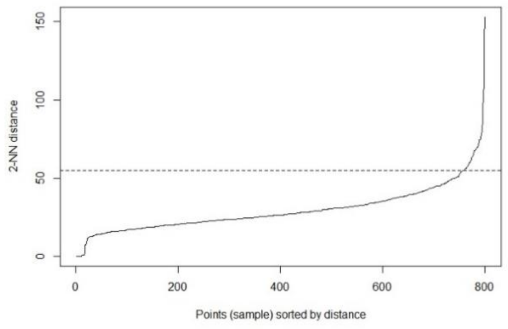


Figure 13: Knee graph

By keeping the Epsilon value constant at 55, we performed clustering operations for different minPts values. None of these clusters produced acceptable results. This was due to the density differences in the dataset, as stated above. You can see the clustering results we obtained with minPts value of 2, 3 and 4 in Figure 14.

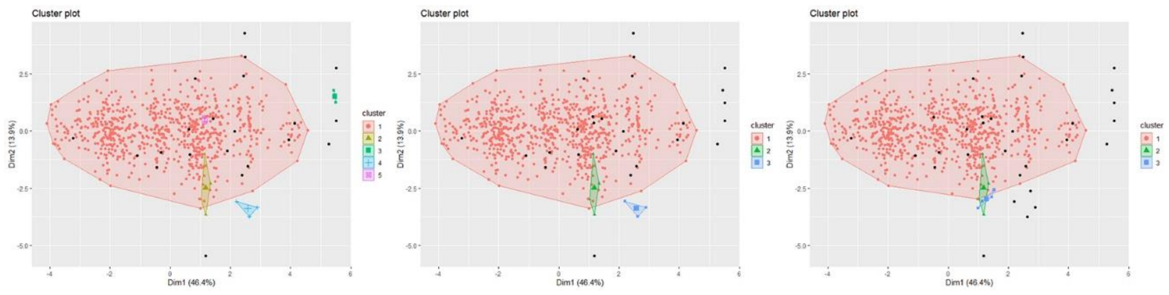


Figure 14: Clustering with DBSCAN algorithm

Only hierarchical and k-means clustering will be taken into account in the clustering validation part, since no result worth evaluating with DBSCAN has been achieved.

5 CLUSTER VALIDATION

So far, clustering between 3 and 6 clusters using hierarchical and k-means clustering has yielded satisfactory results. Therefore, in this part, this method and cluster number combinations will be evaluated by looking at different clustering validation metrics and the clustering that gives the best results for the dataset will be selected as the final clustering structure. Different validation metrics' values according to the different clustering combinations are given in Figure 15.

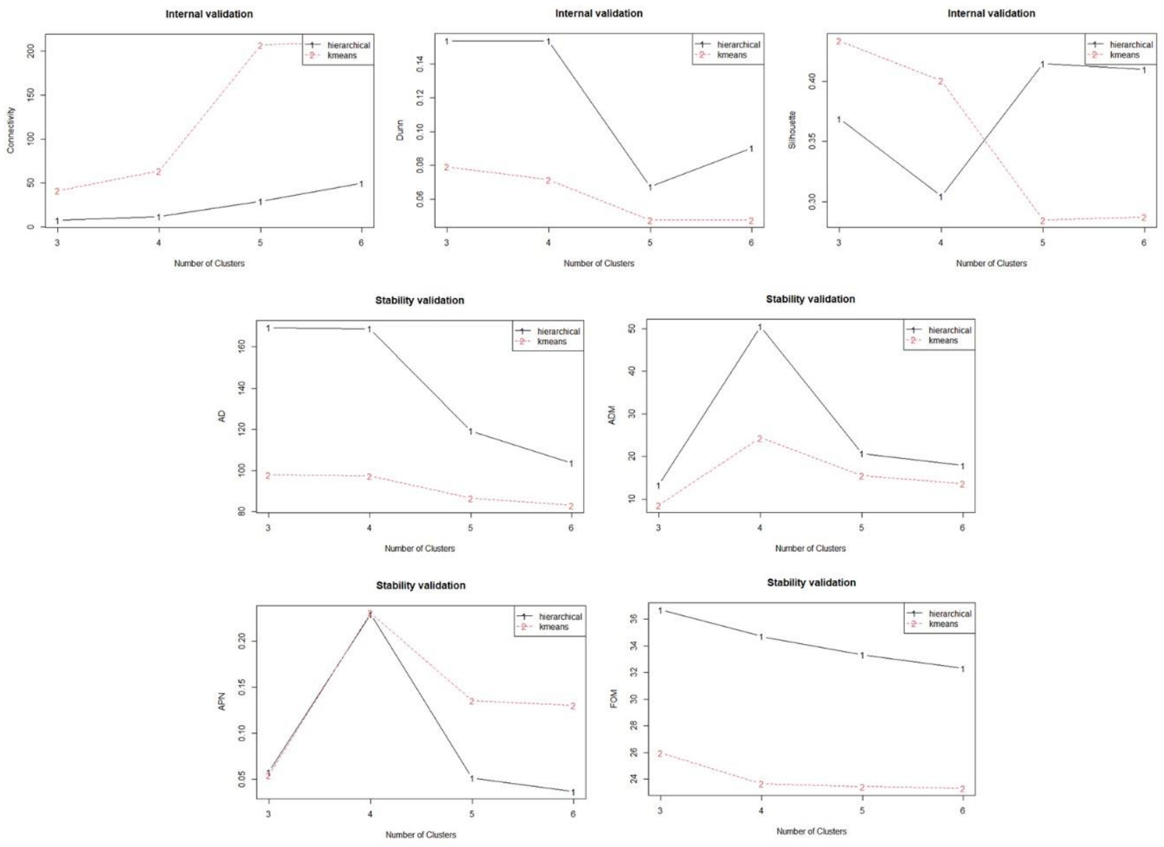


Figure 15: Internal and stability cluster validation with different algorithms

As you can see in the Figure 15, 3-cluster hierarchical clustering and 3 and 6-cluster k-means clustering come to the fore among the most optimal combinations. After making a comparison between the scores obtained in the metrics where these combinations are not the most optimal, it is decided that the best choice for clustering the *Pokemon with stats* dataset is the k-means clustering method with k=3.

6 RESULTS

In the Cluster Validation section, it was observed that the most suitable structure was obtained with 3-means clustering as a result of the different clusterings applied so far. The clustering structure obtained as a result of the whole study is as in Figure 16.

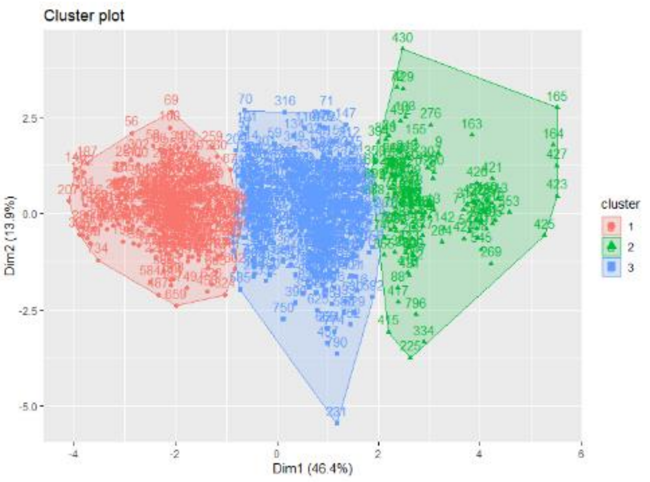
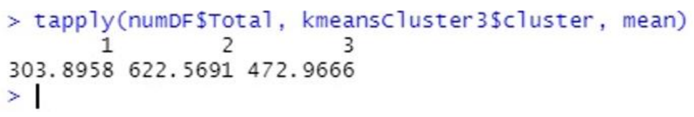


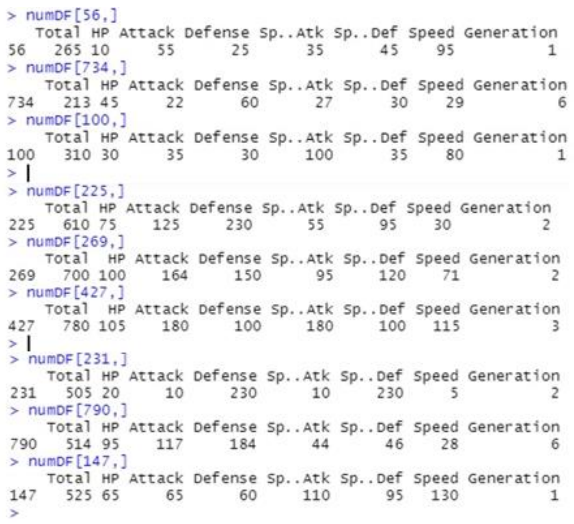
Figure 16: Final clustering

Since the Total feature is the sum of other numerical features, it will play an important role in understanding what the 3 clusters mean on the dataset, as it has the most comprehensive information. When we take the mean of the Total feature in 3 clusters, we get a situation like Figure 17. As can be clearly seen from this table, there are big differences between the Total features of the three clusters. Therefore, we can say that these three clusters consist of “Weak”, “Medium Strength” and “Strong” Pokemons, respectively. To strengthen this assumption, several randomly selected samples from different clusters are visualized in Figure 18.



```
> tapply(numDF$Total, kmeansCluster3$cluster, mean)
       1        2        3 
303.8958 622.5691 472.9666 
> |
```

Figure 17: Mean of Total feature in 3 clusters



```
> numDF[56,]
   Total HP Attack Defense Sp..Atk Sp..Def Speed Generation
56   265 10     55      25      35      45    95          1
> numDF[734,]
    Total HP Attack Defense Sp..Atk Sp..Def Speed Generation
734   213 45     22      60      27      30    29          6
> numDF[100,]
    Total HP Attack Defense Sp..Atk Sp..Def Speed Generation
100   310 30     35      30     100      35    80          1
> |
> numDF[225,]
    Total HP Attack Defense Sp..Atk Sp..Def Speed Generation
225   610 75    125     230      55      95    30          2
> numDF[269,]
    Total  HP Attack Defense Sp..Atk Sp..Def Speed Generation
269   700 100    164     150      95     120    71          2
> numDF[427,]
    Total  HP Attack Defense Sp..Atk Sp..Def Speed Generation
427   780 105    180     100     180     100   115          3
> |
> numDF[231,]
    Total HP Attack Defense Sp..Atk Sp..Def Speed Generation
231   505 20     10     230      10     230     5          2
> numDF[790,]
    Total HP Attack Defense Sp..Atk Sp..Def Speed Generation
790   514 95    117     184      44      46    28          6
> numDF[147,]
    Total HP Attack Defense Sp..Atk Sp..Def Speed Generation
147   525 65     65      60     110      95   130          1
> 
```

Figure 18: Random instances in the clusters showing the assumption

7 CONCLUSIONS

Within the scope of this study, Pokemon segmentation was made using a dataset of Pokemon, which is a Nintendo game and has characters with different scores in different fields. Firstly, the dataset was visualized with various projection methods, so that when clustering methods are used, an idea of how many clusters can be obtained is obtained in advance. Having this information in advance has also provided great convenience in hyperparameter selections. Afterwards, clusters were made with different clustering methods, and after these clusterings were examined with different clustering validation metrics, the best clustering method and number of clusters were selected.

As a result of this research, 3-means clustering was chosen as the most suitable algorithm that can be used to cluster the Pokemon dataset. When we look at these clusters, it can be said that they are formed under the influence of the Total power of the Pokemon. In fact, these clusters can be labeled as Weak Pokemons, Medium Strength Pokemons, and Strong Pokemons, respectively.

REFERENCES

[1] Alberto Barradas. 2017. Kaggle Dataset. Retrieved January 15, 2022 from https://www.kaggle.com/abcsds/pokemon.

[2] Bernardo, J. (2020). Pokmeans - KMeans clustering in Pokedex. Retrieved from https://www.kaggle.com/jbernardo/pokmeans-kmeans-clustering-in-pokedex.

[3] StatQuest with Jost Starmer. (2017,12,19). StatQuest: MDS and PCoA in R[Video].Youtube. URL https://www.youtube.com/watch?v=pGAUHhLYp5Q.

[4] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing Data using t-SNE. Journal of Machine Learning Research 9, 86 (2008), 2579–2605. Retrieved from http://jmlr.org/papers/v9/vandermaaten08a.html

[5] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD'96). AAAI Press, 226–231.