Data Mining and Knowledge Discovery

Analysis of WNBA Player Similarity through Clustering

Chelsy Mena

Professor Fabrice Muhlenbach

1. Introduction

The Women's National Basketball association, or as it is commonly known, WNBA is the biggest women's professional basketball league. It has been active since 1997 and is currently comprised of 13 teams, located in the United States and planning expansion as its popularity grows [6]. Much like it's male counterpart, it is the best league in the world for the sport, and there is interest of using data mining and machine learning techniques to extract insight out of the myriad of statistics carefully gathered in every game.

In basketball players have a designated position, this informs their actions in the court, and so there is a set of expectations that come with each given position. The objective of this project is to test how closely the players in the WNBA match these expectations and if it is possible to surmise the position a player is playing by their performance on the score sheet.

We will consider the classical positions:

- **F (Forward)**: A versatile player responsible for scoring, rebounding, and defending, often playing on the wings or in the post.
- **G (Guard)**: The primary ball-handler and playmaker, responsible for setting up plays and scoring from the perimeter.
- **F-C (Forward-Center)**: A hybrid role combining the scoring and agility of a forward with the size and rim protection of a center.
- **C (Center)**: The tallest player, focused on rebounding, shot-blocking, and scoring near the basket.
- **F-G (Forward-Guard)**: A wing player with playmaking skills, capable of handling the ball and creating shots while also playing as a forward.

Since there is an apparent subdivision of tasks in the court, the statistics should reflect it. Guards should score more three-pointers and have more assists than centers, centers should have high rebounds in comparison to the rest of the positions, and forwards should have more steals and turnovers because of their position in the court spacing. We will apply several clustering algorithms on our original dataset and on one processed with dimensionality reduction techniques to see if these assumptions hold.

2. Data Description & Preprocessing

2.1. Data Description

Professional sports leagues have the most advanced techniques for real-time stat collecting there is. The WNBA makes the score-sheet available as each game progresses and after it finishes and there are plenty sites online who centralize and present this information for sports journalists and fans.

This data includes things like points scored, shots attempted, rebounds, among others; and is then totaled over the season and presented in common aggregations like per minute played, per 100 possessions, per 36 minutes, etc.

For this project, we gathered the "per 100 possessions" statistics of all the players for all the seasons (1997-2024). A possession is basically a turn the offensive team has to score before they either do or lose the ball and have to go on defense. The common aggregation done in the data used in this project scales the stats of a given player per 100 team possessions while they are on the court.

This data is publicly available in the Basketball Reference website [2], and was scraped using a Python script available in the code repository for this project.

The resulting table had information of 1095 unique players, in 4935 entries of 27 columns, detailed in table 1

2.2. Data Preprocessing

Initially, the data had 7 different labels: 'F', 'G', 'G-F', 'F-C', 'C', 'F-G', and 'C-F'. As we can see, there is some overlap in this labeling and to simplify the exercise, we merged the ones that were the same position in an opposite ordering. This made our final labeled positions 'F', 'G', 'F-C', 'C', and 'F-G'.

There were 8 rows where the data was mostly missing, simply players that did not play more than 1 game that year, possibly due to injury or other reasons; these were removed. We found the rest of the missing values in the percentage columns when the player did not attempt the relevant shot type in that year (e.g., a player that did not shoot any free throws had the FT% value set as a missing value), these were imputed with 0.

Table 1. Basketball Player Statistics

Stat	Description
Player	Player Name
Team	Team
Pos	Position
G	Games
MP	Minutes Played
GS	Games Started
FG	Field Goals (2-point + 3-point)
FGA	Field Goal Attempts (2-point + 3-point)
FG%	Field Goal Percentage (FG / FGA)
3P	3-Point Field Goals
3P%	3-Point Field Goal Percentage (3P / 3PA)
3PA	3-Point Field Goal Attempts
2P	2-Point Field Goals
2PA	2-Point Field Goal Attempts
2P%	2-Point Field Goal Percentage (2P / 2PA)
FT	Free Throws
FT%	Free Throw Percentage (FT / FTA)
FTA	Free Throw Attempts
ORB	Offensive Rebounds
TRB	Total Rebounds
AST	Assists
STL	Steals
BLK	Blocks
TOV	Turnovers
PF	Personal Fouls
PTS	Points
Season	Season Start Year

Finally, we were left with 4927 unique entries in our table.

2.3. Feature Selection

An analysis of the correlation matrix yielded results aligned with expectations. The field goals and each type of shot attempted and made is correlated with its made percentage. And the number of games played is correlated with the games started and minutes played, and the total and offensive rebounds as well. It's interesting to note there is a slight negative correlation between rebounds and 3-point shooting, a good clue that we can find some pointers of position by the strong stats per player.

For this project it was important to differentiate between the different types of shooting, 2-point, 3-point and free throws, and we also kept the offensive rebounds and obtained the defensive rebounds by subtracting them from the total, as well as the other defensive stats; and the total points. It is important to remember that these stats are already somewhat scaled and normalized, since they are the stats per 100 possessions.

The final features selected were:

- 3PA (3-Point Attempts)
- 3P% (3-Point Percentage)
- 2PA (2-Point Attempts)
- 2P% (2-Point Percentage)
- FTA (Free Throw Attempts)
- FT% (Free Throw Percentage)
- PTS (Points)
- ORB (Offensive Rebounds)
- DRB (Defensive Rebounds)
- AST (Assists)
- STL (Steals)
- BLK (Blocks)
- TOV (Turnovers)
- PF (Personal Fouls)

3. Clustering Methods & Implementation

3.1. Dimensionality Reduction

In order to use dimensionality reduction techniques, the first step to take is to remove the outliers from the dataset. In this case the method used to determine which data points are outliers was restricting the entries to those inside the range defined by 1.5 times below and above the interquartile range. After this step 3457 data points remained, a 30% reduction in size.

3.1.1. Principal Component Analysis (PCA)

We performed a principal component analysis on this reduced dataset, and found that the variance preserved with the first 3 components was only a bit less than 55%, not ideal. This may be due to the fact that we had already removed the highly correlated features in the dataset. A scree plot of this process can be seen in figure 1.

Even though the first two components only explained 44& of the varince, when plotting these we could see some semblance of zones for the position labels in the dataset. Guards (G) were to the left, while forwards (F) and centers (C) were mostly right and mixed.

3.1.2. t-Distributed Stochastic Neighbor Embedding (t-SNE)

Following PCA, t-SNE was performed. In order to visualize the results, the dimensions were set at 3.

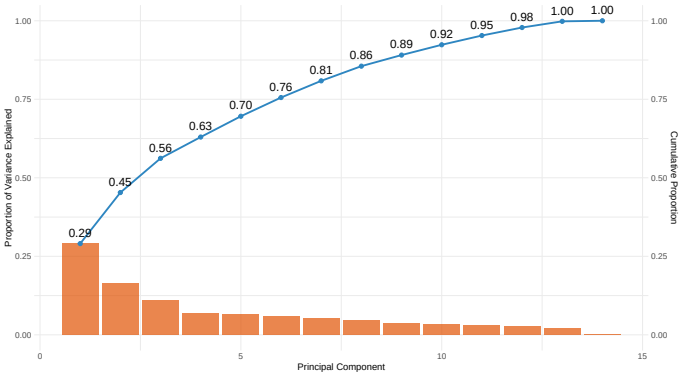

Figure 1. Scree plot for the PCA.

Chaining these two methods of dimensionality reduction allowed us to see a clearer separation of the positions. The guards occupy a clear region, and the forwards another, with the forward-guards in the middle of them. The centers and center-forwards should be the hardest to classify since they are heavily mixed with the forwards but at disadvantage in dataset balance. These regions can be seen in figure 2. This same scatter plot can be interacted with in the R Notebook uploaded to the project repository, turning this data cloud makes it even more evident that there is some kind of separation between positions.

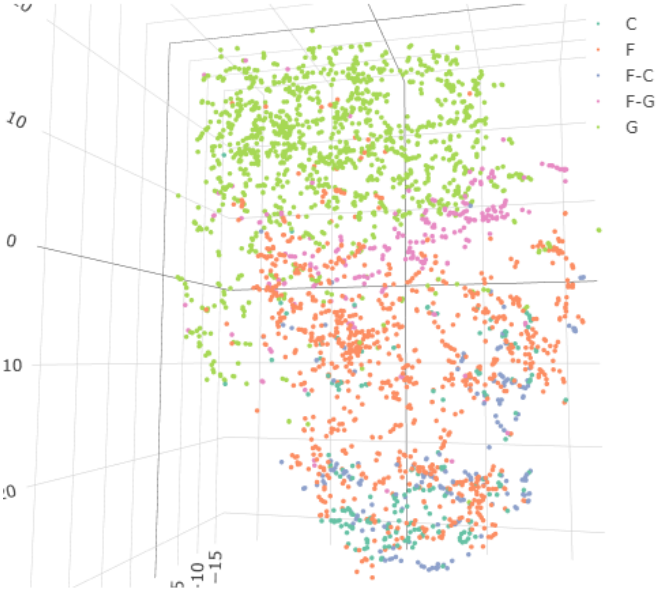

Figure 2. Point cloud resulting from the t-SNE clustering.

3.2. Clustering

We attempted to retrieve the positions from the stats, to see if there is such a thing as a style of play corresponding to the labeled positions. We did so using four different algorithms for clustering and using the original dataset and the one reduced and transformed through t-SNE.

3.2.1. k-Means

k-Means clustering is an algorithm that seeks to create **k** clusters from a data cloud, while minimizing the **within-group sum of squares (WSS)**, which measures how compact each cluster is, or maximizing the inter-cluster distance, which measures how far away a cluster is from the others [3].

This algorithm creates the clusters starting with an initial set of random centroids and going through the assignment of points to each one of them, followed by the calculation of a new centroid as the mean of the newly assigned points, and iterating until it reaches a stable configuration.

We used a 5-fold cross-validation procedure to find the optimal number of clusters (k) for our datasets using the **elbow method**, in which we increase k seeking the point after which the trend of metric improvement changes drastically, thus indicating that the clusters were better formed than the number of clusters before.

The results of this process are shown in figure 3, we can see how there was not a clear number of clusters for which the model improved dramatically, in either dataset. So in this case, we used our previous knowledge of the data and set the number of clusters to 5.

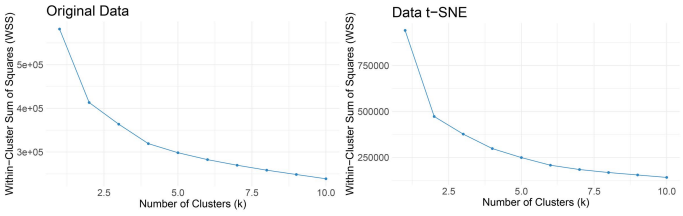

Figure 3. Elbow Method for Selecting the Number of Clusters (k) in the k-Means algorithm.

With k = 5, we found the clusters in our two datasets. The result on the t-SNE dataset is shown in figure 4. As we can see, compared with the true labels shown in figure 2, the clusters are much more balanced. This is likely not a suitable approximation of the real behavior of the data, in which guards were the majority class and occupied almost half of the point cloud.

3.2.2. DBSCAN

DBSCAN is how we usually refer to the Density-Based Spatial Clustering of Applications with Noise algorithm.

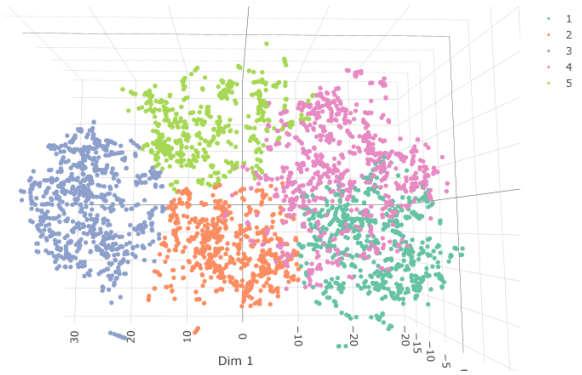

Figure 4. 5 Clusters found in the t-SNE dataset by the k-Means algorithm.

This technique is suited for problems where we can observe dense regions of data points separated by areas of lower density [4]. It uses two key parameters:

- **Epsilon**: The maximum distance between two points for them to be considered neighbors.
- **MinPts**: The minimum number of points required to form a dense region.

It alternates between labeling a point as noise or adding it to a cluster if it is within epsilon of another labeled point, until it reaches a stable configuration. The metric often used to evaluate the clustering created with DBSCAN is the **Silhouette Score**, which varies from -1 to 1 in which -1 is a misclassified point, 0 means a point in the region between clusters and 1 means a well-classified point. The average across all points gives an indication of the quality of the clustering.

In our project, the two mentioned hyper-parameters were tuned using a 5-fold cross-validation and trying to maximize the silhouette score. We can see in figure 5 how every combination of these parameters, on both datasets, performs quite poorly.

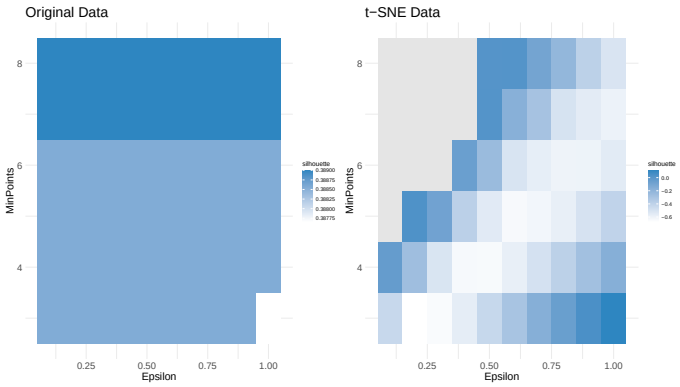

Figure 5. Heatmap for performance with hyper-parameters MinPoints and Epsilon in the DBSCAN Cross-Validation

DBSCAN did not detect any differences between the data in the original dataset, putting everything in the same cluster; and in the t-SNE dataset it divided the data in 472 clusters with 825 points classified as simple noise. We saw initially how there is not a significant spatial separation between the different player positions in our point cloud, which as mentioned is one of the cases this algorithm thrives in.

3.2.3. Hierarchical Clustering

Hierarchical Clustering is a method of clustering that seeks to construct clusters based on the distance of the points, in a hierarchical manner. This results in nested clusters where each level of hierarchy indicates the points are closer than the level before [1].

It computes a distance matrix between our points and assigns a cluster to each of them, and then starts merging clusters based on a criteria and updating the distance between clusters until all the data is merged in one cluster. This results in a dendogram where we can see the step by step joining of our points.

In this algorithm then we can also tune the **k** number of clusters to create, and the **method** of aggregation to use, which controls the decision of when to "merge" two data points based on the distance, using the silhouette score as mentioned before. The result of doing this in a 5-fold cross validation process can be seen in figure 6.

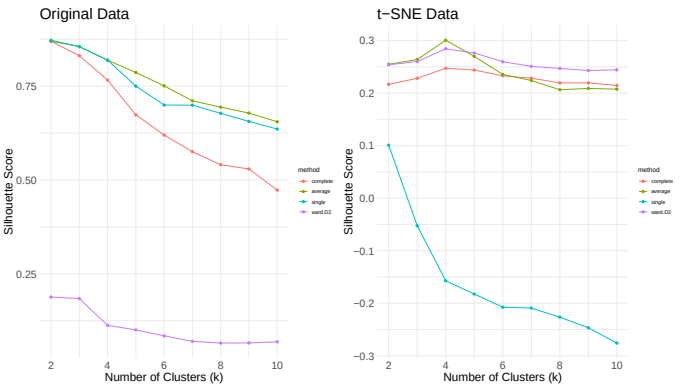

Figure 6. Silhouette Score vs K for each Method in Hierarchical Clustering Cross-Validation

This algorithm performed very poorly in the original dataset too, grouping the entire data into only one cluster. Across both datasets, the most congruent result was the one obtained for the t-SNE data. Where the k with best performance was 4, and the method was averaging the distances. The grouping made by these 4 clusters can be seen in the dendogram in figure 7. We can notice that guards are occupying the leftmost cluster, while forwards and centers appear in the other three groups. But it is

still not a perfect separation of these groups.

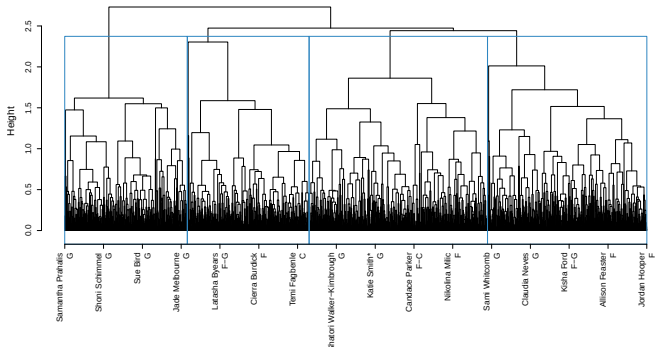

Figure 7. Dendogram of t-SNE data

3.2.4. Gaussian Mixture Models

Gaussian mixture models are a probabilistic approach that seeks to generate clusters by treating the data as if it was generated from a combination of Gaussian distributions [5]. Instead of simply labeling a point with a cluster, they assign probabilities to belong to the existing cluster, this makes them particularly good at overlapping data and outliers.

For this model, the number of clusters is referred to as **gaussian components (G)** and they are tuned to minimize the metric **Bayesian Information Criterion (BIC)**, which controls the trade-off between model complexity and clusters' variance.

As in the other cases, we did a 5-fold cross-validation, of which results can be seen in figure 8. Once again we see that the best model found in the cross-validation is the one where the entire data point cloud is one single cluster. Any other configuration results in a worse value for the evaluation metric.

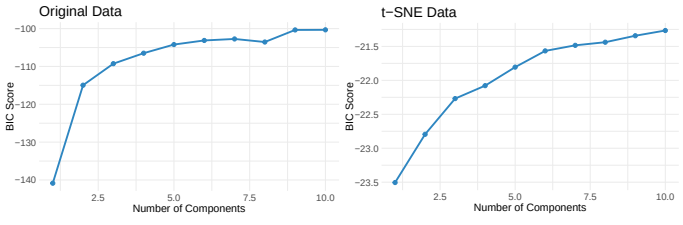

Figure 8. BIC vs Number of Components in GMM Cross-Validation

4. Results

In clustering, **homogeneity** is a measure of how much of a cluster's data points belong to the same class. It ranges from 0 to 1, where 1 means the cluster is fully uniform. We will use this score to assess how well each of our models performed.

In figure 9 and table 2 we can see the homogeneity scores of each of our approaches on both datasets. We can see that our best result was achieved with the t-SNE dataset and the k-means algorithm. And that, in all cases, the t-SNE dataset performs the best. But even the best case scenario has a low homogeneity at approximately 35%.

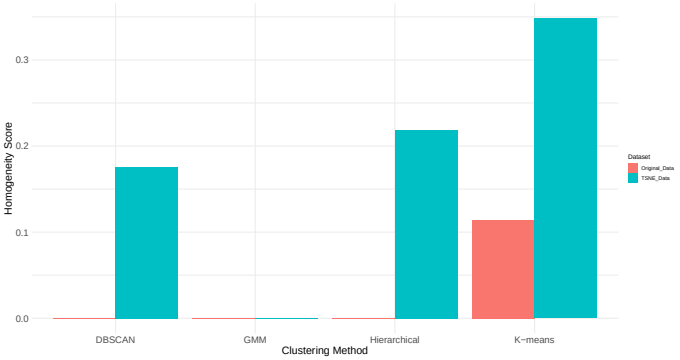

Figure 9. Homogeneity Scores for Clustering Methods

Table 2. Homogeneity Scores for Clustering Methods

Method	Original Data	t-SNE Data
K-Means	0.1139	0.3484
DBSCAN	0.0003	0.1755
Hierarchical	0.0001	0.2188
GMM	0.0000	0.0000

Unfortunately, since the data set was transformed using t-SNE it is not possible to interpret which features were important for each of the clusters. But using 5 k-means clusters on the original dataset, we can propose an idea of which clusters resemble which positions and which features were important for them, using the heatmap seen in figure 10.

From it we can surmise

- **Cluster 1**: Where the high values are FT%, X2P%, PTS, X3PA and X3P%. Very likely high-scorers, who are usually guards and forwards.
- **Cluster 2**: Where the high values are ORB, DRB, BLK X2PA and FTA. These are probably centers, who protect the rim and thus get fouled often but are not known as good shooters in general.
- **Cluster 3**: Where the values are moderate overall.
- **Cluster 4**: Where we have low values for FT%, X2P% and PTS, and a high value for TOV. Perhaps for players with low usage or those who don't contribute too

much on offense but may have a defensive impact not measured by these stats.

- **Cluster 5**: Where there are low values for TOV, X2PA, FTA, ORB, DRB and BLK. Maybe players who play in the perimeter and focus more on passing than shooting, like forwards.

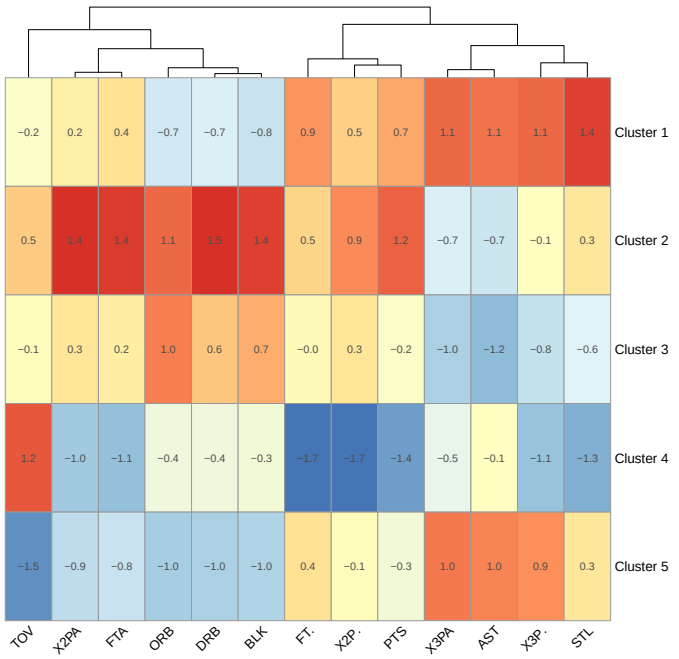

Figure 10. Heatmap Comparing Features and Clusters found with k-Means on the Original Dataset

5. Conclusion

This was a tricky data set to cluster. These well-known algorithms did not perform well with the data as is. However, we saw a few things moved the needle: removing outliers, scaling the data and ensuring the features remained uncorrelated made a difference in the performance of the algorithms. Another important aspect to consider in any future work is the balance of the data, in this case we saw that the guards were the majority class and it is hard to keep guards from any of the clusters predicted by the algorithms; in cases like this a synthetic oversampling of the other classes might improve the clustering.

These algorithms also suffered because of the shape of our data and the location of the classes. Our data cloud was too uniform and although there were clear regions for some of the minority classes, there was no separation between these regions and this hurt the performance of all algorithms. K-means performed the best, but it would have performed even better had the dataset contained clusters that are similar in size.

Another thing to consider is that although these features might not result in a data cloud with enough separation between the zones belonging to the different positions, there is a myriad of stats not considered in this work that could aid the clustering.

References

[1] IBM, *What is hierarchical clustering?*, Accessed: 2025-03-31, 2024. [Online]. Available: https://www.ibm.com/think/topics/hierarchical-clustering.

[2] Basketball Reference, *Basketball statistics and history*, 2025. [Online]. Available: https://www.basketball-reference.com/wnba.

[3] IBM, *K-means clustering*, 2025. [Online]. Available: https://www.ibm.com/think/topics/k-means-clustering.

[4] scikit-learn developers, *Clustering — scikit-learn 1.3.0 documentation*, 2025. [Online]. Available: https://scikit-learn.org/stable/modules/clustering.html#dbscan.

[5] scikit-learn developers, *Gaussian mixture models — scikit-learn 1.3.0 documentation*, Accessed: 31-Mar-2025, 2025. [Online]. Available: https://scikit-learn.org/stable/modules/mixture.html.

[6] Wikipedia contributors, *Women's national basketball association*, 2025. [Online]. Available: https://en.wikipedia.org/wiki/Women%27s_National_Basketball_Association.