NBA Scouting: A Data-Driven Approach

Ender Orman M Beşir Acar

Abstract

The National Basketball Association (NBA) has evolved into a data-driven league where analytics and technology play an important role in decision-making. Traditional methods of player scouting, based largely on subjective assessments and limited observations, are no longer sufficient to uncover hidden talent, predict future performance or ensure optimal team composition. Hence, our project aims to develop a ML model to assist basketball coaches in their scouting efforts through the analysis of NBA data. We planned to use various clustering and classification techniques such as K-Means, GMM, Logistic Regression, LightGBM, Random Forest etc. The data to be utilized consists of 270964 plays and 26 features in total. After segmenting player performance into four distinct clusters, we analyze the percentage of plays classified into the top-performing cluster for each position and identify high-potential transfer candidates. The player with the highest percentage will be identified as a possible transfer.

1. Introduction

In the NBA, teams employ extensive scouting departments that evaluate potential players through live game observations, video analysis, and performance statistics. Scouts assess not only a player's current abilities but also their potential fit within the team's system, taking into account factors such as playing style, team chemistry, etc. However, this traditional approach faces several limitations. Human scouts can only observe a finite number of games, and their assessments may be influenced by personal biases or recent performance fluctuations. Additionally, the complexity of modern basketball, with its numerous performance metrics, makes it increasingly difficult for human analysts to process all available information effectively. In this sense, we wanted to assign this task to the ML model we were going to develop by using sufficient amounts of play and trade data. After clustering and labeling the play data between 2015 and 2021 as a train set, we classify the 2022 and 2023 data into clusters using the best performing model among Logistic Regression, Random Forest and LightGBM methods.

2. Related Work

The first related study improves football player tracking by building better features that take into account the unique role of each player depending on the team's formation and style of play in soccer.[1] It takes inputs from the user or coach about the vacant position, desired player traits, current team lineup, playing style, and formation, and uses this data to run the team's requirements through an algorithm that returns a list of top N player candidates. This list is then fed into a ML model that analyzes team chemistry and provides a final ranking of the top players most suited for the team.

In order to validate and test the model, we thought it might be useful to select transfer candidates using the rumors and interviews of the relevant year through some NLP task. A similar approach examines whether linguistic signals from NBA players' pre-game interviews can provide useful information to predict their in-game actions and strategic choices beyond using past performance metrics.[2] The authors collected a dataset of interview transcripts and in-game metrics, and developed neural models to predict deviations from average player actions using the textual signals alone or combined with past performance data. Their text-based models

outperformed baselines using just performance metrics. In addition, they provided an interpretability analysis to understand what the models have learned.

Another similar study predicts the success of MLB (Major League Baseball) prospects from baseball scouting reports by developing a deep learning approach.[3] Given challenges like small dataset size, class imbalance (75–25 split), and highly specialized baseball language, the authors propose three key improvements: data augmentation, hierarchical pre-training using varying levels of sports language specificity, and a mixture of experts combining multiple pre-trained models. Their approach outperforms existing solutions in predicting player success from scouting reports.

In the most relevant study to our problem, based on NBA in-game statistics from the 2018-2019 season, dimensionality reduction was applied using Pearson Correlation, Variance Inflation Factor and PCA to extract six principal components explaining 61% of the data variance.[4] Players were clustered into nine groups based on their statistical tendencies such as rebounding, scoring efficiency, shooting distance and defensive impact.

3. Methodology

3.1. Datasets

After detailed research, we decided to fetch the play data to be utilized within the scope of the developed model from the BoxScoreAdvancedV2 endpoint provided by NBA API. In order to validate the model, we planned to make transfer predictions for the past few seasons as well. For this reason, we fetch 10 years of records from the 2015–16 season to 2024–25. Specified data consists of detailed player statistics for an individual game. The advanced attributes can be listed as follows:

• START_POSITION: The starting position of the player in the game (e.g., “G” for Guard, “C” for Center).

• MIN: Total minutes played by the player during the game.

• OFF_RATING: Points scored per 100 possessions by the team while the player was on the court.

• DEF_RATING: Points allowed per 100 possessions by the team while the player was on the court.

• AST_PCT: The percentage of teammate field goals that the player assisted while on the court.

• AST_TOV: Ratio of assists to turnovers made by the player.

• AST_RATIO: Assists per 100 possessions used by the player.

• OREB_PCT: Percentage of available offensive rebounds a player grabbed while on the court.

• DREB_PCT: Percentage of available defensive rebounds a player grabbed while on the court.

• REB_PCT: Percentage of total rebounds (offensive + defensive) a player grabbed while on the court.

•TM_TOV_PCT: The percentage of team possessions ending in a turnover while the player was on the court.

• EFG_PCT: A shooting percentage adjusted to account for the fact that three-point shots are worth more than two-point shots.

• TS_PCT: A shooting efficiency metric that accounts for field goals, three-pointers, and free throws.

• USG_PCT: The percentage of team plays used by the player while on the court (includes field goal attempts, free throw attempts, and turnovers).

• PACE: The number of possessions per 48 minutes by the player’s team while they were on the court.

• PACE_40: Adjusted pace statistic per 40 minutes instead of 48.

• POSS: Total number of possessions the player was involved in during the game.

• PIE: A metric that estimates a player’s overall statistical contribution to the game relative to the total statistics of the game.

Unlike the player stats, we created the transfer/trade dataset by scraping through sources like *Hoopshype* and *Bleacher Report*. This data includes transfer information for the 2022-23 and 2023-24 seasons and includes the following features:

- Team looking to transfer an athlete

- Transferred athlete
- All candidates that the team is interested in before the athlete joins the team
- Signature date of the transferred athlete
- Position of the transferred athlete

First, we divide our unlabeled play data into train and test. Plays from 2015 to 2021 are determined as train set, while plays from 2022 to 2024 are determined as test. We cluster and label train data according to existing performance metrics. The cluster methods to be used here are K-Means and GMM.

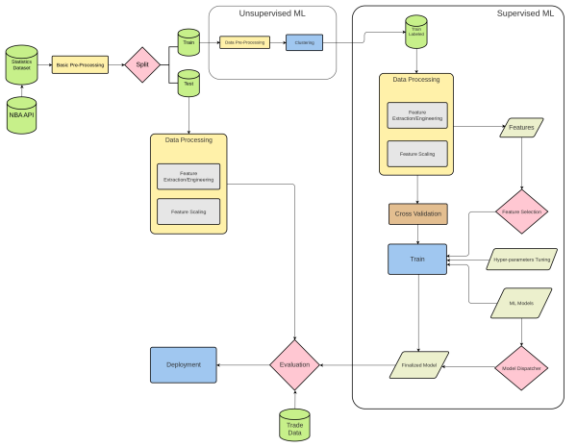

Figure 1. Flowchart of the developed model.

We plan to do some feature elimination before moving on to classification as can be seen in *Figure 1*. Because some metrics may affect the bias-variance tradeoff of the model. We try three different classifiers with the identified features, find out which one has a higher accuracy on the train data, and classify the clusters for the plays in 2022 and 2023 with the selected optimal classifier.

In the transfer data we collect for the 2022-23 and 2023-24 seasons, after finding the cluster percentages of the transfer candidates that the teams are interested in for any position, we check whether the teams have indeed selected one of the top three best performing athletes among the candidates. If we find a satisfactory result, we set up a pipeline for the plays in 2024, run the model with the candidates identified by the teams for the 2024-25 season and report the most suitable transfer candidate.

But of course, in this process, we need to proceed by optimizing some values. For example, how many clusters should be there to best represent the data or which hyperparameters should be used for the classifiers to give a good accuracy score. Some of the planned and required methods are described below.

3.2. WCSS

This is a metric used in the K-Means clustering algorithm to evaluate the compactness of clusters. It measures the total variance within each cluster by summing up the squared distances between each data point and its respective cluster centroid. Mathematically, it is expressed as:

$$WCSS = \sum_{i=1}^{k} \sum_{x \in C_i} \| x - \mu_i \|^2$$

where:

- k: Number of clusters
- C_i: The set of data points in cluster i.
- x: A data point in cluster i.
- μ_i: The centroid of cluster i.
- $\|x - \mu_i\|^2$: The squared Euclidean distance between the data point x and the centroid μ_i.

The goal of K-Means is to minimize WCSS. By plotting WCSS against the number of clusters, the "elbow point" indicates the number of clusters where adding more clusters does not significantly reduce WCSS.

3.3. Bayesian Gaussian Mixture

A Bayesian Gaussian Mixture extends traditional GMM by treating the model parameters as random variables and placing prior distributions on them. The key advantage is that it can automatically determine the effective number of components by using a variational inference approach. It can be described mathematically as:

$$P(x \mid \theta) = \sum_{k=1}^{K} \pi_k \cdot N(x \mid \mu_k, \Sigma_k)$$

where:

- π_k are mixing coefficients (weights)
- $N(x \mid \mu_k, \Sigma_k)$ is the Gaussian distribution with mean μ_k and covariance Σ_k
- θ represents all parameters $\{\pi, \mu, \Sigma\}$

In the Bayesian approach, we add priors:

- Dirichlet prior on mixing coefficients π
- Normal-Wishart prior on means μ and precisions Λ (inverse of covariance Σ)

The variational inference then optimizes the Evidence Lower BOund (ELBO):

$$ELBO = \mathbb{E}_{Q(Z,\theta)}[\log P(X,Z,\theta)] - \mathbb{E}_{Q(Z,\theta)}[\log Q(Z,\theta)]$$

where:

- $Q(Z,\theta)$ is the variational distribution
- Z are the latent variables (cluster assignments)

The key to automatic component selection is that the variational inference can effectively "shut down" unnecessary components by setting their mixing coefficients close to zero.[5] This happens when components don't significantly improve the model fit, and allows the model to automatically determine the effective number of clusters.

3.4. Grid Search

Grid search is a method for hyperparameter tuning that performs an exhaustive search over a specified parameter grid. It evaluates all possible combinations of hyperparameters using cross-validation to find the best set based on a chosen performance metric (e.g., accuracy or F1-score). After defining the model, a grid of hyperparameter values, and the number of cross-validation folds, grid search trains and evaluates the model for each combination, selecting the one that achieves the best cross-validation score. We want to use this method to optimize classifiers, although it is computationally expensive for large datasets like ours.

4. Experimental Results

4.1. PCA

When you look at it, we have a lot of player statistics. But we know that models like K-Means clustering typically give better results in low-dimensional spaces. Therefore, before clustering the play data between 2015 and 2021, Principal Component Analysis (PCA) was applied to reduce its dimensionality.

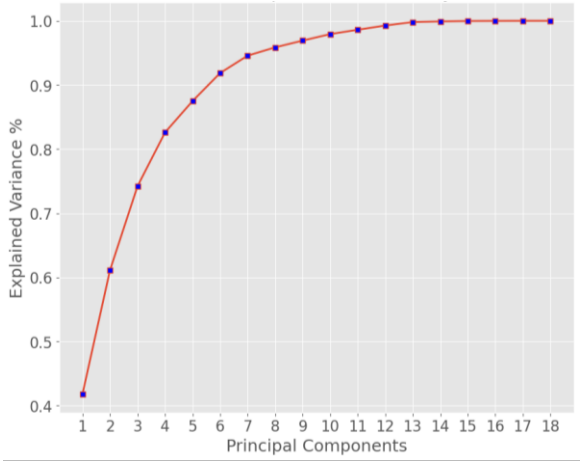

Figure 2. Explained variance by number of components.

By looking at the curve in *Figure 2*, we opt for the 4-component solution as it explains at least 80% of the variance in the population.

4.2. K-Means Clustering

The K-Means is applied to the data which transformed with the components we obtained from PCA. Moreover, the optimal number of clusters is determined using the Within-Cluster Sum of Squares (WCSS) as we want some well-separated clusters.

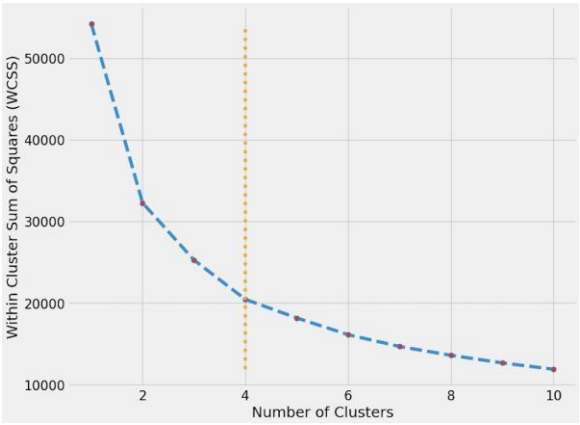

Figure 3. WCSS curve for number of clusters.

The WCSS is calculated for k ranging from 1 to 10 as can be seen in *Figure 3*. By analyzing the "elbow" in the curve, the ideal number of clusters was identified as 4, as the reduction in WCSS became negligible beyond this point.

While the resulting clusters were generally satisfactory, some overlap was observed between the

clusters based on the first two principal components as can be seen in *Figure 4*.

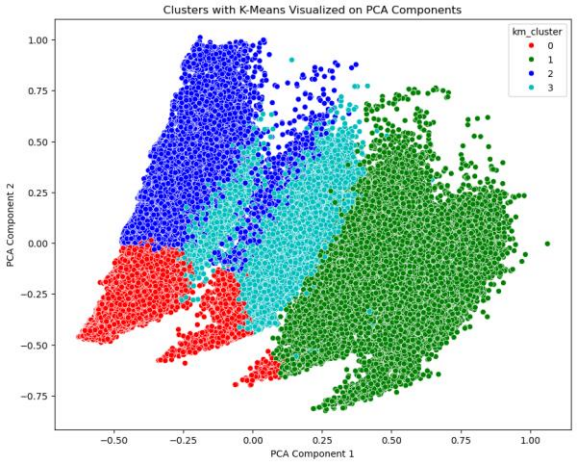

Figure 4. Clusters with K-Means visualized on PCA components.

4.3. Gaussian Mixture Models

Since K-means did not achieve the desired level of distinction in the clusters, we try another clustering method, Gaussian Mixture Models (GMMs).

GMMs is distribution-based model, rather than distance-based like K-Means. They do not assume clusters to be of any geometry, like K-Means which bias the cluster sizes to have specific structures (circular). Furthermore, they work well with non-linear geometric distributions.

The main disadvantages concerns its potential quick convergence to a local minimum, which is not optimal. However, we can adjust its parameters appropriately.

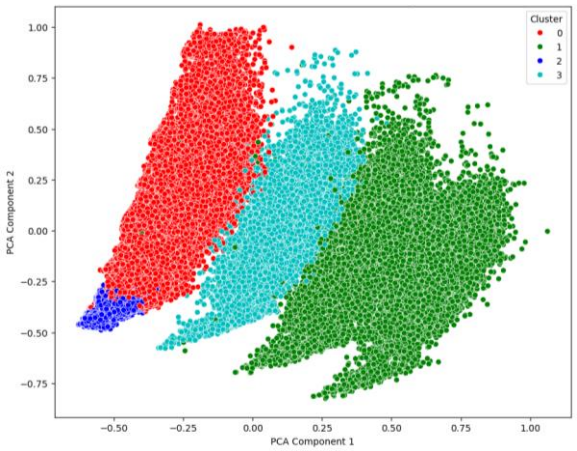

Figure 5. Clusters with GMMs visualized on PCA components.

We used Bayesian Gaussian Mixture to pick the number of clusters. In brief, the weights of clusters are returned, erroneous clusters weighted below 0.10 and automatically removed. As a result, we have 4 distinct clusters that are better separated than K-Means, as can be seen in *Figure 5*. Balance of the clusters shown in the *Figure 6* below.

```
0    83023
1    52314
3    34475
2    17993
```

Figure 6. Total number of data points in each cluster.

We also check in which clusters each position (Center, Guard and Forward) gets the highest performance values:

```
C    cluster_1
G    cluster_3
F    cluster_3
```

Figure 7. Best performing cluster in each position

Let's keep the *Figure 7* in mind because we assume that when a team wants to make a transfer at any position, the vast majority of his plays should be classified to the cluster that gives the highest performance values for that position.

4.4. Feature Elimination

After labeling the train dataset with GMMs, we will train several classifiers in order to predict the testing dataset's clusters. But at the same time, we need to apply validation so that we don't run into overfitting and underfitting issues. The skewed clusters (*Figure 5*) implies that a Stratified K-Fold cross validation has to be chosen over a random one. This will keep the labels' ratio constant in each fold and whatever metric we choose to evaluate, it will give similar results across them all.[6] And speaking of metrics, the F1-score (harmonic mean of precision and recall) looks more appropriate than accuracy, since the targets are skewed.[7]

In our first attempt in Logistic Regression with normalized features (mapped with `_n` suffix), we achieved 99.64% accuracy. But we shouldn't be comfortable with such a tremendous score from the

very beginning. This is where feature importance comes into play.

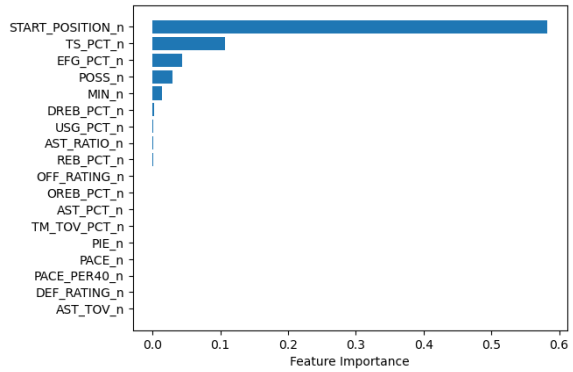

Figure 8: Permutation feature importance with all features.

Using the START_POSITION_n feature alone, we achieved an impressive accuracy of 90% in our Logistic Regression model. However, this feature dominates the model and causes overfitting, as it outperforms the other features significantly (*Figure 8*). Also OFF_RATING, AST_TOV, EFG_PCT, TS_PCT & POSS get the minimum levels for the START_POSITION 0 or NaN (we encoded the NaN positions with 0). Which means that this variable *betrays* that those players didn't start the game, hence there is high possibility for them to have played less time than the other and as a result have worse stats. To be more particular, the less you play the lower the chance to increase any records (pass, points, etc). In the same context, another variable may also be guilty; MIN. It precisely expresses the time a player spent in the court and so we have to ignore it, too. In order to confirm let's try it again after removing START_POSITION_n.

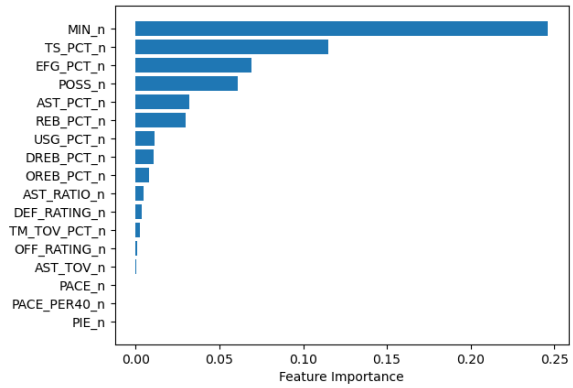

Figure 9: Permutation feature importance after removing START_POSITION_n.

As can be seen in *Figure 9*, the same applies for the case of MIN - it leaks information of the time spent in the court by the athlete. So, the model knows *'apriori'* that the player with higher duration may have better stats.

After removing these two attributes, we can apply PCA to the normalized dataset. The reason for applying this is that if we predict the clusters with the components obtained with PCA, maybe we can get higher accuracy than the original features.

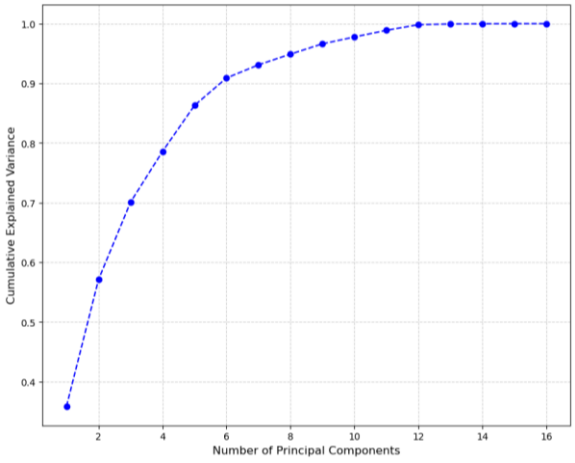

Figure 10: Explained variance by number of components.

We can choose any number above 6, since it will explain more than %90 of variance (*Figure 10*). In our case, we chose 7 as optimal numbers of components. So, we will initialize the Logistic Regression again. It should be noted that we still have many features and removing some of them can improve the performance of the models.

4.5. Logistic Regression

Initially, we selected a subset of 12 features from the dataset after applying recursive feature elimination. Trained model achieved a validation accuracy of 71.95%. We can infer that the model is not overfitted with this level of accuracy. However, in order to further enhance performance, we conducted a grid search to systematically tune the inverse regularization parameter, *C*. Testing six values ranging from 0.001 to 100, we observed the highest F1-score (0.7151) at *C*=100. Using this optimized hyper-parameter, the model was re-initialized and achieved a slightly improved accuracy of 72.00%. Additionally, we trained the tuned model with PCA components instead

of the original features and obtained an accuracy of 71.61%.

4.6. Light Gradient-Boosting Machine

When we first train the model with the 12 selected features in LightGBM, we get an accuracy of 72.84%. Hyperparameter optimization was performed using grid search, which yielded an F1-score of 0.7218. The optimal hyperparameters identified during the search are as follows:

- colsample_bytree: **0.8**
- learning_rate: **0.1**
- max_depth: **-1** (means no limit)
- min_child_samples: **20**
- n_estimators: **100**
- num_leaves: **50**
- subsample: **0.8**

The tuned LightGBM model achieved a validation accuracy of 72.81%. Furthermore, training with PCA components reduced the accuracy of the tuned model to 72.06%.

4.7. Random Forest

Like previous approaches, we first trained the Random Forest model with selected 12 features and obtained an accuracy of 72.13%. After grid search, we determined the optimal hyperparameters and achieved an F1-score of 0.7189. The best-performing parameters identified are as follows:

- criterion: **gini**
- max_depth: **15**
- n_estimators: **500**

Using these optimized parameters, the trained model achieved an accuracy of 72.60%. In comparison, training the model on components obtained through PCA resulted in a slightly lower validation accuracy of 71.89%.

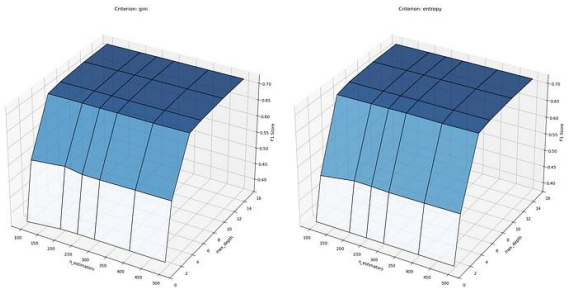

Figure 11: Surface plots of Random Forest classifier for each hyper-parameter.

As can be observed in the surface plots (*Figure 11*) visualizing the grid search for Random Forest (two plots for two criterion: *gini* and *entropy*), F1-score converges as the depth of the trees (*max_depth*) increases. However, increasing the number of decision trees (*n_estimators*) in the forest appears to have minimal effect on the overall performance.

4.8 Evaluation

Ultimately, we use LightGBM, which yields the highest metric scores among the tuned models to predict the clusters of the 2022 and 2023 plays.

Assuming that teams will consider athletes' performances in the previous year when making transfers, we use the 2022 data for athletes transferred in the 2022-23 season and the 2023 data for athletes transferred in the 2023-24 season. If you remember the structure of our trade/transfer data, it includes the transfer candidates that a team is interested in at any position and the preferred athletes among these candidates. Let's go through an example for better understanding:

The Dallas Mavericks have identified 5 different candidates for the 2022-23 season due to the need for a Guard. Candidates are, respectively: Fred VanVleet, Dejounte Murray, Jaden Ivey, Kyrie Irving and Collin Sexton. The player they chose among these candidates was Kyrie Irving and his transfer took place on February 6, 2023.

We know that, `cluster_3` gives the best performance criteria for the Guard position. So, what we're going to do is first find out which cluster these candidates' 2022 plays classified into and then determine what percentage of them classified to

`cluster_3`. If the percentage of Kyrie Irving's plays that classified to `cluster_3` is in the top three of these candidates, then we can say that the model we have chosen has done a good job in classifying the plays to the respective clusters. In other words, we prove that the team is indeed making transfers based on performance.

When we apply this for each record in the trade dataset, we see that 77.78% of the transfers in the 2022-23 season and 70.59% in the 2023-24 season were in the top three in terms of performance among the candidate players. In this context, we can say that our model performs well overall. However, our scores are not close to perfect as there are many factors such as teams' preferences are not always performance-oriented, sometimes they are financially insufficient, athletes who are offered to join the team do not accept or the team looks for substitutes instead of starters, etc.

5. Conclusions

As an output of this whole process, with the pipeline we built for the plays in 2024, after determining the transfer candidates of any team in any position for the 2024-25 season, the model can be run and the most suitable transfer candidate for that team can be determined.

Of course, the model could be improved in the future to provide better transfer suggestion. For instance, NLP techniques can be used to analyze transfer rumors and news to identify more precise candidate athletes in transfer/trade data. Furthermore, increasing the amount of data can influence the optimal number of clusters and improve clustering results.

6. References

[1] S. Ghar, S. Patil, and V. Arunachalam, ”Data Driven football scouting assistance with simulated player per- formance extrapolation,” 2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA), Pasadena, CA, USA, 2021, pp. 1160-1167.

[2] Nadav Oved, Amir Feder, and Roi Reichart, ”Predicting In-Game Actions from Interviews of NBA Players,” *Computational Linguistics*, vol. 46, no. 3, pp. 667–712, 2020.

[3] A. Malhotra, E. Maheshwari, and A. Singh, ”Mix- ture of Hierarchical Unified Neural Domain Experts (MHUNDE): Transforming Scouting in Major League Baseball,” Stanford CS224N Custom Project, 2020.

[4] Ahmed Jyad, ”Redefining NBA Player Classifications Using Clustering,” *Towards Data Science*, 2020. [Online].

[5] Corduneanu, Adrian and Charles M. Bishop. “Variational Bayesian Model Selection for Mixture Distributions.” *Microsoft Research* (2001).

[6] A. Thakur, “Approaching (Almost) Any Machine Learning Problem”, 1st edition (2020), ISBN-10: 9390274435

[7] Bex T., “Comprehensive Guide to Multiclass Classification Metrics.”, *Towards Data Science,* 2021. [Online].