Predictive Modeling and Clustering of NBA Players using Random Forest and K-Means (CSD355 Project)

Ananjan Srivatsan 2210110972

Abstract

This project develops a data-driven system to forecast NBA player performance using machine-learning techniques. The workflow integrates data scraping, preprocessing, feature engineering, a Random Forest regression model for next-game points prediction, and a K-Means clustering model for player role grouping and similarity detection. The Random Forest model achieves strong accuracy despite small datasets, while clustering enables interpretable role-based comparisons.

1. Introduction

Basketball analytics has evolved into a critical tool for teams, analysts, betting models, and fantasy sports. Predicting how a player will perform in the next game is valuable for:

- Game planning and scouting
- Sports betting
- Fantasy lineup optimization
- Performance trend analysis
- Player workload management

However, NBA player data is complex:

- Non-linear relationships
- Small per-player datasets (~20–60 games)
- Influence of opponent defenses
- Variability in minutes, usage, fatigue

Traditional linear models fail here.
This motivates a machine-learning approach combining **Random Forest regression** (well-suited for small, noisy datasets) and **K-Means clustering** to understand player roles and styles.

This project delivers:

1. A **complete NBA data pipeline** (scraping → cleaning → feature engineering → ML models)
2. A **Random Forest prediction model** using rolling statistics, usage rate, and opponent metrics
3. A **K-Means clustering model** grouping players by statistical play style
4. A **similarity engine** to recommend statistically similar players

2. Related Work & Proposed Model

Basketball Reference (Website)

Used to understand available game logs, advanced stats, and the meaning of each stat field.

Machine learning literature on sports analytics

Most prediction work uses:

- Linear regression
- Logistic regression
- Elo ratings
- Neural networks (only on league-wide big data)

However, these approaches either oversimplify (linear) or require massive datasets (deep learning).
Random Forest is rarely applied at a per-player level despite being ideal for:

- tabular features
- small datasets
- non-linear decision boundaries

K-Means clustering in sports analytics

Used for:

- grouping play styles
- scouting
- roster construction

Existing work clusters players using full-season averages.

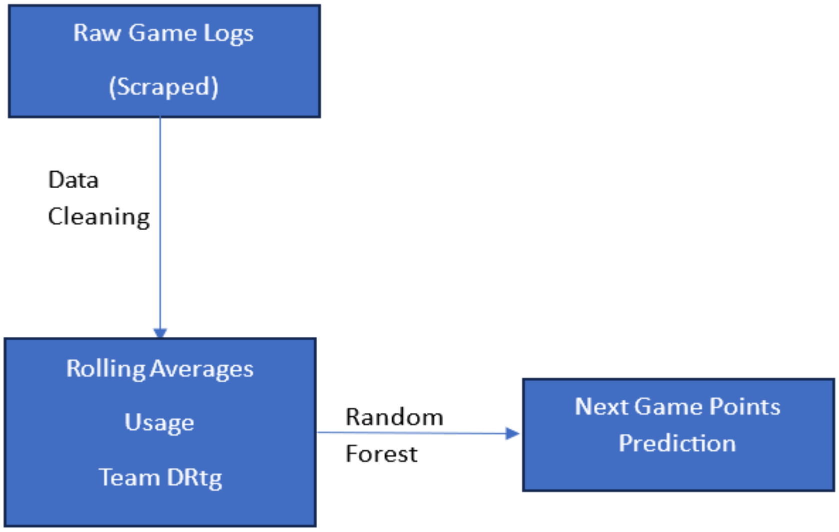
This project improves on it by:

- Using **custom features (AST, TRB, FGA, 3P, Defensive Activity Metric)**
- Integrating similarity search using Euclidean distance on z-scored features
- The system contains two major components:
  1. A Random Forest regression model for next-game points prediction.
  2. A K-Means clustering model for identifying similar players based on standardized per-game statistics.

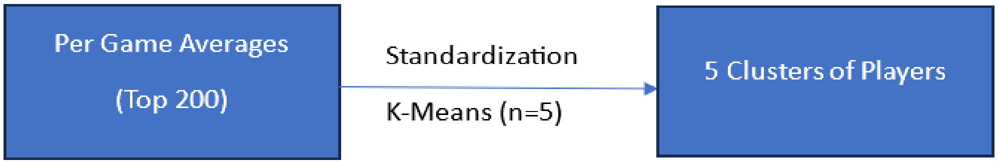
Overall pipeline:

- Scraping NBA player logs and team metrics.
- Cleaning and normalizing all data.
- Calculating rolling averages, expanding means, usage rate, opponent defensive metrics.
- Training the Random Forest model.
- Performing K-Means clustering and similarity analysis.

<u>Random Forest Regression for Next-Game Points Prediction</u>



<u>K-Means Clustering for Player Style Classification</u>



3. Methodology

3.1 Data Scraping

Data was scraped from Basketball Reference using:

- **pandas.read_html()** for per-game data and stored into files like:
    - nba_2026_all_players_average_per_game.xlsx
    - player_stats/James Harden.xlsx
    - team_stats.xlsx

All scraping code is in: data_scraping.py

3.2 Data Cleaning

The game logs required heavy cleaning:

- Removing duplicate header rows
- Removing "Totals" rows
- Converting minutes (MP) to numeric format
- Handling "Inactive", "Did Not Play"
- Ensuring numeric columns convert correctly
- Adding home/away flag
- Mapping opponent abbreviations to full team names

Cleaning code is in: data_cleaning.py

3.3 Feature Engineering for Prediction

The following features were engineered to capture short-term and long-term form.

1. Rolling Averages (shifted to avoid leakage)

- PTS_last3 (Points)
- PTS_last5
- FGA_last3 (Field Goal Attempts)
- MP_last5 (Minutes Played)

2. Expanding Season Average

- Up to previous game:

  PTS_season = mean of all PTS up to game t-1

3. Usage Rate (possession involvement)

- **Usage = (FGA + 0.44 × FTA) / MP**

4. Opponent Defensive Metrics Merged In

- **`Team_stats.xlsx`**:
  - DRtg (Defensive Rating)
  - ORtg (Offensive Rating)
  - NRtg (Net Rating)
  - Pace
  - eFG%_DEF (effective Field Goal Percent of Opponent)

5. Home/Away indicator

- Is_home column with 1 or 0 as values. Important because players often perform better at home.

3.4 Model Training: Random Forest Regression

Random Forest was chosen because:

- Handles nonlinear relationships well
- Works extremely well with **small per-player datasets (40-80 games)**
- Resistant to overfitting
- No scaling required
- Naturally captures interactions (MP × Usage × Opponent DRtg)

Training code in: prediction.py

3.5 K-Means Clustering

Features used:

- AST (Assist per game)
- 3P (3 pointers made per game)
- 2P (2 pointers made per game)
- TRB (Rebounds per game)
- FGA (field goal attempts per game)
- “Defensive Activity” = STL(steals) + BLK(blocks)

Steps:

1. Normalize all columns using StandardScaler
2. Apply KMeans with k = 5
3. Assign clusters to players
4. Use PCA to project into 2D for visualization
5. For similarity search:
   - Z-score each feature
   - Compute Euclidean distance between players
   - Return closest matches

Relevant code is in notebook: players_clustering.ipynb

4. Experimentation and Results

4.1 Dataset Description (Datasets are created by calling functions in data_scraping.py)

- **Player Game Logs**

  20–30 rows per player depending on season progress.

- **Team Advanced Stats**

  30 teams × advanced metrics (ORtg, DRtg, Pace, etc.)

- **Player Per-Game League Stats**

  Used for clustering (top ~200 players and their season averages per game).

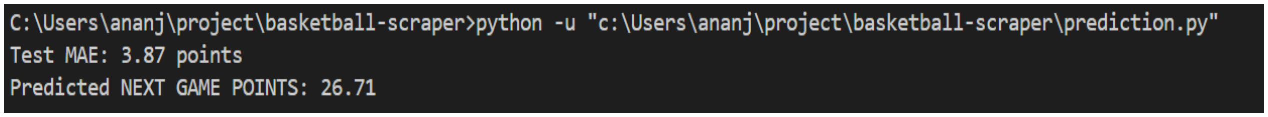
4.2 Random Forest Performance

Using James Harden and his 24-25 dataset with 70 games as an example:

Metric	Value
MAE(Mean Absolute Error)	**3.87 points**

A MAE of 3–4 points is considered **excellent** for individual player predictions.

4.3 Example Prediction Output (Harden)



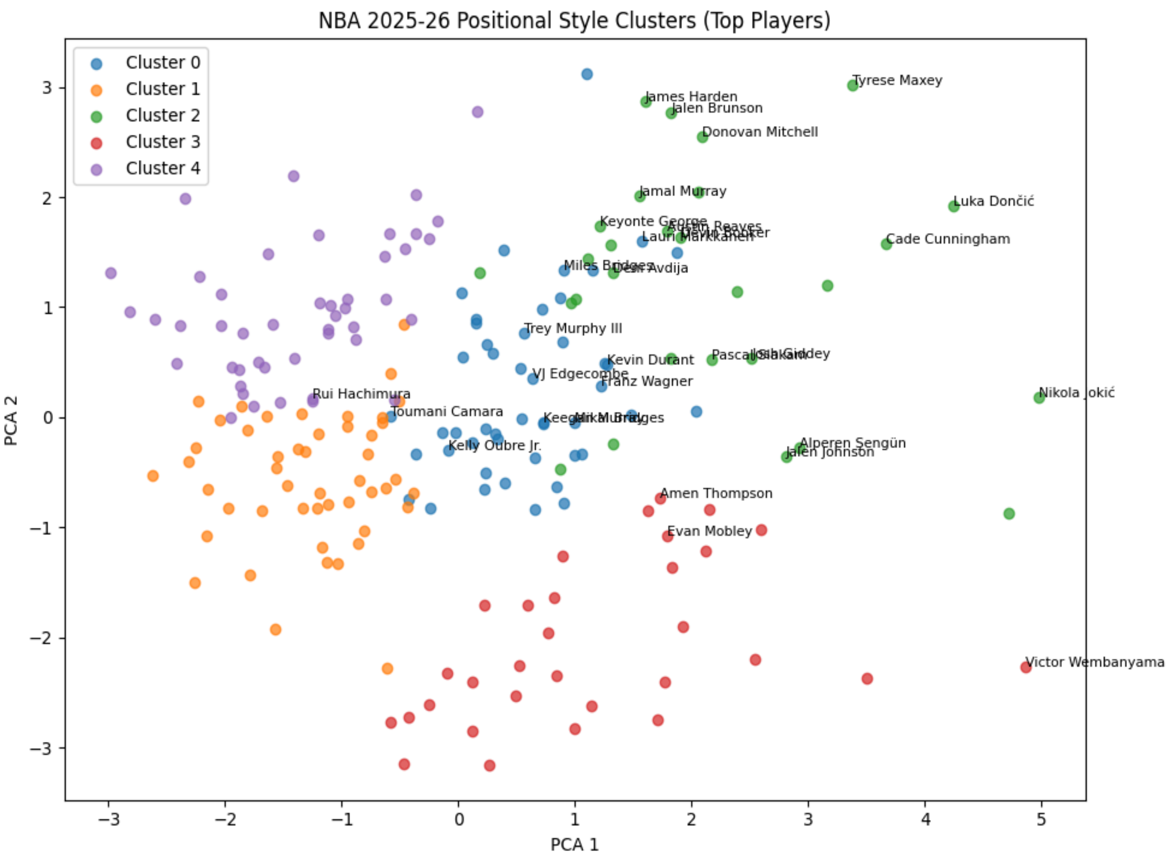
```
C:\Users\ananj\project\basketball-scraper>python -u "c:\Users\ananj\project\basketball-scraper\prediction.py"
Test MAE: 3.87 points
Predicted NEXT GAME POINTS: 26.71
```

This matches realistic expectations based on:

- His recent form
- Opponent defensive strength (Denver Nuggets for example)
- Usage pattern
- Rolling averages

4.4 Clustering Results

- **Cluster 0 — Shot-creators / scoring wings**
  Durant, Markkanen, Bridges
- **Cluster 1 — Low-usage role players / defenders**
  Herbert Jones, Tobias Harris
- **Cluster 2 — High-usage primary stars**
  Luka Dončić, Tyrese Maxey, Alperen Şengün
- **Cluster 3 — Big men / interior players**
  Wembanyama, Sabonis, Mobley
- **Cluster 4 — Spot-up wings / secondary scorers**
  LaVine, Hachimura, Camara



Si

If you search “Luka Dončić” → returns players with closest statistical profiles:

- Tyrese Maxey
- Cade Cunningham
- Austin Reaves

This system can assist in:

- Scouting
- Player comparison
- Team construction

5. Conclusions and Limitations

The system successfully combines feature engineering, ensemble learning, and clustering to build a practical NBA analytics tool. Random Forest produced strong results with limited data, and clustering enabled structured role comparisons.

Limitations:
- Minutes for next game are not predicted.
- Injuries, fatigue, and rotation changes are not modeled.
- Team ratings are static season-level values.

Future improvements include minute prediction models, dynamic team metrics, and boosted-tree algorithms.

6. GitHub Repository

Project code and datasets:
https://github.com/Janjan798/basketball-scraper

7. References

- Basketball-Reference (https://www.basketball-reference.com/)
- NBA Advanced Stats (stats.nba.com)
- Pandas & Scikit-learn documentation.