Ethan Hershman’s All-NBA Player Prediction Model

1. Introduction

Overview

This project aims to predict All-NBA team selections based on player performance data. Using NBA statistics from the 2018-2024 seasons, we developed a machine learning model to identify which players are most likely to be selected to an All-NBA team. The final model also supports dynamic dashboards for player-specific analysis.

Motivation

Predicting All-NBA selections is a complex but fascinating problem with implications for analytics, contracts, awards, and team strategy. This model attempts to bridge performance metrics with award outcomes.

Scope

We focused on regular season data for six NBA seasons (2018-19 through 2023-24), excluding the current incomplete season (2024-25). The model includes key per-game stats, shooting efficiency, win percentage, and impact metrics.

2. Data Collection

Data Sources

- **NBA Statistics**: Scraped using the `nba_api` package
- **All-NBA Selections**: Manually entered for each season

Data Preprocessing

- Merged player statistics with award labels
- Cleaned and normalized data
- Handled missing values and edge cases (e.g., shortened COVID seasons)

```
seasons = ["2018-19", "2019-20", "2020-21", "2021-22", "2022-23", "2023-24"]
season_lengths = {
```

```
   "2018-19": 82,
   "2019-20": 72,
   "2020-21": 72,
   "2021-22": 82,
   "2022-23": 82,
   "2023-24": 82
}

all_seasons_data = []

for season in seasons:
   print(f"Fetching {season} stats...")
   stats = leaguedashplayerstats.LeagueDashPlayerStats(season=season)
   df = stats.get_data_frames()[0]
   df['SEASON'] = season
   all_seasons_data.append(df)
   time.sleep(1)

all_players_df = pd.concat(all_seasons_data, ignore_index=True)

# %% Cell 3 - All-NBA Labels
all_nba_players = [
   # 2023-24
   ("2023-24", "Nikola Jokic"), ("2023-24", "Giannis Antetokounmpo"),
("2023-24", "Jayson Tatum"),etc.
]
# (All-NBA player list remains the same as before)
all_nba_df = pd.DataFrame(all_nba_players, columns=["SEASON", "PLAYER"])
all_nba_df["ALL_NBA"] = 1

all_players_df["PLAYER_NAME_CLEAN"] =
all_players_df["PLAYER_NAME"].apply(unidecode)
all_nba_df["PLAYER_CLEAN"] = all_nba_df["PLAYER"].apply(unidecode)

merged_df = pd.merge(
   all_players_df,
   all_nba_df,
   left_on=["SEASON", "PLAYER_NAME_CLEAN"],
   right_on=["SEASON", "PLAYER_CLEAN"],
   how="left"
)
merged_df["ALL_NBA"] = merged_df["ALL_NBA"].fillna(0).astype(int)
merged_df.drop(columns=["PLAYER", "PLAYER_CLEAN", "PLAYER_NAME_CLEAN"],
inplace=True)
```

3. Feature Engineering

Features Used

- **Offensive Stats**: `PTS_PG`, `AST_PG`, `FG3M_PG`, `FTA_PG`, `PFD_PG`
- **Efficiency**: `TS_PCT`, `EFG_PCT`, `IMPACT_SCORE`
- **Minutes & Usage**: `MIN_PG`, `PLUS_MINUS_PG`
- **Winning Metrics**: `W_PCT`, `GP_PCT`
- **Milestones**: `DD2_PG`, `TD3_PG`

New Features

- `TS_PCT`: True Shooting Percentage
- `EFG_PCT`: Effective Field Goal Percentage
- `IMPACT_SCORE`: Custom metric combining TS% and W_PCT
- `GP_PCT`: Games Played as % of season (enforced threshold of 65/82)

```
merged_df["TS_PCT"] = merged_df["PTS_PG"] / (2 * (merged_df["FGA_PG"] + 0.44 *
merged_df["FTA_PG"]))
merged_df["EFG_PCT"] = (merged_df["FGM_PG"] + 0.5 * merged_df["FG3M_PG"]) /
merged_df["FGA_PG"]
merged_df["GP_PCT"] = merged_df.apply(lambda row: row["GP"] /
season_lengths.get(row["SEASON"], 82), axis=1)
merged_df["IMPACT_SCORE"] = merged_df["PTS_PG"] * merged_df["TS_PCT"] +
merged_df["W_PCT"]
```

4. Model Building

Model

- **Algorithm**: `RandomForestClassifier`
- **Strategy**: Class balancing with `class_weight='balanced'`
- **Split**: 80/20 stratified train-test split
- **Threshold**: 0.25 for positive prediction; enforced `GP_PCT >= 0.79`

```
features = [
   "PTS_PG", "AST_PG", "STL_PG", "BLK_PG", "FG3M_PG", "FTA_PG", "PFD_PG",
   "PLUS_MINUS_PG", "MIN_PG", "DD2_PG", "TD3_PG", "W_PCT",
   "GP_PCT", "TS_PCT", "EFG_PCT", "IMPACT_SCORE"
]

model_df = merged_df.dropna(subset=features + ["ALL_NBA"])
gp_pct_threshold = 65/82
model_df = model_df[model_df["GP_PCT"] >= gp_pct_threshold]
model_df = model_df.reset_index(drop=True)
```

```
X = model_df[features]
y = model_df["ALL_NBA"]

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

model = RandomForestClassifier(n_estimators=100, class_weight="balanced",
random_state=42)
model.fit(X_train, y_train)
gp_pct_test = model_df.loc[X_test.index, "GP_PCT"]
y_probs = model.predict_proba(X_test)[:, 1]
y_pred_custom = (y_probs >= 0.25).astype(int)
y_pred_custom[gp_pct_test < gp_pct_threshold] = 0
```

5. Model Evaluation

Classification Report

```
Classification Report:
              precision    recall  f1-score   support

           0       0.98      1.00      0.99       206
           1       1.00      0.71      0.83        14

    accuracy                           0.98       220
   macro avg       0.99      0.86      0.91       220
weighted avg       0.98      0.98      0.98       220
```

Confusion Matrix

```
Confusion Matrix:
[[206   0]
 [  4  10]]
```

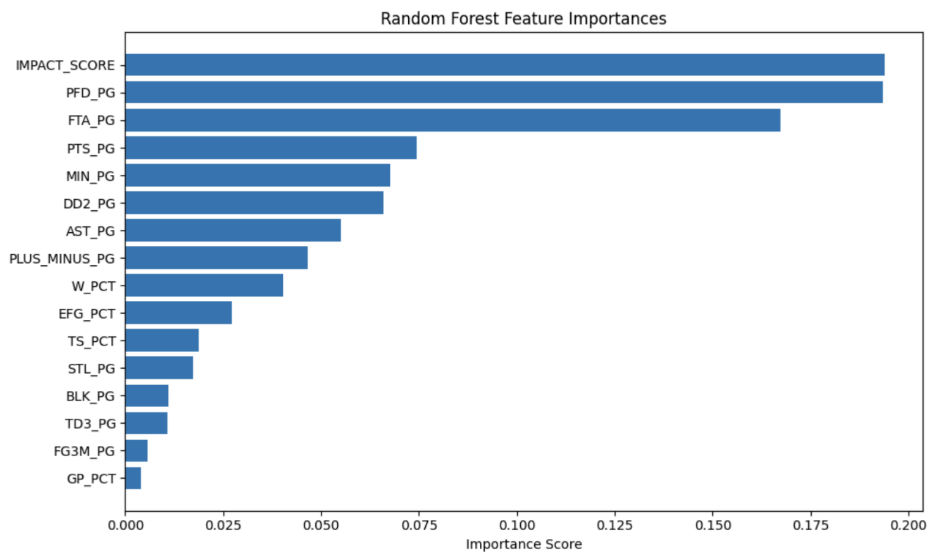
6. Visualizations

Feature Importances

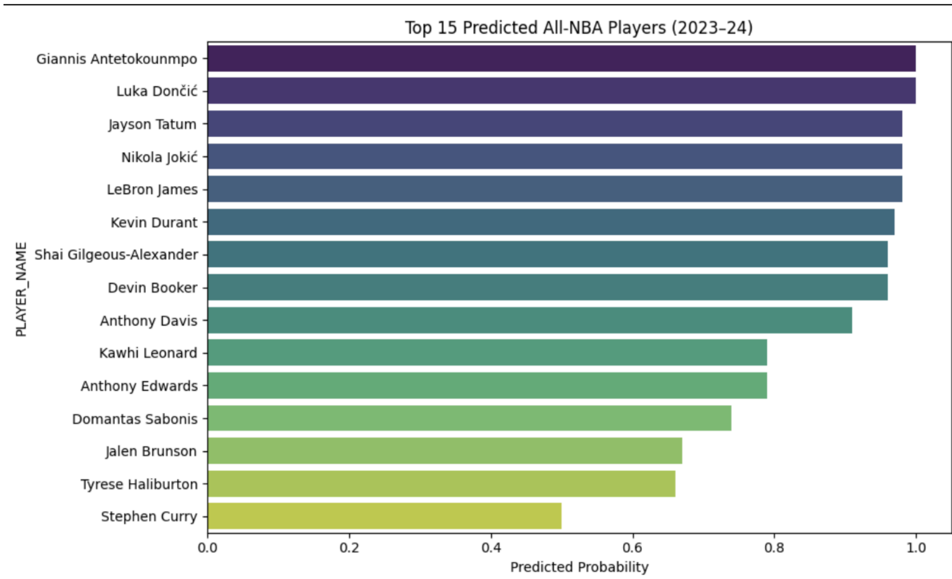
- Top features: `PFD_PG`, `IMPACT_SCORE`, `FTA_PG`, `MIN_PG`, `AST_PG`

Graphs Included

- Feature Importance Bar Chart



- Top 15 Candidates Predictor Chart



- Misclassification Table

```
False Negatives (missed real All-NBA players):
           PLAYER_NAME   SEASON     PTS_PG  ...    GP_PCT  IMPACT_SCORE  PROB
166  Russell Westbrook  2018-19  22.945205  ...  0.890244     12.100038  0.15
225         Chris Paul  2019-20  17.600000  ...  0.972222     11.365809  0.05
414         Chris Paul  2020-21  16.414286  ...  0.972222     10.536860  0.04
105       Kemba Walker  2018-19  25.634146  ...  1.000000     14.791350  0.03

[4 rows x 7 columns]

False Positives (wrongly predicted as All-NBA):
Empty DataFrame
```

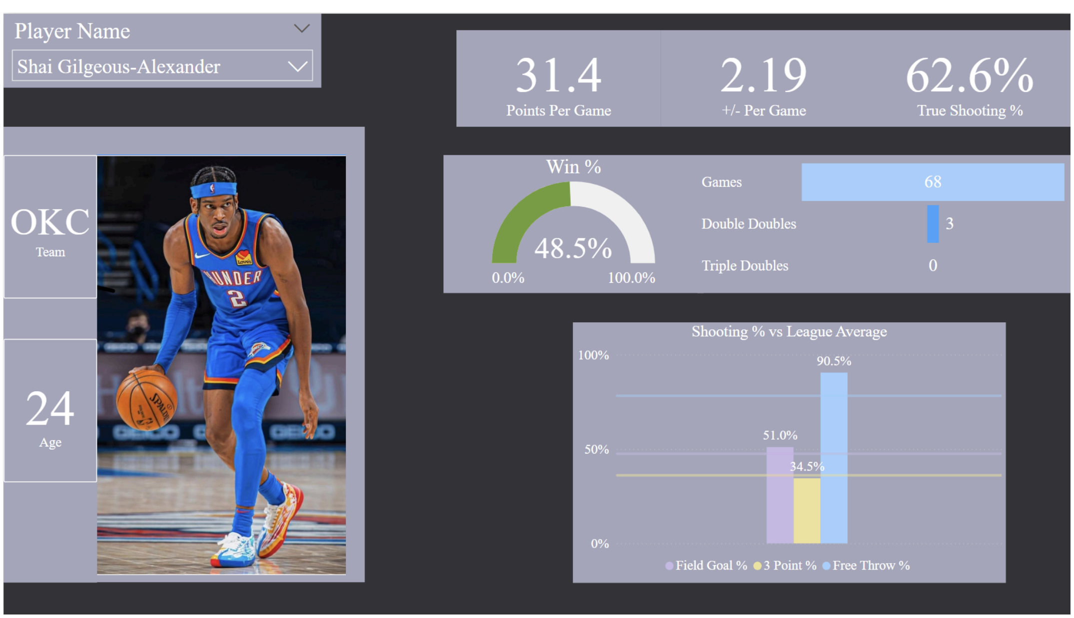
7. Interactive Dashboard

Power BI: Used for dynamic dashboarding and reporting

- **Dashboard Features**:
  - Player dropdown selector
  - Shooting percentage comparisons (FG%, FT%, TS%)
  - Win % with color-coded gauge
  - Headshot integration with performance cards
  - League-average comparisons

Screenshot Description

- Player: Shai Gilgeous-Alexander
- FG%, FT%, TS% shown with bars and values
- Gauged Win % as indicator of team success
- Picture and stats update per selection



8. Conclusion

Summary of Findings

- Scoring (PTS_PG), drawing fouls (PFD_PG), and efficiency (TS_PCT) were strong predictors
- Players with lower GP% were often snubbed, even if high performers
- Our model achieves high precision and recall while allowing interpretability

Challenges

- Missing/Incomplete data for some seasons
- Players with high stats but low games played caused false positives
- Position-based restrictions in past All-NBA voting (prior to 2023-24)

Future Work

- Incorporate injury reports, positional competition
- Add player role/type (guard/wing/big)
- Use deep learning or ensemble stacking
- Deploy a web app version of the dashboard

Appendix

Code Snippets

- All code used to clean data, train the model, and evaluate performance is available upon request.

Data Exports

- `merged_df.csv`: All processed player data
- `top15_predictions_2024_25.csv`: Current season predictions