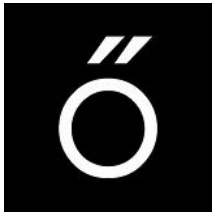
Predicting NBA Player Retention

Alex Pandya, Peter Johnson, Andrew Newman, Ryan Moruzzi, Collin Litterell



THE ERDŐS INSTITUTE

DATA SCIENCE BOOT CAMP

Overview

- The National Basketball Association (NBA) is a major professional sports league with a growing global reach and many interested parties including:
  - Basketball fans,
  - Sport bettors - casual and professional,
  - TV networks and advertisers,
  - The NBA itself, etc.
- For these stakeholders, it is important to have a framework for analyzing the future state of the NBA when making decisions such as
  - An NBA front office constructing their roster, or
  - The NBA and networks negotiating TV contracts.

Problem description

Goal: Predict if a given NBA player will play in the next NBA season based on their current season:

- On-court performance, age, and experience,
- Salary,
- Transactions.

Data collection

We gathered:

- NBA counting statistics via an NBA API library
- NBA advanced statistics via kaggle
- Player Salaries scraped from hoopshype.com
- Player transactions (waived/traded) from basketball-reference.com

In [5]:
```
#look at the data
all_data.head(3)
```

Out[5]:

	PLAYER_ID	SEASON_ID	LEAGUE_ID	TEAM_ID	TEAM_ABBREVIATION	PLAYER_AGE	GP	GS	
0	2	1983-84	00	1610612747	LAL	23.0	74	49	1
1	2	1984-85	00	1610612747	LAL	24.0	81	65	2
2	2	1985-86	00	1610612747	LAL	25.0	76	62	2

Data cleaning

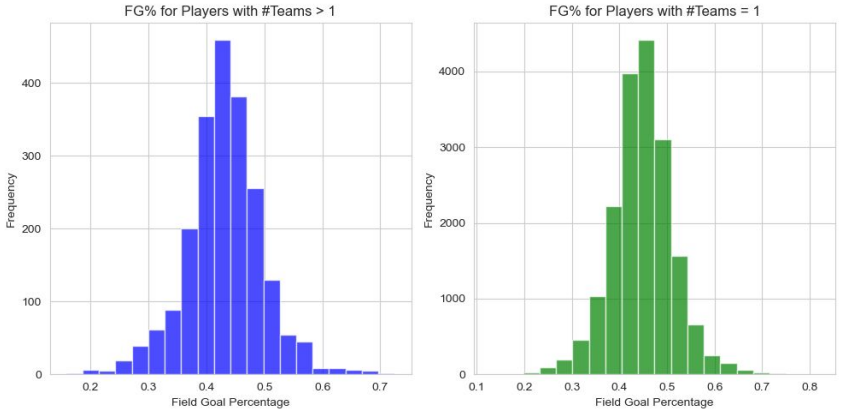
The statistic data was gathered into multiple .csv files and decisions were made for consistency of those files. For example:

- Creating a SEASON_START column, making year into a single value instead of a span of years, e.g. 2015 for 2015-2016.
- Create a team abbreviation column.
- Matching player identification numbers and names.

We then merged our .csv files into one file to analysis

- The key step to merge the data was to utilize player_id’s, and join the .csv files by aligning SEASON_START, player_id, and team abbreviations.

Feature selection/engineering



We first explored basic stats for statistically significant differences between waived and non-waived player.

In the aggregate, waived players are worse than non-waived, but on a player by player basis, prediction is difficult.

- To narrow down the features, we looked at correlation, and rescaled relevant stats on a per-minute scale.
- We also wrote a custom version of the Standard Scaler to normalize features on a per season basis to account for general player improvement over the years.

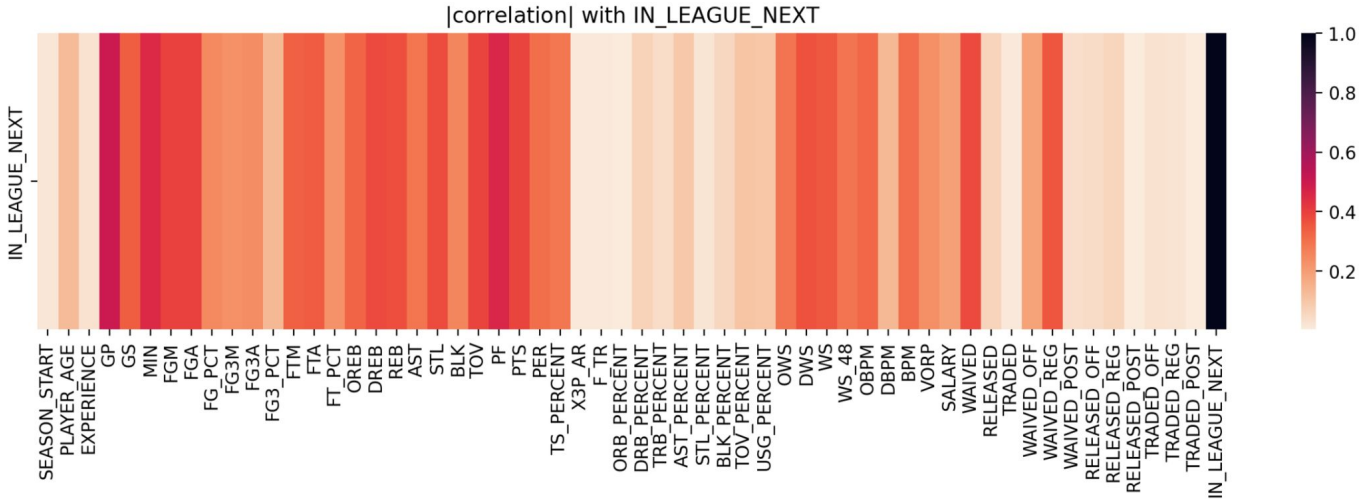
Exploratory data analysis

Metric	KNN Model	Baseline
Accuracy	0.8846	0.7952
Balanced Accuracy	0.5538	0.5000
F1 Score	0.1985	0.1158
1-Brier Score	0.9124	0.8976

- We attempted a few models to predict if a player would be waived ahead of the start of next season.
- The table shows the results of a single instance of KNN (k = 15). The baseline is random guess at the correct proportion.
- Other methods (Decision Tree, Logistic Regression, Random Forest Classifiers) produced similar results.

Model selection I: key question

- Corr(stats, IN_LEAGUE_NEXT) is reasonably large

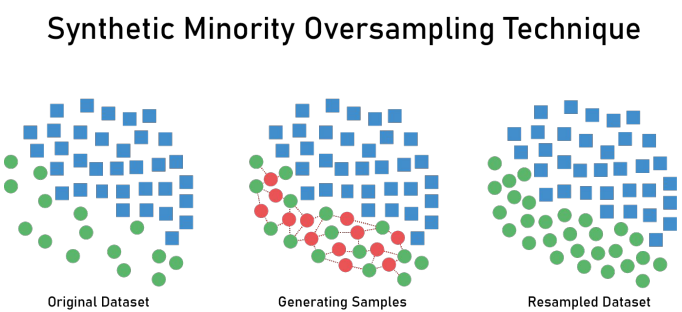


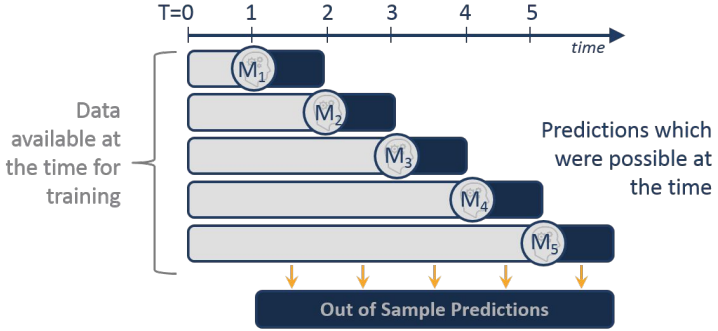
- Key question: can we predict *player retention* for the next season using current-season data?

Model selection II: approach

- Classification problem with two considerations:
  1. Imbalanced classes (~80% players retained)
  2. Time series structure (cannot use future data in predictions!)

- Solutions:
  1. Use synthetic data to balance training set (SMOTE)
  2. Time series (forward) cross-validation to avoid data leakage

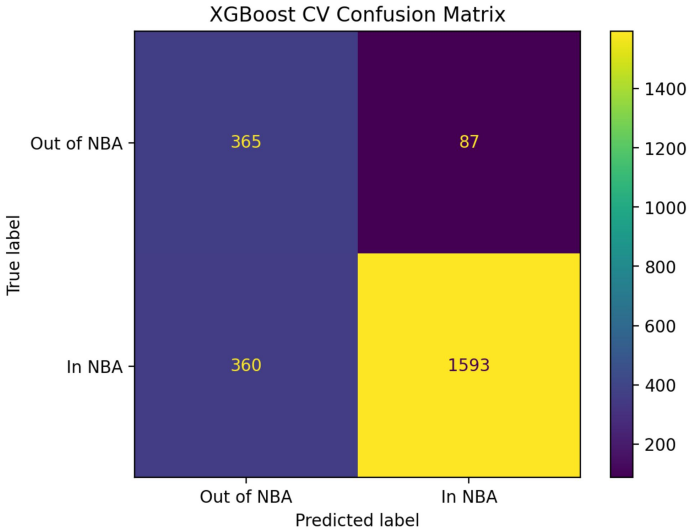




Model selection III: results

- Compared 10 models based on CV *balanced accuracy score*
- Best-performing model was **XGBoost**

Rank	Model	Balanced Accuracy	Precision	Recall (sensitivity)	NPV	Specificity	Hyperparameters
1	XGBoost Classifier	0.8116	0.9482	0.8156	0.5041	0.8076	`n_estimators=400, learning_rate=0.005`
2	Logistic Regression w/ SMOTE	0.8079	0.9566	0.7654	0.4561	0.8503	`C=0.00125`
3	Random Forest Classifier	0.8071	0.9494	0.7971	0.4824	0.8170	`n_estimators=50, max_depth=5`
4	PCA + KNN	0.8065	0.9664	0.7211	0.4253	0.8920	`pca__n_components=30, knn__n_neighbors=84`
5	PCA + QDA	0.8051	0.9553	0.7649	0.4547	0.8453	`pca__n_components=30, qda__reg_param=0.3`
6	AdaBoost Classifier	0.8024	0.9498	0.7833	0.4674	0.8214	`n_estimators=500, learning_rate=0.1`
7	Decision Tree Classifier	0.8009	0.9485	0.7868	0.4710	0.8150	`criterion='gini', max_depth=5`
8	LDA	0.8009	0.9637	0.7189	0.4208	0.8828	`shrinkage=0.8`
9	Gaussian Naive Bayes	0.7876	0.9713	0.6593	0.3839	0.9158	`var_smoothing=0.1`
10	Logistic Regression	0.7308	0.8950	0.9385	0.6637	0.5232	`C=1`



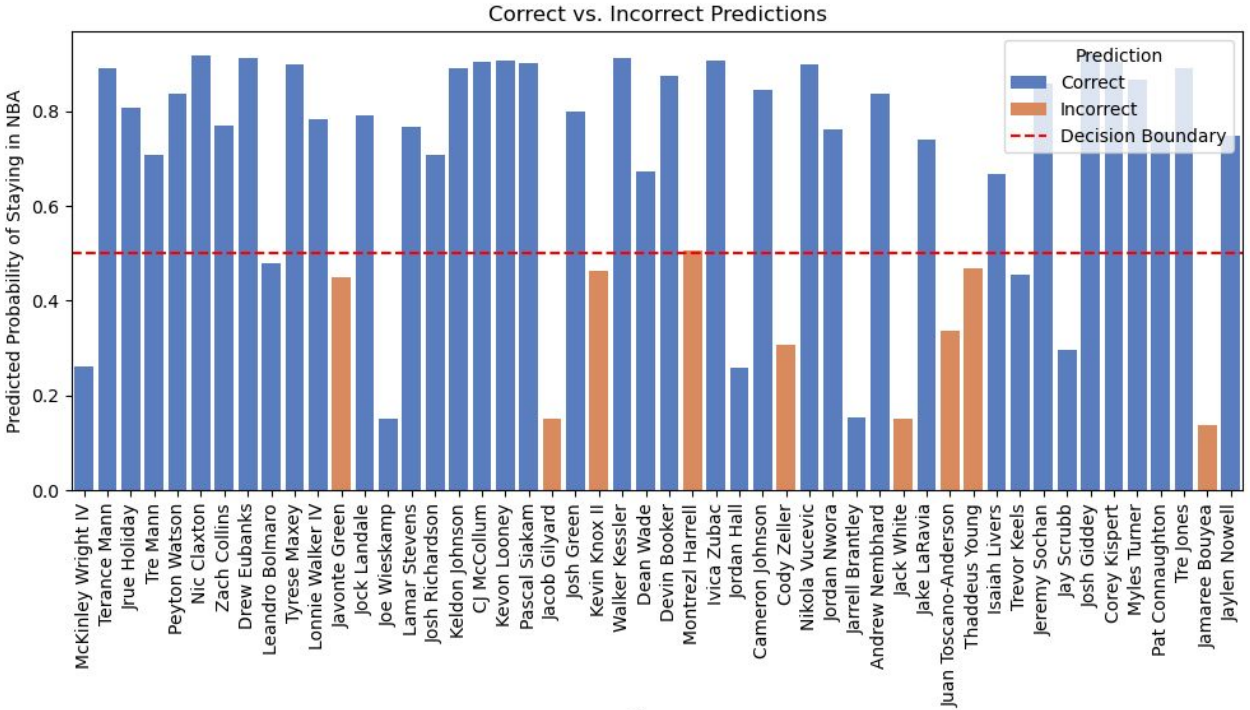
Final results

- Final model: **XGBoost Classifier with SMOTE**
- Evaluate performance on test set using walk-forward testing

Test Season	Balanced Accuracy	Precision	Recall (sensitivity)	NPV	Specificity
2017-18	0.8054	0.9405	0.7670	0.5294	0.8438
2018-19	0.8079	0.9345	0.7850	0.5567	0.8308
2019-20	0.8041	0.9570	0.7678	0.4389	0.8404
2020-21	0.7878	0.9529	0.7644	0.4078	0.8111
2021-22	0.8459	0.9375	0.8314	0.6697	0.8605
2022-23	0.7982	0.9605	0.7556	0.4022	0.8409
Avg.	**0.8082**	**0.9471**	**0.7785**	**0.5008**	**0.8379**
Baseline (always predicts 1)	0.5000	0.7876	1.0000	0.0000	0.0000

Final results

- Visualizing predictions from 2022-23 season:



Acknowledgements

We thank:

- The Erdős Institute
- Nadir Hajouji (our project mentor)
- Steven Gubkin
- Alec Clott
- Roman Holowinsky

Sources

[1] SMOTE fig: https://emilia-orellana44.medium.com/smote-2acd5dd09948

[2] Walk-forward CV fig: https://alphascientist.com/walk_forward_model_building.html