Predicting NBA MVP Using Machine Learning

Wahyunan Andika

August 2024

1. Introduction

When we talk about the NBA, we always hear about famous basketball players like Michael Jordan, Allen Iverson, Kobe Bryant, Shaquille O'Neal, LeBron James, and Stephen Curry. So what do these players have in common? Is it because they are amazing scorers or have the ability to make clutch shots at critical times? Well, those are all true, and one thing they also have in common is that they are winners of the NBA MVP award. So what's the NBA MVP? The NBA MVP is one of the most prestigious honors in the NBA. This award is given annually to the player who is considered the most valuable to their team during the regular season, based on their performance, impact on games, and overall contribution.

Predicting the MVP winner is a popular activity, especially for betting purposes, and involves analyzing various factors like player stats, team dynamics, and historical performances. This task can be quite complex, given huge amount of data and the need for accurate predictions.

In this project, I've tackled this challenge by extending the dataset from 1991 to 2024. I use dataset from the 90s era because one of the players holds the record for the highest career three-point shooting percentage, and three-point shooting is closely associated with the modern basketball era.

To enhance the prediction process, I'm using advanced Machine Learning techniques, including time series cross-validation (TSCV) and additional metrics like MAP@K. These methods not only aim to predict the MVP winner but also identify the top 5 candidates.

Furthermore, I'm experimenting with Principal Component Analysis (PCA) to see how dimensionality reduction impacts model performance. This approach may help refine the accuracy of predictions and capture evolving player trends more effectively.

2. Related Works

Predicting NBA MVP using Machine Learning by Gabriel Pastorello

Gabriel Pastorello aimed to predict the NBA MVP for the 2022 season using machine learning models. His approach involved analyzing player performance data from the NBA from 2000 to 2021.

The model that he used was Support Vector Machines, Elastic Net, Extreme Gradient Boosting, Random Forest, Adaboost, Gradient Boosting, Light Gradient Boosting Machine, with evaluation metrics RMSE and R^2.

His result were:

- SVM: RMSE = 0.087, R^2 = 0.867
- Elastic Net: RMSE = 0.153, R^2 = 0.585
- Random Forest: RMSE = 0.096, R^2 = 0.837
- AdaBoost: RMSE = 0.119, R^2 = 0.752
- Gradient Boosting: RMSE = 0.108, R^2 = 0.794
- LGBM**:** RMSE = 0.107, R^2 = 0.797

His strength for his works was:

- Diverse Model Comparison (he knew that the dataset was high dimensional, so he used complex model for diversity and comparison).
- Using both RMSE and R^2 allows for a thorough evaluation of model performance. RMSE measures prediction accuracy, while R^2 provides insight into the variance explained by the model.
- Wide Dataset (He used a dataset where physicality is less emphasized compared to the 90s season, and the 2000s are characterized by the beginning era of more flexible players.)
- High Performance

His Weakness for his works was:

- Model Complexity (Some Model like XGB and LGBM, are complex and require high computation resource and times.)
- Evaluation Metrics (RMSE and R^2 are useful but may not fully capture the nuances of MVP prediction. Including additional metrics like MAP@K could provide more insights into model performance.)
- High Dimensionality Dataset

Opinion:

In this project, I've expanded the dataset to cover the years 1991 to 2024. I chose Steve Kerr as a benchmark because his record for the highest career three-point shooting percentage marks a significant milestone in the evolution of three-point shooting, representing the beginning of the modern NBA era. I'm also using time series cross-validation (TSCV) and additional metrics like MAP@K to not only predict the MVP winner but also to identify the top 5 candidates. Additionally, I'm experimenting with PCA to explore how dimensionality reduction affects model performance.

3. Dataset & Features

The dataset is sourced from web scraping Basketball Reference (https://basketball-reference.com/). It consists of four major components:

- Player Statistics per Season (1991-2024): Detailed stats for individual players each season.
- Player Advanced Statistics per Season (1991-2024): Advanced metrics that provide deeper insights into player performance.
- Team Performance per Season (1991-2024): Data on how teams performed each season.
- MVP Candidates per Season (1991-2024): Information about players considered for MVP each season.

Preprocessing begins after exporting the data from web scraping to CSV files. Missing values are addressed, particularly in shooting percentages where some players didn't attempt certain shots (like free throws, three-pointers, or field goals). Player names are cleaned by removing any asterisks that indicate MVP winners. For the position column, where players may have multiple positions listed, it is standardized to reflect their primary role or the position they played most during that season. In the GB column, which shows the discrepancy between the highest win and the selected team's win, '-' is replaced with 0 to simplify the data.

For the train-test split, data from 1991-2023 is used for training and 2024 for testing, with time series cross-validation (TSCV) applied using 5 splits due to the temporal nature of the data. Data normalization is performed as part of the preprocessing pipeline once all datasets are merged.

The dataset includes a comprehensive set of features related to player and team performance, with a total of 61 features:

- Player Information: Includes player name ('Player'), position ('Pos'), age ('Age'), and team ('Tm').
- Game Statistics: Covers games played ('G'), games started ('GS'), minutes per game ('MP'), field goals made ('FG'), field goals attempted ('FGA'), and shooting percentages ('FG%', '3P%', '2P%', 'eFG%', 'FT%').
- Shooting Stats: Details three-point shooting ('3P', '3PA', '3P%'), two-point shooting ('2P', '2PA', '2P%'), and free throws ('FT', 'FTA').
- Rebounding and Defense: Includes offensive rebounds ('ORB'), defensive rebounds ('DRB'), total rebounds ('TRB'), steals ('STL'), blocks ('BLK'), turnovers ('TOV'), and personal fouls ('PF').
- Scoring and Efficiency: Metrics such as points scored ('PTS'), player efficiency rating ('PER'), true shooting percentage ('TS%'), and various advanced stats including usage rate ('USG%') and win shares ('WS', 'WS/48').
- Advanced Metrics: Features like offensive box plus-minus ('OBPM'), defensive box plus-minus ('DBPM'), box plus-minus ('BPM'), and value over replacement player ('VORP').
- Team Performance: Team-specific data such as team wins ('W'), losses ('L'), win/loss percentage ('W/L%'), games behind ('GB'), points scored per game ('PS/G'), points allowed per game ('PA/G'), and simple rating system ('SRS').

These features provide a thorough view of player performance, team dynamics, and advanced metrics over the seasons from 1991 to 2024.

4. Methods

- XGBoost

XGBoost is an advanced implementation of gradient boosting, known for its speed and performance. It builds an ensemble of weak prediction models, typically decision trees, in a sequential manner.

$$F(x) = \Sigma_{t=1}^{m} y_i h_i(x)$$

$F(x)$ = Represents the final prediction for a given input x.

$\Sigma_{t=1}^{m} y_i h_i(x)$ = This sum represents the ensemble of weak learners (decision trees).

$h_i(x)$ = the i^{th} weak learner (a decision tree in XGBoost).

y_i = The weight assigned to the predictions of the i^{th} tree (in XGBoost, the trees are added sequentially, and each learner tries to correct the errors of the previous ones)

XGBoost starts with an initial simple model and iteratively adds new models to correct the errors of the existing ensemble. Each new tree focuses on the residuals of the previous prediction. It uses gradient descent to minimize a regularized objective function, which combines a convex loss function and a penalty term for model complexity. This approach helps in reducing overfitting. XGBoost also employs various optimizations like parallel processing and cache-aware access for improved speed.

- Gradient Boosting

Gradient Boosting is a machine learning technique for regression and classification problems. It produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees.

$$F_m(x) = F_{m-1}(x) + \eta \cdot h_m(x)$$

$F_m(x)$ = The prediction made by the model after adding the m^{th} weak learner (often a decision tree).

$F_{m-1}(x)$ = The prediction from the ensemble model at the previous step (after adding m−1m-1m−1 weak learners).

η= The **learning rate**, which controls how much the new weak learner contributes to the overall model. A lower learning rate reduces the impact of each learner, making the model build more conservatively.

$h_m(x)$ = the m^{th} weak learner, which is trained to predict the **residuals** (errors) from the previous model.

The algorithm starts by fitting a simple model (e.g., a decision tree) to the data. It then calculates the residuals (the differences between predicted and actual values) and fits a new model to these residuals. This process is repeated, with each new model focusing on the errors

of the previous ensemble. The learning rate η controls how much each new model contributes to the overall prediction. This iterative process continues until a specified number of models have been added or until the model's performance stops improving.

- Ridge Regression

Ridge Regression is a regularized version of linear regression, designed to mitigate the problem of multicollinearity in regression.

$$||y - X\beta||^2 + \alpha||\beta||^2$$

y = the vector of target values.

X = the matrix of predictor variables (features).

β = the vector of regression coefficients (parameters).

α = the regularization parameter (also denoted as λ in some texts).

$||y - X\beta||^2$ = the residual sum of squares (RSS) or the mean squared error (MSE) between the observed values y and the predicted values $X\beta$.

Ridge regression adds a penalty term to the ordinary least squares objective function. This penalty is proportional to the sum of the squared coefficients (L2 norm). The α parameter controls the strength of this penalty. As α increases, the model becomes more regularized, shrinking the coefficients towards zero (but not exactly to zero). This helps prevent overfitting by reducing the model's sensitivity to individual features. Ridge regression is particularly useful when there are many features with similar predictive power, as it tends to distribute the importance across these features rather than arbitrarily selecting one.

- Lasso Regression

Lasso (Least Absolute Shrinkage and Selection Operator) is another regularized version of linear regression that performs both regularization and feature selection.

$$||y - X\beta||^2 + \alpha||\beta||_1$$

y = the vector of target values.

X = the matrix of predictor variables (features).

β = the vector of regression coefficients (parameters).

α = the regularization parameter (also denoted as λ in some texts).

$||y - X\beta||^2$ = the residual sum of squares (RSS) or the mean squared error (MSE) between the observed values y and the predicted values $X\beta$.

Similar to Ridge regression, Lasso adds a penalty term to the ordinary least squares objective. However, Lasso uses the L1 norm of the coefficients instead of the L2 norm. This difference leads to a crucial property: Lasso can force some coefficients to be exactly zero, effectively performing feature selection. As α increases, more coefficients are set to zero, resulting in a sparser model. This makes Lasso particularly useful when you believe only a few features are

relevant. The optimization problem in Lasso is solved using techniques like coordinate descent or the least angle regression algorithm.

- Elastic Net

Elastic Net is a regularized regression method that linearly combines the L1 and L2 penalties of Lasso and Ridge regression.

$$||y - X\beta||^2 + \alpha||\beta||_1 + \alpha||\beta||^2$$

y = the vector of target values.

X = the matrix of predictor variables (features).

β = the vector of regression coefficients (parameters).

α = the regularization parameter (also denoted as λ in some texts).

$||y - X\beta||^2$ = the residual sum of squares (RSS) or the mean squared error (MSE) between the observed values y and the predicted values $X\beta$.

Elastic Net addresses some limitations of Lasso, particularly in scenarios where the number of predictors is much larger than the number of observations or when there are groups of correlated predictors. It combines the penalties of Ridge and Lasso regression, allowing for both shrinkage (like Ridge) and automatic variable selection (like Lasso). The mix of L1 and L2 penalties is controlled by a mixing parameter. When this parameter is 1, Elastic Net becomes Lasso; when it's 0, it becomes Ridge. Values between 0 and 1 blend the two approaches, potentially outperforming either method alone.

- Principal Component Analysis (PCA)

PCA is a dimensionality reduction technique used to transform a high-dimensional dataset into a lower-dimensional space while preserving as much variability as possible.

$$X = U\Sigma V^T$$

X = an $m \times n$ matrix (e.g., a dataset with m samples and n features).

U = an $m \times m$ orthogonal matrix (the matrix of left singular vectors).

Σ = an $m \times n$ diagonal matrix with non-negative real numbers on the diagonal (the singular values).

V^T = an $n \times n$ orthogonal matrix (the matrix of right singular vectors, transposed).

This equation is to find the principal component.

PCA works by identifying the principal components of the data, which are the directions of maximum variance. It starts by computing the covariance matrix of the centered data. Then, it calculates the eigenvectors and eigenvalues of this matrix. The eigenvectors represent the directions of the principal components, while the eigenvalues represent the amount of variance explained by each component. PCA sorts these components by decreasing eigenvalue and selects the top k components to create a lower-dimensional representation.

This process effectively rotates the data to align with these principal components, potentially revealing underlying structures in the data.

- Random Forest (RF)

Random Forest is an ensemble learning method that operates by constructing multiple decision trees during training and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

$$f(x) = \frac{1}{B} \Sigma_{b=1}^{N} f_b(x)$$

$f(x)$ = The final prediction function of the Random Forest model for a given input x.

$\frac{1}{B}$ = The normalization factor to average the predictions.

$\Sigma_{b=1}^{N}$ = Summation operator that aggregates the predictions from all BBB decision trees.

$f_b(x)$ = The prediction made by the bbb-th decision tree in the Random Forest for the input x.

Random Forest builds multiple decision trees on bootstrapped samples of the original dataset (a technique known as bagging). Each tree is grown using a random subset of features at each split, which introduces additional randomness and helps to decorrelate the trees. This randomness helps to make the model more robust and less prone to overfitting. When making predictions, each tree in the forest makes its own prediction, and the final prediction is the average (for regression) or the majority vote (for classification) of all trees. Random Forests often perform well out-of-the-box and are less sensitive to outliers in the data.

- Adaboost

AdaBoost, short for Adaptive Boosting, is an ensemble learning method that combines multiple weak learners to create a strong learner.

$$F(x) = \Sigma_{t=1}^{T} \alpha_t h_t(x)$$

$F(x)$ = The final prediction function of the AdaBoost model for a given input x.

$\Sigma_{t=1}^{T}$ = Summation operator that aggregates the predictions from all TTT weak learners.

α_t = The weight assigned to the t_{th} weak learner $h_t(x)$).

$h_t(x)$) = The prediction made by the t_{th} weak learner for the input x.

AdaBoost works by iteratively training weak learners (often shallow decision trees) on the dataset. After each iteration, it increases the weights of incorrectly classified instances, forcing subsequent weak learners to focus more on these difficult cases. The algorithm assigns a weight α_t to each weak learner based on its accuracy. The final model is a weighted sum of all weak learners, where more accurate learners have higher weights. This process continues for a specified number of iterations or until the model's performance stops improving.

AdaBoost is particularly effective at reducing bias and can achieve high accuracy, but it may be sensitive to noisy data and outliers.

5. Experiments/Results/Discussion

- Experiments

In this project, I use RMSE and MAP@k as evaluation metrics for the MVP candidate prediction model due to their complementary strengths. Since RMSE effectively measures the accuracy of player performance predictions, which is very crucial for indentify the winner of NBA MVP. Since my goal for this project is predicting the top 5 candidates (not only the winners), I used MAP@K, specifically tailored for top-K predictions, assesses the model's ability to correctly rank and identify the most likely MVP candidates within the top 5 selections. With combining these together, these metrics provide a comprehensive evaluation of both the model's predictive accuracy and its ability to identify and rank the most promising MVP candidates.

For experimental purpose, I use PCA to reduce the dimensionality of the data, and check if the PCA also make a better model. However, the final results showed that models trained without PCA, such as Random Forest and XGBoost, achieved better performance on the usual test (without PCA) set, it indicates that original feature set had enough valuable information, and PCA did not offer significant advantages for this particular dataset. Consequently, the use of PCA was deemed unnecessary in this case, as it did not improve model accuracy or ranking performance.

Also, I use time-series cross-validation since the data had a temporal structure, that mean the model depends the order of the data over time (per year). Its crucial in time-sensitive prediction NBA MVP candidate rankings, where events are sequential and shuffling the data would introduce the data leakage. For the cross-validation, I use 5-fold TSCV, the are no particular reason for choosing this value since it was built in the TSCV that provides a good balance between training time and model evaluation consistency.

- Results

The models were evaluated using Mean Average Precision at K (MAP@K) and Root Mean Square Error (RMSE) across training, validation, and test sets. Initially, the performance of models with PCA looked promising based on training and validation scores from the grid search cross-validation. However, when evaluated on the test set using RMSE and MAP@K, PCA did not offer the expected improvements. Below is a summary of the results:

Model	Train Score GridSearchCV	Valid Score GridSearchCV
Random Forest with PCA	0.018772	0.033112
Adaboost DT with PCA	0.019940	0.033859
Gradient Boosting with PCA	0.017942	0.034033
XGB with PCA	0.017780	0.035193
Random Forest	0.015922	0.035439
AdaBoost DT	0.016231	0.036406
Gradient Boosting	0.010163	0.037198
XGB	0.015148	0.037417

Ridge	0.048889	0.050544
ElasticNet	0.048938	0.050621
Lasso	0.048832	0.050672
Ridge with PCA	0.051364	0.051906
ElasticNet with PCA	0.051372	0.051907
Lasso with PCA	0.051352	0.051916
Baseline with PCA	0.058868	0.058671
Baseline	0.058868	0.058671

Model	MAP@K	RMSE
XGB	0.008042	0.018860
Random Forest	0.007692	0.018517
XGB with PCA	0.006993	0.023138
Random Forest with PCA	0.006993	0.030149
Adaboost DT with PCA	0.006993	0.031851
Lasso	0.006643	0.045063
Adaboost DT	0.006294	0.023484
Gradient Boosting	0.006294	0.029389
Ridge	0.006294	0.044986
ElasticNet	0.006294	0.045019
Lasso with PCA	0.005944	0.047506
Ridge with PCA	0.005944	0.047511
ElasticNet with PCA	0.005944	0.047516
Baseline	0.000000	0.054093
Baseline with PCA	0.000000	0.054093

During the initial grid search cross-validation, models with PCA showed relatively strong validation scores, giving the impression that PCA might help reduce overfitting and improve generalization. For example:

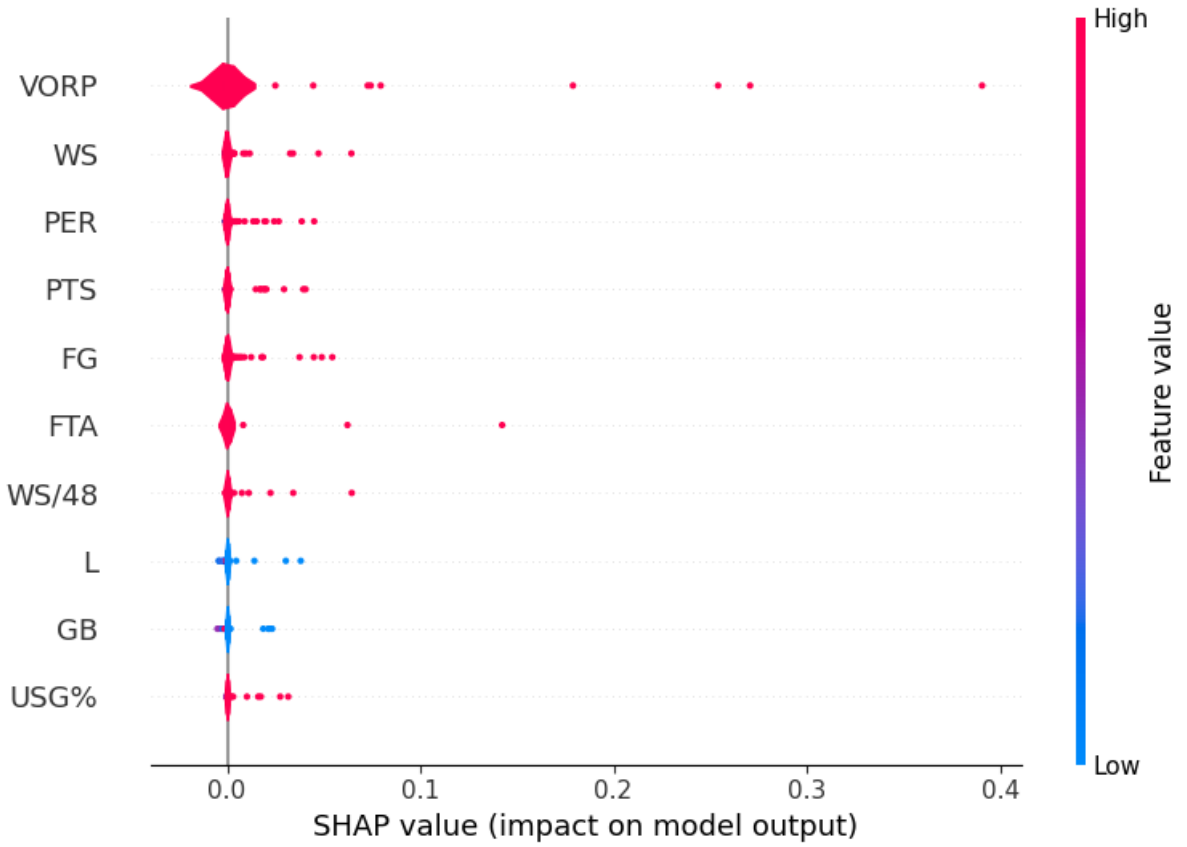
- Random Forest with PCA had a validation score of 0.033112, which was better than its non-PCA counterpart's validation score of 0.035439.
- Similarly, XGB with PCA had a slightly better validation score than the non-PCA model (0.035193 vs. 0.037417).

However, these improvements did not translate to the test set, where PCA versions of the models underperformed compared to their non-PCA counterparts. For instance:

- XGB (without PCA) achieved a MAP@K of 0.008042 and an RMSE of 0.018860 on the test set, while XGB with PCA had a MAP@K of 0.006993 and a worse RMSE of 0.023138.
- Similarly, Random Forest (without PCA) had a slightly better MAP@K of 0.007692 and RMSE of 0.018517 compared to Random Forest with PCA's test scores of 0.006993 and 0.030149, respectively.

This shows that while PCA helped reduce the dimensionality of the feature space, it did not necessarily lead to better generalization for this particular task. The test set performance metrics (RMSE and MAP@K) suggest that the models without PCA were better at capturing the necessary complexity for predicting NBA MVPs.

To further understand feature importance, SHAP values were used to analyze the **XGB** model, which performed best on the test set:



A. Top Features

- VORP (Value Over Replacement Player) had the highest impact on the model, confirming that individual player contribution to team success is a key driver of MVP predictions.
- Win Shares (WS) and Player Efficiency Rating (PER) also contributed significantly to the model's decisions, showing that performance metrics focused on player efficiency and contribution were crucial.

B. Lower Impact and Negative Values

- Losses (L) and Games Behind (GB) had a negative impact, which makes sense as MVPs are less likely to come from teams that are performing poorly.

Player	Share	Year	Predictions	Rank Actual	Rank Prediction
Nikola Jokić	0.935	2024	0.805677	1	1
Shai Gilgeous-Alexander	0.646	2024	0.729995	2	2
Luka Dončić	0.572	2024	0.453909	3	4
Giannis Antetokounmpo	0.194	2024	0.579004	4	3
Jalen Brunson	0.143	2024	0.124736	5	5
Jayson Tatum	0.087	2024	0.089731	6	6
Anthony Edwards	0.018	2024	0.056299	7	8

From the result (XGB Model without PCA), we can see that :

a. Accuracy: The XGBoost model accurately predicted the top two MVP candidates (Nikola Jokić and Shai Gilgeous-Alexander) and their rankings.
b. Top 5 Prediction: The model correctly identified 3 out of the top 5 MVP candidates, with only a slight discrepancy in the order of the 3rd and 4th positions.
c. Ranking Precision: For 4 out of 7 players, the model's predicted rank matched the actual rank exactly.
d. Share vs. Predictions: The model's predictions generally align well with the actual share values, especially for the top candidates.
e. Minor Discrepancies: The model slightly overestimated Giannis Antetokounmpo's ranking (predicted 3rd, actually 4th) and underestimated Luka Dončić's ranking (predicted 4th, actually 3rd).
f. Lower Ranks: The model's performance is slightly less accurate for lower-ranked candidates, as seen with Anthony Edwards (predicted 8th, actually 7th).

While PCA seemed promising based on the grid search validation scores, the test set results contradict this initial assumption. The non-PCA models, especially XGB and Random Forest, consistently outperformed their PCA counterparts in both MAP@K and RMSE, showing better generalization. The SHAP analysis reinforced that individual player metrics (VORP, WS, and PER) are the most predictive features for MVP status, while team-based metrics (L, GB) play a lesser role.

- **Discussions**

In this project, I use different models to predict the NBA MVP, and while PCA was expected to help, the results showed that it didn’t provide the improvements that hoped for. At first, during grid search cross-validation, the PCA models showed better validation scores, but when tested on unseen data, the non-PCA models like XGB and Random Forest performed better. This was clear when comparing metrics like RMSE and MAP@K as key indicators of model accuracy.

The XGB model did a very good job predicting the top MVP candidates. For example, it predicted Nikola Jokić as the top candidate with a high predicted share value, closely matching his actual dominance in the MVP race. Other players, such as Shai Gilgeous-

Alexander and Luka Dončić, were also ranked highly, showing the model's ability to capture top-tier performance.

However, there were some mismatches in the lower rankings. For example, Kevin Durant was predicted to rank 9th but his actual performance placed him 14th. This shows that while the model excels at predicting the top candidates, its accuracy for mid-tier or lower-ranked players diminishes. This could be due to smaller statistical differences between those players or noise in the data. To improve the accuracy for lower-ranked players, one of the approaches I will take is adding more features that reflect the value of the discrepancy between the current data and the last season data, adding more insight to the model.

In summary, while the core models like XGB and Random Forest didn't show major signs of overfitting, some of the weaker-performing models (like PCA and Adaboost) might have been overfitting slightly. The general strategy of using cross-validation, tuning hyperparameters, and testing on unseen data helped control overfitting to a large extent, but further improvements could be made to ensure more consistent performance across models, particularly in handling mid-tier or lower-ranked players.

6. Conclusion/Future Work

- Conclusion

This project for predicting NBA MVP candidates give insightful results, with XGBoost (XGB) and Random Forest models demonstrating superior performance, their success can be attributed to their ensemble nature, which allows them to capture complex relationships within the data while minimizing overfitting. These ensemble methods proved particularly effective in handling the complex nature of MVP selection.

While Principal Component Analysis (PCA) showed promise during grid search validation, the test set results revealed a different story. Non-PCA models, particularly XGB and Random Forest, consistently outperformed their PCA counterparts in both MAP@K and RMSE metrics, demonstrating better generalization capabilities.

SHAP (SHapley Additive exPlanations) analysis provided crucial insights into feature importance. Individual player metrics such as VORP (Value Over Replacement Player), WS (Win Shares), and PER (Player Efficiency Rating) emerged as the most predictive features for MVP status. Team-based metrics like Losses (L) and Games Behind (GB) played a comparatively lesser role in predictions.

The most significant hurdle i encountered was in the data preparation phase. Cleaning and standardizing web-scraped data required meticulous attention to ensure consistency across different sources before merging datasets.

- Future Work

For future work, the next goal is to deploy the predictive models on a portfolio website to offer real-time MVP predictions throughout the NBA season. I've also considering implementing self-training models like River Forest or advanced XGBoost variants to minimize manual oversight and continuously improve the predictions as new data becomes

available. To enhance the accuracy and timeliness of the forecasts, I plan to develop an automated pipeline for data collection and processing. Additionally, i'll delve into advanced player statistics revealed by SHAP analysis to further boost our model's predictive capabilities. Lastly, i aim to design an intuitive dashboard that will visualize predictions and feature importance, making our insights more accessible to a broader audience.