NBA Line-Up Prediction Report

Faculty of Engineering and Applied Science

SOFE 4620U: Machine Learning & Data Mining | CRN: 74292 | Section: 001

Due: March 19th, 2025

Group #18

Rodney Stanislaus (100822918)

Ontario Tech University

Oshawa, Ontario

rodney.stanislaus@ontariotechu.net

Calvin Reveredo (100825740)

Ontario Tech University

Oshawa, Ontario

calvin.reveredo@ontariotechu.net

Naftanan Mamo (100822222)

Ontario Tech University

Oshawa, Ontario

naftanan.mamo@ontariotechu.net

Table of Contents

Purpose

Machine learning serves as a powerful tool for prediction, and can lead to the development of strong and effective models. These models are capable of building knowledge from prior data received and utilizing this foundation for future decision-making. When provided with new data, the model will decide how the data should be classified, with the possibilities being the defined labels associated with the model. The output of the model with this classification can be used in a variety of applications that would not be possible or as simple without machine learning.

In this project, machine learning will be used in the development of a model, that is capable of predicting the optimal 5th player for a home team's current NBA line-up in order to maintain a favourable outcome against the away team's current line-up. This model is developed using data records of previous NBA games containing several associated features. Examples of provided test data would not contain a 5th home team player, and it is the task of the model to predict the optimal 5th player using its prior knowledge from training data and the logic that dictates how to determine the potential candidates for the answer.

Methodology

To accomplish the development of a model for predicting the optimal 5th player in the home team line-up, an extensive process was created with all the necessary steps to conduct. This included:

- **Data Collection and Preparation**: The dataset available consists of NBA game lineups from 2007 to 2015. In order to prepare the data for use, preprocessing was pivotal in order to ensure consistency and quality. Preprocessing involved cleaning the data, removing entries (rows with -1 outcomes), and combining multiple CSV files into a single dataset that seamlessly merges together.

- **Feature Engineering**: The provided raw data contained significant features that needed to be extracted, in order to create a model who's predictions possess the highest level of accuracy. The features of primary importance and relevance include categorical features (player and team names) and numerical features (season, starting minute). After being extracted, they were encoded and transformed to a usable format for model training.

- **Model Development**: A Random Forest Classifier was used as the foundation for the model due to the countless advantages it provides, including robustness, interpretability, and its ability to handle mixed data types. The model was trained on a subset of the data, and validated on a separate test set to verify the model's predictive level in a variety of scenarios.

- **Evaluation and Analysis**: The success of the project is dependent not only on the creation of a functioning model, but a model of high accuracy in its results. Accuracy was the primary metric used in evaluating the model's performance, where predictions were compared against the true values, and a detailed analysis was conducted to identify trends, strengths, and limitations.

The outlined methodology ensured that all aspects of the task were thoroughly considered and addressed, leading to a complete understanding of what needed to be done and the necessary steps to take. As a result, a reliable and effective machine learning model was able to be created.

Feature Engineering

The provided data records of previous NBA games consist of several attributes. For the developed model, restrictions were placed on which of these features could and could not be used in the training and subsequent testing of the model. The permitted features were the following:

- Game
- Season
- Home_Team
- Away_Team
- Starting_Min
- Home_0
- Home_1
- Home_2
- Home_3
- Home_4
- Away_0
- Away_1
- Away_2
- Away_3
- Away_4

Based on the provided data for testing, the “Game” feature of the data records was not included, and therefore does not serve a use in our model. All other features that were provided had data associated with them for each record.

- “Season” is important for indicating what year that the provided game took place, in order to gain context of the time period.
- “Home_Team” and “Away_Team” give indication to the teams we are working with.
- The “Home_#” and “Away_#” players show the current line-ups for each team, which is the primary component needed for determining an optimal 5th player for the home team.

Selected Model

In order to accomplish the outlined task, a Random Forest model was selected to be used. It is used for prediction by taking numerous sub-samples of a provided dataset, and invoking a specified number of decision trees onto them [1]. These trees give classification results of the data, and a majority vote between trees is taken in order to ensure optimal accuracy in what the final decision is [2]. Doing so helps to mitigate potential mistakes that could have been made, by considering multiple different sources as opposed to just one.

The sampling is conducted through the bootstrap sampling technique, where every tree has its own training set generated for it [2]. The trees then progress using randomly selected features, which is used in the process of their splitting [2]. Decisions of each tree would then be retrieved once they have reached their finalized state, which gives way for the voting process to begin. The process and operation of Random Forest aligns with what is needed for our task, as decisions will need to be made based on predefined features of what the optimal 5th player will be for provided line-ups.

In addition to the functionality of Random Forest helping with the success of the task, there are also many associated benefits with this particular model that increase its desirability, including: managing complexity, overfitting mitigation, feature importance, parallel functionality, and scalability [3].

- Manage Complexity: Relationships that Random Forest deal with do not have to be linear, which is important in this case where aspects such as player performance and behaviour of the team is able to be understood by Random Forest.
- Overfitting Mitigation: The use of many trees avoids issues related to overfitting compared to if only 1 tree were generated, thus eliminating this concern and providing results of excellent accuracy [3].
- Feature Importance: Each feature is analyzed thoroughly by Random Forest, which helps to highlight which are most pivotal to the predicted result and enhances the model's overall process of arriving at a decision [3].
- Parallel Functionality: The creation of the trees is executed in parallel, meaning there is no impact on run time, while still gaining the advantage of improved accuracy from multiple trees.
- Scalability: A large amount of data is able to be managed and used with this model, meaning the NBA data records will pose no issue [3].

Training Process

A pivotal step in the training process was the creation of training samples using the *generate_training_samples* function. This function was designed to simulate the process of making a prediction, by masking one player in each home team lineup and treating the masked player as the targeted optimal player. For each of the five positions in the home team lineup (home_0 to home_4), a copy of the dataset was created. The player in the current position (home_i) was masked by replacing their name with a placeholder ("?"), while the original player's name was stored in a new column, *target_player*, representing the label that the model must predict. This process was repeated for all five positions, resulting in five datasets that were slightly altered from one another. To be used with the model, all the datasets were then concatenated into a single training dataset. The process of masking each position iteratively enabled the function to effectively increase the size of the training dataset, ensuring that the model successfully learned to predict any player in the lineup, regardless of their position.

The dataset was divided into training and testing sets using an 80-20 split. This was done to permit training of the model on a large portion of the data, while maintaining the remaining portion as a separate subset for validation purposes. An approach of this nature ensured that the

model's performance could be evaluated on data that was not yet seen by the model, thus providing a strong indication of its level of prediction.

Categorical features, such as player and team names, were encoded into numerical format using *LabelEncoder* from the *sklearn.preprocessing* module, to ensure compatibility with the Random Forest model and its associated algorithm. The encoded dataset was then used to initialize and train a RandomForestClassifier. The model was configured with 50 decision trees (n_estimators=50) and a maximum depth of 25 (max_depth=25) to maintain an appropriate balance with complexity and computational efficiency. The random_state parameter was set to 42 to ensure reproducibility of results. During training, numerous decision trees were created by the algorithm, and each tree was trained on a random subset of the data and features. The utilization of this approach leads to a desirable decrease in overfitting and improvement in generalization.

Pre and Post Processing

Pre-Processing
The pre-processing stage involved a series of steps designed to clean and refine the dataset for effective model training. A Python script was developed to automate the loading, cleaning, and merging of multiple CSV files from the NBA-CSV directory. The script leveraged pandas for efficient data manipulation and logging to track its progress and handle potential errors.

During the cleaning process, irrelevant columns related to shot attempts, points, and percentages for both home and visitor teams were removed to reduce noise in the dataset. Additionally, rows with an 'outcome' value of -1 were excluded to ensure only valid game results were retained. The cleaned data from all CSV files was then combined into a consolidated dataset named matchups_combined_cleaned.csv. This refined dataset provided a structured and reliable foundation for training the prediction model.

Post-Processing
To enhance prediction accuracy and improve model evaluation, a dedicated post-processing step was implemented to identify new players appearing in the **2016** test data. This step ensured that players absent from the training dataset were accounted for, minimizing prediction errors.

Steps Involved:

1. Extracting Unique Players from Training Data:
    - The training dataset (`matchups_combined_cleaned.csv`) was analyzed to compile a list of all unique player names. This created a reference set of known players.
2. Loading and Processing Test Data:
    - The final 100 rows of the test data (`NBA_test.csv`) were isolated for analysis, corresponding to the 2016 season.
3. Identifying New Players:
    - The extracted player list from the test data was compared against the reference list from the training data. Any players present in the test data but absent in the training dataset were flagged as new players.

4. Analyzing Rows with New Players:
    - Rows in the test data containing these newly identified players were highlighted. This allowed for a clear understanding of where unexpected entries appeared, aiding in refining the prediction model's behavior.

This post-processing step helped identify gaps in the model's training data, enuring that unrecognized player data could be addressed separately. By isolating these instances, the prediction model could be adjusted or tuned to improve accuracy and account for newly introduced players.

Evaluation

To evaluate our training model, we experimented with different initialization parameters for the Random Forest Classifier.

n_estimators: The number of decision trees in the forest.

Increasing the number of trees to 100 resulted in a marginal improvement in accuracy but significantly increased the computational cost. Given the trade-off between performance and resource usage, we opted to use 50 trees for the final model.

max_depth=25: The maximum depth of each tree.

A max_depth of 25 was decided based on the performance of the model after testing it with different values. The below describes that testing:

unset
    training 80-20 split accuracy: 72.93%
    test data accuracy: 88.6%
    2016 33/100 - 33%
10
    training 80-20 split accuracy: 36.71%
    test data accuracy: 35.6%
    2016 17/100 - 17%
15
    training 80-20 split accuracy: 58.48%
    test data accuracy: 61.9%
    2016 27/100 - 27%
20
    training 80-20 split accuracy: 68.78%
    test data accuracy: 82.6%
    2016 32/100 - 32% (setting random_state to 43 produced an accuracy of 28%)
25
    training 80-20 split accuracy: 72.02%
    test data accuracy: 87.4%
    2016 29/100 - 29%
30

training 80-20 split accuracy: 72.87%
test data accuracy: 87.8%
2016 29/100 - 29%

It is noted that as the max_depth increases, the model begins to overfit the training data, resulting in minimal improvements in test accuracy. Deeper trees increase the training time and resource requirements, making the model less efficient without substantial gains in performance. Also, the model's accuracy on the 2016 test data remained relatively stable across different max_depth values, highlighting the challenges posed by new players not present in the training dataset.

random_state=42: Ensured reproducibility of results.

Performance

The machine learning model achieved an overall accuracy of 87.40% (874/1000) on the provided test dataset spanning from 2007 to 2016. Below is the breakdown of correct predictions by year:

Year	Correct Predictions	Accuracy
2007	93/100	93.00%
2008	92/100	92.00%
2009	92/100	92.00%
2010	96/100	96.00%
2011	96/100	96.00%
2012	94/100	94.00%
2013	98/100	98.00%
2014	91/100	91.00%
2015	93/100	93.00%
2016	29/100	29.00%

In the 2016 test data (last 100 rows), there are 41 new players that were not apart of the combined cleaned dataset that was used for training.

The number of rows that contain a player not in the training set is 67. Of these rows, 29 of them have two or more players not in the training set, while 8 rows have three or more players not in the training set, and one row has 4 players not in the training set.

This significantly lowers the odds of our model correctly choosing the optimal player, in some cases, the chance is zero as the correct label was one of the new players, which can't be selected with our trained model (6 correct labels are new players that are not present in our dataset).

If 29 rows were taken out (the ones which contain 2 or more new players) and the 6 rows with impossible predictions (there are 2 rows that overlap so only 33 rows are taken out), our 2016 prediction accuracy is 23/67 - 34.32% (removed 6 correct predictions as well which overlapped), which is a bit better than 29% rate.

If 67 rows were taken out that contain a player that is not in the data set (all 6 rows with impossible predictions overlap), our 2016 prediction accuracy is 13/33 - 39.39% (removed 16 correct predictions as well which overlapped)

The 6 new players that were the actual label are namely:

- Frank Kaminsky
- T.J. McConnell
- Devin Booker
- Josh Richardson
- Tyus Jones
- Nemanja Bjelica

Conclusion

Through the outlined methodology of what needed to be accomplished for the success of the project, and the use of Random Forest as a foundational starting point, a machine learning model was able to be developed to predict a 5th player for an NBA team line-up. The prediction made optimizes the team's performance through maintaining the home team's favourable outcome against the away team. Provided test data functions seamlessly with the trained model, and the results produced are of high accuracy and reliability. The task of predicting the 5th player is a complex classification problem that can be difficult to achieve with all the components and features that must be considered. The successful implementation highlights the extraordinary lengths of what can be created using the capabilities of machine learning.

GitHub Repository Link

Link: https://github.com/Calizard/NBA_Lineup_Prediction

References

[1] scikit-learn. *RandomForestClassifier*. Published for the API Reference of scikit-learn. [Online] Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[2] Baladram, S. (2024). *Random Forest, Explained: A Visual Guide with Code Examples.* Published for Medium. [Online] Available: https://medium.com/data-science/random-forest-explained-a-visual-guide-with-code-examples-9f736a6e1b3c

[3] AIML.com. (2024). What are the advantages and disadvantages of Random Forest?. Published for the Machine Learning Interview Preparation of AIML.com. [Online] Available: https://aiml.com/what-are-the-advantages-and-disadvantages-of-random-forest/