1. **Details on Data Processing (https://github.com/amazingzhi/MatchFixing/tree/master).**
   1. Original data

|  |  |  |
| --- | --- | --- |
| **Table 1: Original Data Variables** | | |
| **Variable Name** | **Explanation** | **Target Variables** |
| GAME\_ID | Unique Game Id identify Games |  |
| TEAM\_ID | Unique Team Id identify Teams |  |
| TEAM\_ABBREVIATION | Team Name |  |
| TEAM\_CITY | City that this team belongs to |  |
| PLAYER\_ID | Unique Player Id identify Players |  |
| PLAYER\_NAME | Player names |  |
| NICKNAME | Nick name of players |  |
| START\_POSITION | Start Position in Basketball (G, F, C) |  |
| COMMENT | Is Empty if this player take part in this game; Otherwise, giving the reason why this player not take part in this game |  |
| MIN | Minutes played in this game for this player |  |
| FGM | Field Goal Make |  |
| FGA | Field Goal Attempt |  |
| FG\_PCT | FGM/FGA |  |
| FG3M | 3 points Make |  |
| FG3A | 3 points Attempt |  |
| FG3\_PCT | FG3M/FG3A |  |
| FTM | Free Throw Make |  |
| FTA | Free Throw Attempt |  |
| FT\_PCT | FTM/FTA |  |
| OREB | Offensive Rebound get |  |
| DREB | Defensive Rebound get |  |
| REB | Total Rebound |  |
| AST | Assist |  |
| STL | Steal |  |
| BLK | Block |  |
| TO | Turn Over |  |
| PF | Personal Fouls |  |
| PTS | Points get | True |
| PLUS\_MINUS | points difference between with or without this player on the court | True |

* 1. Data Cleaning Player data
     1. Fill nan value of start position to bench. Drop nick name and comment. Change MIN to Seconds. (data\_cleaning\_player\_first.py)
     2. After first EDA, drop PF, TO, BLK, STL, OREB because these variables are not even related to player points in the same game. (data\_cleaning\_after\_EDA\_player\_second.py)
     3. Add pre, regular, and post season types, add home team id, loc, year, month, day. (data\_cleaning\_after\_EDA\_player\_second.py)
     4. Add oppo id (data\_cleaning\_add\_oppoid\_player\_third.py)
     5. Sort year, month, day, game\_id, team\_id all descending, start\_position ascending. (data\_cleaning\_sort\_datetime\_player\_forth.py)
     6. Get dummy variables for start position and season. (data\_cleaning\_player\_non\_time\_data\_get\_dummy\_fifth.py)
  2. Feature Engineering and Explanatory Data Analysis Player Data
     1. Feature Creation (feature\_creation\_player\_data\_non\_time.py): according to our understanding of basketball and some guess, we created below features and then test if these features are related to player performance.
        1. VS\_oppo last one game features: this is defined by the last time this player played against the same opponent. We think that there might be a relationship between the last a few times and this time of player performances against the same opponent.
           1. For example of any row of the original data, we can find a player id and team id. We added the opponent id based on the same game id and finding this player’s opponents players. By this way, we can find the last a few times this player played against the same opponent id.
           2. By EDA, we found the last time one player play against the same opponent has some relationships with this time player performances. We did this by correlation check and random forest feature selection methods. We also tried to find the last two times. However, because some times players miss some games, we have to drop a lot observations if we want to find all observations with their last two games against the same opponent.
        2. Last N games features: this is defined by last N games’ variables in the original data that this player played. It is possible that one player’s performance in the next game is related to his recent performances.
           1. For example of any row of the original data, we can identify the player ID and identify other rows happened before this row by date with the same player ID.
           2. Our logic is that if the target variable itself in the past N games is not related to the next game’s target variable, so does other variables in the last N games.
           3. Therefore, we apply PCAF method to determine how many last games can be features. We found last 13 games’ players’ points have relationship with the next game’s player points.
        3. Opponent Let Average: this is defined by the average performances of the are players who played against the same opponent and are at the same position and location. Our logic is that player performance is not only depending on himself but also depending on his opponent. We quantify the opponent’s defensive power by checking how an average player perform against this opponent with the same position and location. Another issue is how many games to look back to find all qualified players played against the same opponent. We tried different numbers and found that different numbers’ features are highly correlated. Therefore, we choose the highest correlation which is the 82, which means we first find last 82 games this opponent played and then find players who played against this opponent with the same position and location and average all qualified players’ performances to be features.
        4. Rolling Average last N games: this is defined by the average performances of a player in his last N games.
           1. Our logic is that the player’s next game performances maybe also related to his long-term average performances.
           2. Similarly, we found that all the numbers of averaged we tried are highly correlated. Therefore, we used the highest correlation one which is the last 82 games’ average features.
        5. Pure Performances: this is defined by player performances that rule out the effects of his opponent. This is calculated by last N games one player played minus the Opponent Let Features. We thought by this way, we may get a better correlation between next game’s performances and past game performances. However, we found that the Purred features are worse than original Last N games Features. Therefore, we didn’t include these features in our final features.
     2. Feature Engineering Player Data
        1. Feature Importance: we applied random forest algorithm to rank features we created and drop irrelevant features. We have two target variables, therefore, we created two datasets based on these two target variables’ dropping features.
        2. Feature transformation
           1. Standardization and normalization: we applied these two techniques to create two datasets to do model training.
           2. Dimension reduction: we applied Principal Component Analysis (PCA) to create two more datasets to do model training.
        3. Exploratory Data Analysis on created features
           1. We found that bench player’s correlation with target variables are different from starting player’s. Therefore, we split our data into two parts (bench players and starting players).
           2. We also checked the correlation difference between home game and away game. We didn’t find any significant difference.
           3. We also checked the correlation difference between different years. We found that different years have their correlations, but the differences are small.
           4. Notice that both player points get and Plus Minus are important to the game outcomes. Therefore, we set up two target variables to be predicted (Points and Plus Minus).
  3. Train Test Split: we split our data to train and test based on time period. 2004-2020 years are training data, while 20-22 are testing data. The reason we didn’t applied random split is that random split cannot present models’ generalization power when new data comes. However, if our testing data is the last a few years data, the test accuracy represents the accuracy of new data, which shows our model’s generalization power.
  4. Player Prediction
     1. Algorithms used: since each algorithm need to run 16 models (2 starting positions \* two target variables \* four data transformation methods), we only implemented 3 algorithms to save time. These algorithms are OLS, SGD, and Decision Trees.
     2. Accuracy Results:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **TABLE 2:**  **Accuracies of Player Performance Forecasting** | | | | | | | |
|  |  |  | Train (04-18) | Test (2019) | | | |
|  |  |  | MAE | RMSE | Max Error | MAE | R^2 |
| OLS | Bench | PlusMinus | 6.88 | 8.91 | 55.16 | 6.87 | 0.03 |
| PTS | 3.67 | 4.77 | 38.5 | 3.67 | 0.32 |
| Not Bench | PlusMinus | 9.32 | 11.64 | 51.62 | 9.3 | 0.12 |
| PTS | 5.01 | 6.39 | 47.44 | 5 | 0.43 |
| SGD | Bench | PlusMinus | 6.87 | 8.92 | 55.63 | 6.87 | 0.03 |
| PTS | 3.67 | 4.76 | 38.4 | 3.66 | 0.32 |
| Not Bench | PlusMinus | 9.32 | 11.65 | 51.17 | 9.31 | 0.12 |
| PTS | 5.01 | 6.39 | 47.6 | 5 | 0.43 |
| DTR | Bench | PlusMinus | 6.9 | 8.96 | 54.83 | 6.89 | 0.02 |
| PTS | 3.85 | 4.98 | 44.67 | 3.85 | 0.25 |
| Not Bench | PlusMinus | 9.48 | 11.85 | 52.33 | 9.48 | 0.09 |
| PTS | 5.22 | 6.64 | 55.84 | 5.21 | 0.39 |

* + 1. Get predicted player performances: we found that SGD algorithm had the highest accuracy results. Therefore, we used predicted Plus Minus and Points from SGD algorithm to be new features fed to game outcome prediction.
    2. Why not merging player features and game features? For each team, different games may have different players to participate, which makes it hard to set a fixed number of player features for each game.
  1. Feature Engineering and Explanatory Data Analysis Game Data
     1. Feature Creation (feature\_creation\_player\_data\_non\_time.py): according to our understanding of basketball and some guess, we created below features and then test if these features are related to game outcomes.
        1. Get Game data by sum player rows to get game rows: group by game ID and sum other variables of players to get game variables. For example, sum points get by players can get total points of a team for one game. This data can be seen as the original game data. We don’t use this data as features to predict game outcomes because these variables are in the same games.
        2. Team performance against the same opponent: this is defined by the last N games that this team played against the same opponent. After EDA, we found the last two games playing against the same opponent are related to the next game outcome. Another reason we chose two games is because during one season, at least two games played between any two teams, one for home game and one for away game. Therefore, we can find at least two games in last 82 games, which is the number of games in one season.
        3. Last N games team and opponent played: it is possible that the next game’s outcome is related to last N games team and opponent played because these features represent these two teams’ recent performances. We found that last 13 games played by team and opponent are related to the next game outcome. We determined this number by PCAF. Our logic is that if the points or point spreads are autocorrelated in a number of games, their relevant variables are also possible features that contribute to the next game outcome.
        4. Opponent let rolling average N games: this is defined by the average N games’ performances of teams that played against the same opponent. This opponent is also the opponent of the current observation’s team. The logic is that the game outcome of a team should also be related to this team’s opponent. Therefore, checking how other teams played against this team should be relevant to the next game’s outcome. We tried different number of games to be averaged and found 73 games’ features have the highest correlation. All other number of games’ features are highly correlated to the 73 games’ features. Therefore, we dropped those features to avoid multicollinearity.
        5. Pure past N games features: this is defined by the past N games features minus the Opponent let rolling average 73 games’ features. The idea was that we thought maybe we can roll out the opponent bias and get more purified and related features by minus opponent let rolling average games. However, we found these features are worse than original time series features in correlations comparison.
     2. EDA on new features.
        1. After we get all new features, we did correlation checks and feature importance checks by random forest algorithm. Some columns were dropped.
        2. Also, we found pre-season features are significantly different from regular-season features and post-season features in correlation. Only a small portion of games are pre-season games. Therefore, we dropped the pre-season games.