UNIVERSITY OF PISA

DEPARTMENT OF INFORMATICS

Master Degree in Data Science and Business Informatics

Distributed Data Analysis and Mining Project

Matteo Rofrano - Francesco Lanci Lanci - Simone Artesi - Gabriele Gori - Davide Piccoli

Academic Year 2023/2024

Contents

Chapter 1

Dataset description

The dataset utilized[1], is sourced from Basketball-reference[2], a website that aggregates comprehensive statistical and biographical information on NBA players and coaches. Specifically, our dataset encompasses all NBA games from the 1996-1997 season to the 2020-2021 season and consists of six CSV files: 'boxscore', 'player_info', 'games', 'salaries', 'coaches', and 'play_data'. However, for our purposes, only the first three will be utilized:

- The 'Boxscore' file collects, for each player and each game, in addition to the team affiliation and minutes played, 17 statistics related to their on-court performance.
- The 'Player_info' file provides details about each player's height, weight, and playing position.
- Lastly, the 'Games' file provides statistics for each played match, such as the points scored by the home team and those scored by the away team.

Each of these three CSV files will be merged into a single dataset using two common features: 'player_name' and 'game_id.'

1.1 Data Understanding

The main objective of the project is to perform player role classification based on the previously mentioned statistics from individual game encounters. Initially, an attempt will be made with three primary classes: C (Center), F (Forward), and G (Guard). Subsequently, two additional classes, C-F and G-F, representing hybrid players adapting to specific match needs, will be introduced.
During the analysis of NBA seasons, it was noted that three years were underrepresented: 1998, 2011, and 2020. Two factors primarily caused this underrepresentation. Firstly, the lockout, a game stoppage due to team owners' request to reduce player salaries, affecting the first two mentioned years. Secondly, the COVID-19 pandemic impacted the year 2020.

[1] https://www.kaggle.com/datasets/patrickhallila1994/nba-data-from-basketball-reference

[2] https://www.basketball-reference.com/

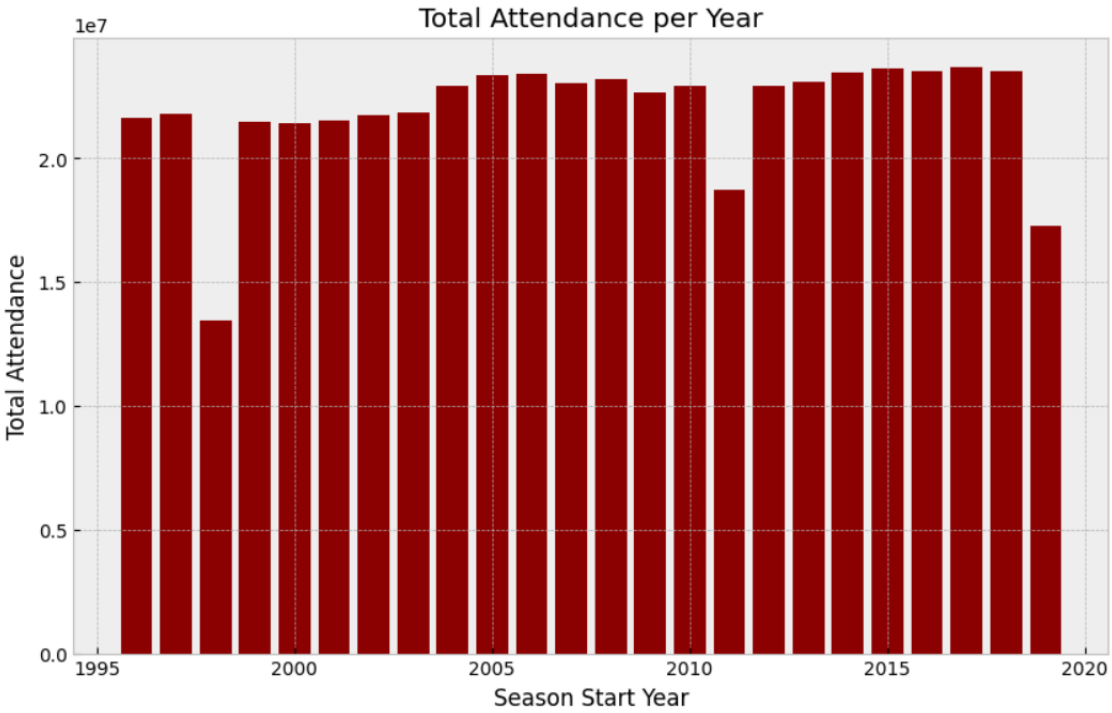


Figure 1.1: Attendance for each year

Subsequently, we observed that the three classes were not represented in the same way in our available data, especially the C class, which was approximately one-third compared to the other classes for each year. This discrepancy is normal, considering that the Center position is occupied by a single player on the court.
Nevertheless, analyzing the role distribution concerning the player statistics at our disposal, we observed that, on average, they followed the same distribution. In fact, from the plot below we can see that the number of players during the years has increased but the proportion of players in each role is almost the same.

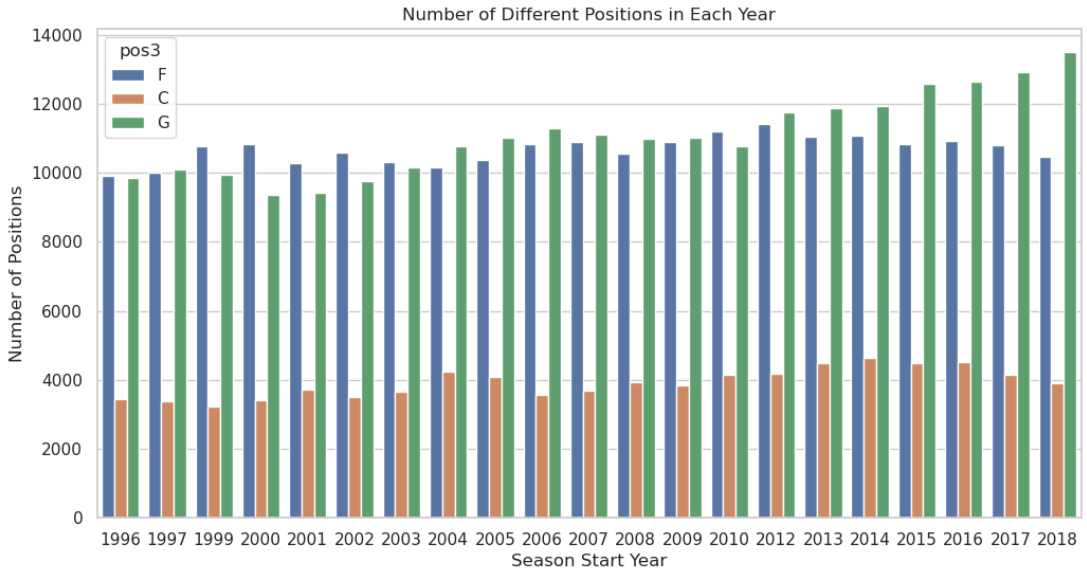


Figure 1.2: Position distribution per year

Moreover, by taking the distribution of the "+/-" and its composition by player role we can see that all the 3 positions follows the same normal distribution with the central role with a more peaked bell.

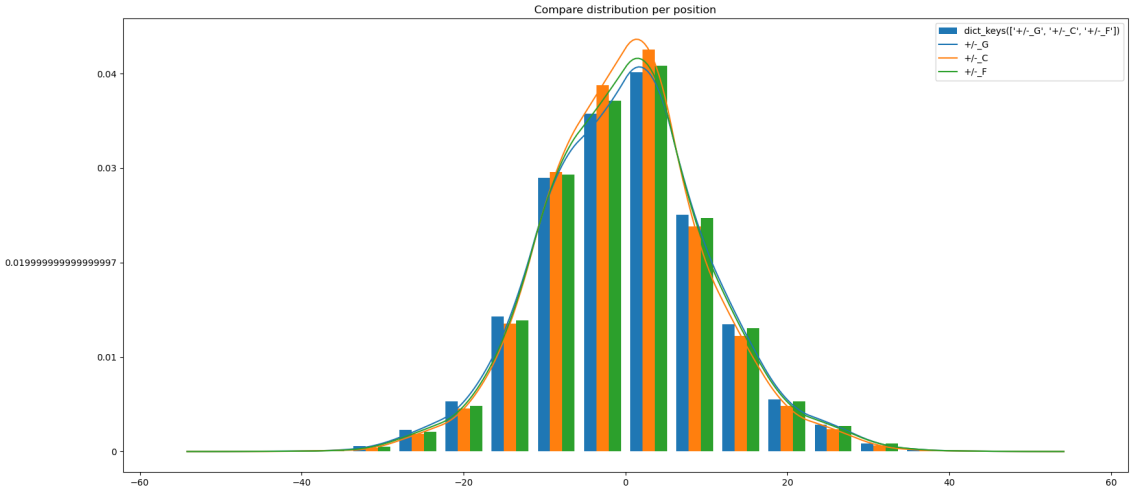


Figure 1.3: +/- distribution by role

Then we analyzed if in some attributes there has been a change during the years of the mean value. It is interesting to see that the average 3 points scored by a player in a certain match of a certain year has increased till nowadays. This fact underlies that the 3 point shootings are now more important and justifies the creation of more advanced statistics that take into account the different kinds of shooting (free shots, regular shots and 3 point shots). In the feature engineering section, we have created 3 new variables that leverage on this intuition.

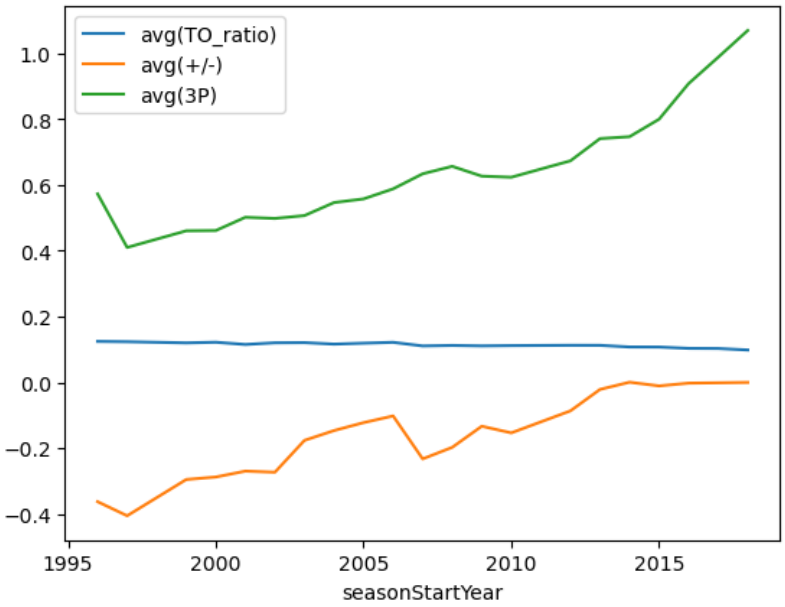


Figure 1.4: Historical changes

Chapter 2

Preprocessing

At first, we removed all those matches played in the 3 years (named in the previous chapter) with less attendance in the Boxscore and Games CSVs, then we removed columns deemed useless for our future tasks in all 3 of our CSVs, like "attendance", "notes", "birthDate" of the player, etc.

2.1 Missing values

In the Boxscore CSV there were 87 nulls in the "+/-" column (total between points made and conceded by the team in the match during the player's stay), which were reported when the player had played only a few seconds of the match, so we decided to remove these rows. Then, in the Games CSV there were no null values, while in the playerInfo CSV, there were 5 missing values in the weight of 5 players, so we decided to apply regression to replace these values.

2.1.1 Regression

Even though the number of missing values for this task is extremely low, we decided to perform a linear regression anyway. Our main purpose is the role classification, but we would also like to become familiar with the Spark framework; moreover, height and weight can be interesting data to analyze for the classification task (for more details, see Chapter 3). The first issue here was the weight and height format, which were expressed in the american metric system. First of all, we converted 'height' into decimal system (meters). Then, our training data (the entire dataset apart from the 5 missing values we want to predict) was plugged into a vector assembler and ultimately into the Spark linear regressor. Using the Spark RegressionEvaluator, we found:

- $R^2 = 0.666648$
- RMSE on test data = 14.902392
- Coefficients: 2.32073 with Intercept: -250.317968

The so-computed values were replaced in the original df_boxscore. At this point, our dataset didn't contain any missing value and we could perform joins and other operations.

2.2 Joins and managing data types

Here, we did the join between the 3 CSVs, via 'game_id' and 'playerName', resulting in a single boxscore dataframe, with inside the statistics of each player in each game (from boxscore), the height and weight of each player (from playerInfo), and whether each game is from the regular season or playoffs (from games).
Next, we noticed that all the statistics were of type string, so we converted all possible ones to integers and removed all rows that contained non-numeric statistics, as some players had strings like "Did Not Play" and "Did Not Dress."
A feature that deserved more work instead was MP (minutes played), which had a xx:xx format, so we converted everything to seconds.
In the end, from the Pos feature that contained the position of each player in each match (which had 7 possible combinations C, F, G, and the combinations between C and F and G and F), we created 2 features: one with only 3 roles (C, F, and G), the other with 5 roles (C, F, G, C-F, and G-F, since F-C is equal to C-F and F-G is equal to G-F).

2.3 Feature Engineering

We decided to add to our data some advanced stats that are not present in the original data. In particular, we added the "Effective field goal percentage" EFG. This statistic adjusts for the fact that a 3-point field goal is worth one more point than a 2-point field goal. The Effective Field Goal Percentage can tell you at a glance which team is having more success from the field. The team with the higher percentage is scoring more effectively from the field. This metric can be useful since as showed in image (1.4) the 3 points scores are getting more and more important in today's matches.
Another useful metric is True shooting percentage, aka TSP, is a metric that factors a player's or a team's performance at the free-throw line and considers the efficiency of all types of shots. This NBA statistic helps us compare players with varying responsibilities and shooting abilities on the floor. A higher True Shooting Percentage generally indicates a more efficient scorer, while a lower percentage indicates a less efficient scorer. So this is another metric that takes into account the different types of shooting.
The last metric we included is the Turnover Ratio (TO_ratio). Turnover percentage is a statistic used to measure a player's ability to take care of the ball. It is used to measure how often a player turns the ball over while they are in possession of it. Turnover ratio attempts to normalize turnovers for each team by showing what percentage of possessions ended in a turnover for a team.

$$TSP = \frac{PTS}{2 * FGA + 0.44 * FTA}$$

$$EFG = \frac{FG + 0.5 * 3P}{FGA}$$

$$TO\_ratio = \frac{TOV}{FGA + FTA * 044 + AST + TOV}$$

2.4 Distribution analysis

In this section we will discuss some aspects about our features distribution. In the first plot below we can see that the majority of matches regard the regular season. It is important to note that the majority of the numerical features have a skewed distribution. An example is the PTS feature where most of the observations have 0 since there are players which do not play and some that score very few points.
Just the '+/-' show a perfect normal distribution while there are others like weight, height and EFG that have a bimodal distribution.
Since we are dealing with a classification task and we use 2 tree-based model (which are not affected about the skewness in the data) over 3 we have decided to not try to find a specific technique to transform these variables in order to have a normal distribution but it could be a possible improvement that can boost the neural network performance.

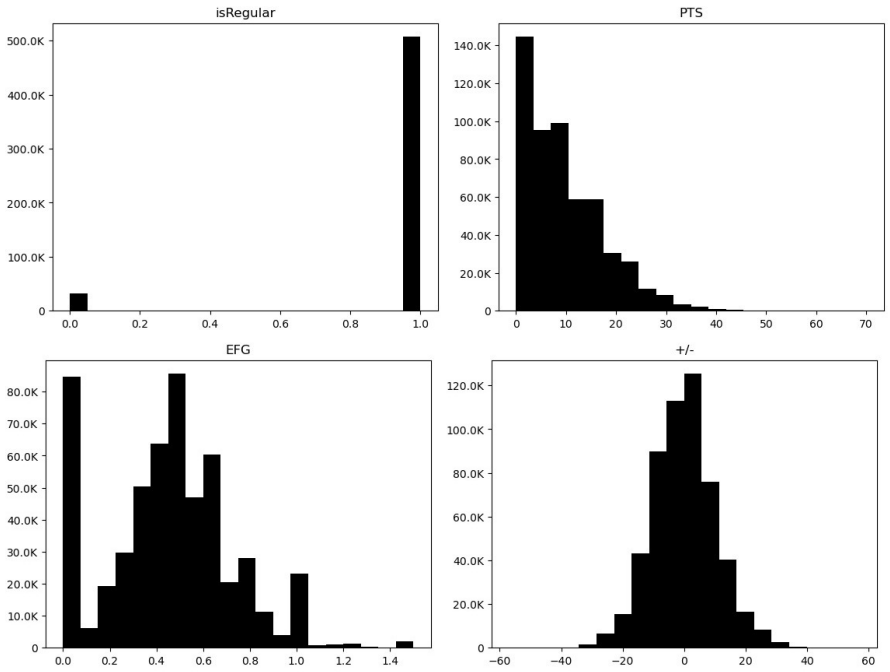


Figure 2.1: Features distribution

2.5 Correlation analysis

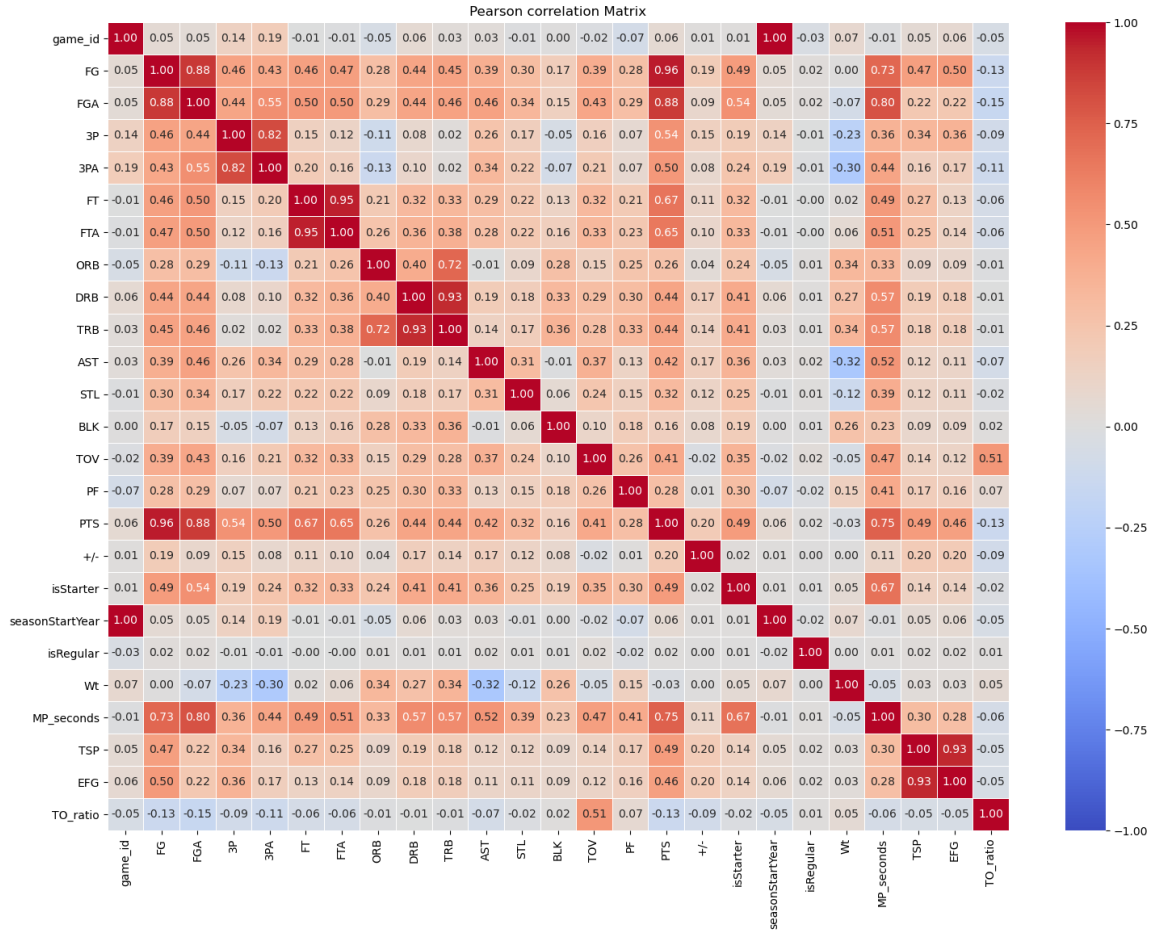


Figure 2.2: Pearson correlation matrix

In order to reduce the dimensionality of our data and avoid the multicollinearity problem we decided to analyze the correlation between our numerical variables. In particular we computed both the Pearson (linear) and the Spearman correlation (non-linear) by using the vectorAssembler function.
We have set as threshold the 90% of correlation, so if 2 variables have a correlation higher that 90% we can drop one of the 2. We noticed that the 2 methods tend to give us more or less the same result so we will report only the Pearson correlation matrix.
From the plot we can see that we can remove the following features: 'game_id', 'FG', 'FT', 'TRB', 'TSP'.

Chapter 3

Classification

For this task, we finally exploited the clean Boxscore dataset. This chapter will be divided in 2 sections, one for the 3 classes problem and one for the 5 classes problem. In the latter we also considered 'F-G' and 'F-C' roles, while in the former version the position column has been just truncated to the first one. The data preparation is almost the same, except of course for the target variable.

3.1 Data Preparation and methodology

We needed to transform categorical values into numerical values. To do so, we used the ml.Spark *StringIndexer*, then we just dropped the starting ones. We decided to drop also useless features, like 'game_id', 'PlayerName', 'seasonStartYear', and 'teamName' since they don't deliver any information for our purposes. We are basically left with just performance features related to actual games. Also personal stats, like 'height' and 'weight' have been dropped, but we will discuss this point later on. The next step was to get rid of too correlated features, and they are:

- FG
- FT
- TRB
- TSP

For this purpose, we selected a >90% threshold for Spearman. The final statistics for the training are 18, and they were all normalized with a MinMaxScaler between 0 and 1. At this point, we splitted data into training, validation and test set:

- Test split: 0.7/0.3
- Validation split: 0.8/0.2

A random seed was used to make results comparable. Every algorithm was tested using a **random search** on the validation set to fine-tune hyperparameters, and here we report the best result we have obtained using each of the proposed models, both for 3 and 5 classes.

3.2 3 classes problem

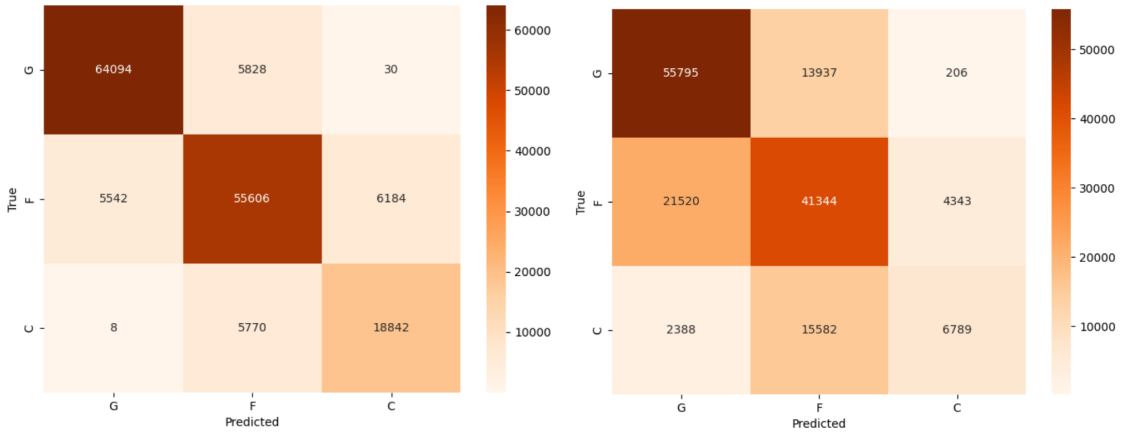
Here we discuss results obtained with only F, G, and C classes. It is an unbalanced problem, since their ratio is (roughly):

- G: 43%
- F: 42%
- C: 15%

We want to underline that personal information, such as 'Height', has been removed for a particular reason. We have done some tests with them, and we have discovered that their correlation with the role is too high, making all the other features almost useless. Since the goal of our project was to identify the role through actual game statistics, we removed everything else. However, in Figure 3.1(a) we report a test with the 'Height' feature, that we can use as a benchmark for other models.

3.2.1 Decision tree

As we said before, we made some initial tests including the 'Height' feature, and in this case the accuracy is around 85.5705% on the test set, which to us is the ideal score. 'Height' (Ht) feature importance was around 90%. Even though we were satisfied with this performance, we wanted to do a step further to keep things interesting. We underline this fact because we believe that a more interesting classifier should only consider game stats, to see whether the model can infer the role only based on player behaviors rather than player personal information (which, as we said, occupy all the feature importance). Best model (considering only game stats) has: *maxDepth*=9, *maxBins*=64, *impurity*='entropy'. Validation accuracy is 64.2182%, while test accuracy is 64.1806%. Overfitting was effectively avoided. We are aware that accuracy has decreased noticeably, but we also consider this task definitely more interesting (and challenging).



(a) With 'Height' feature (b) Only game statistics

Figure 3.1: Confusion matrices for Decision Tree

Results appear so different using only game statistics, the problem becomes harder for the classifier. Even though results as not as good as with 'Height', of course, we

consider our results to be good enough since they overperform a trivial classifier. Moreover, it is good to keep in mind that the problem is unbalanced. 'Center' class has fewer records, and the Decision Tree struggles to identify it properly. Given that the 'Forward' class is a role which appears like a mixture of the other two (talking about game statistics), it is way harder for a classifier to identify it. The

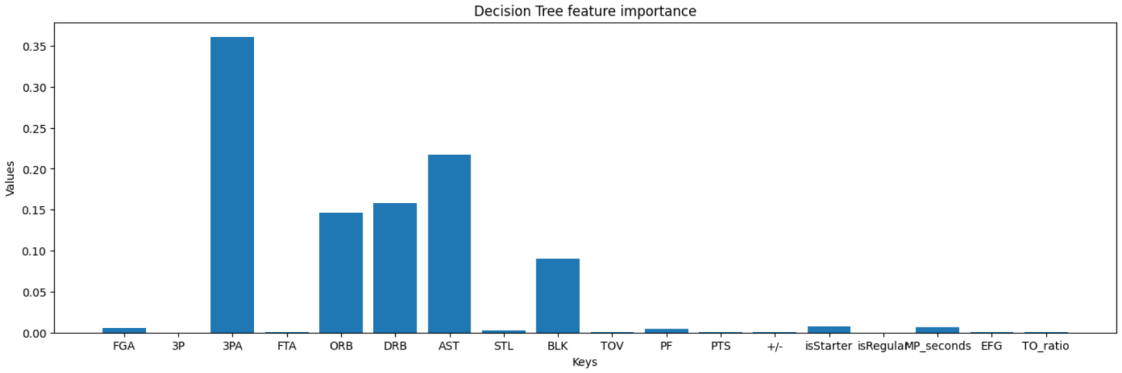


Figure 3.2: Feature importance for Decision Tree (only statistics)

most important feature appears to be *3 points attempts* (3PA), followed by *assists* (AST) and *blocks* (BLK). This result makes sense to us, actually: these are statistics related to player behavior on the court, and they gather plenty of information about the areas preferred by the player. 3PA is a good example, because 3 points attempts can only be made outside the 3 points line, telling us that the player is playing far from the basket (very common behavior for a Guard player); C class and G class are not confused too many times, this one can be a good explanation. Figure 3.2 clearly shows that 5 major features gather information about the role, while the other ones don't deliver enough details.

3.2.2 Random Forest

The best parameters are: numTrees= 50, maxDepth= 9, maxBins= 16, impurity= 'gini', with an accuracy of 64.9447%. in Figure 3.3 we can observe a slightly different feature importance, but substantially we still have 5 main features with the addition of 3 points (3P), which is obviously related to 3 points attempts (3PA).

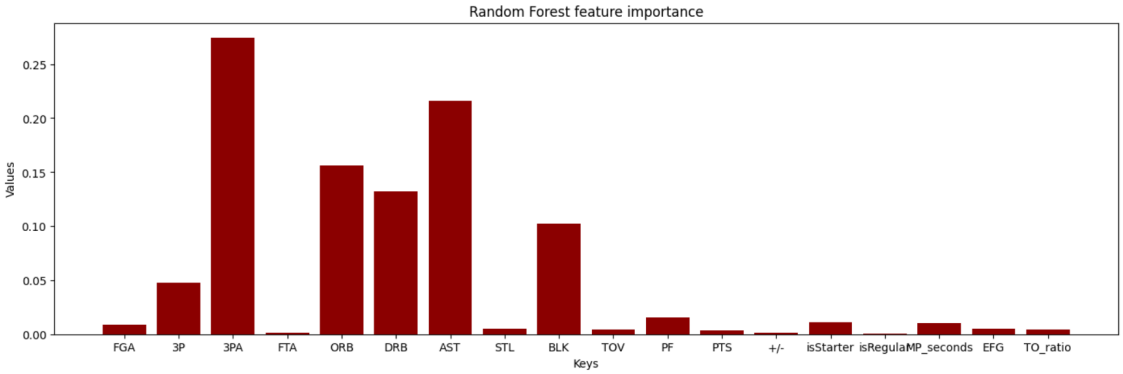
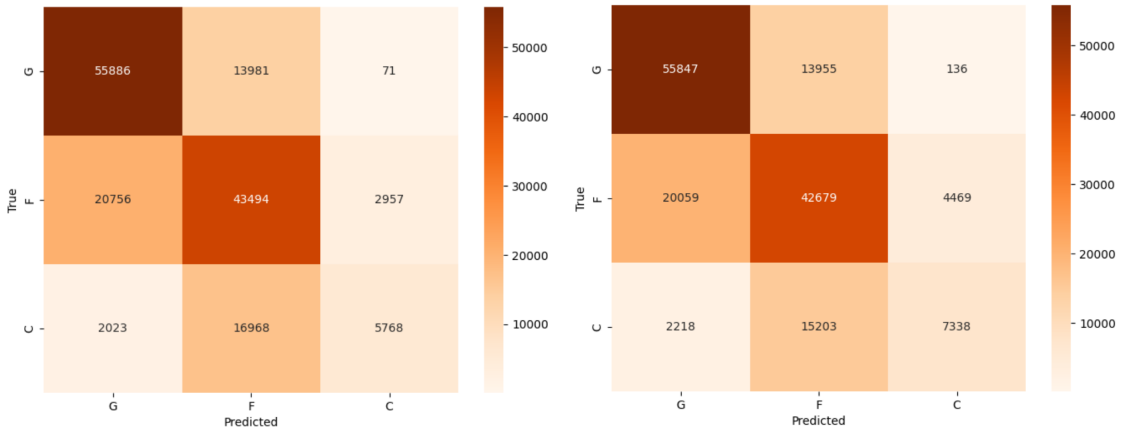


Figure 3.3: Feature importance for Random Forest (only statistics)

3.2.3 Multilayer perceptron classifier

Test accuracy here is 65.3869%, without overfitting, using a [30, 30] network, with maxIter=80 and stepSize=0.01. In random search we tested also 3, 4 and 5 hidden layers nets, with worse results. Performances have increased compared to Decision trees, but running time is way longer. Anyway, the same pattern we have discovered for the other models appears, since F class is often confused with both G and C.



(a) Random Forest (b) Multilayer perceptron

Figure 3.4: Confusion matrices

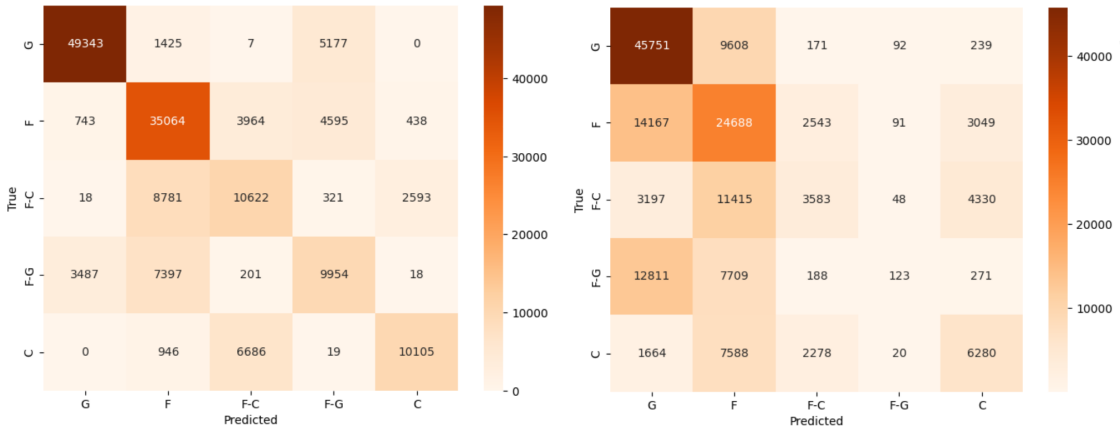
3.3 5 classes problem

Here we discuss results for an harder task: since players can have two roles, we decided to also include them in the classification problem. There are some players in our database which has two positions, and they can be 'F-G' or 'F-C', meaning that they cover both positions on the court during a match (our data never allow a 'G-C' role). This is still an unbalanced task:

- G: 35%
- F: 28%
- C: 13%
- F-G: 13%
- F-C: 11%

3.3.1 Decision tree

Test accuracy with 'Height' feature is 71.0841%, while without it drops to 49.6745%. Optimal parameters were the same as for 3 classes. We don't observe any overfitting issue here too. Here we immediately notice that hybrid roles are very difficult to be detected for a classifier as Decision trees, with a disappointing score for 'F-G'. Given that these roles have no signature features, it is not surprising afterall. Feature importance remains the same as before, but now '3PA' is even more important (>40%).



(a) With 'Height' feature (b) Only game statistics

Figure 3.5: Confusion matrices for Decision Tree (5 classes)

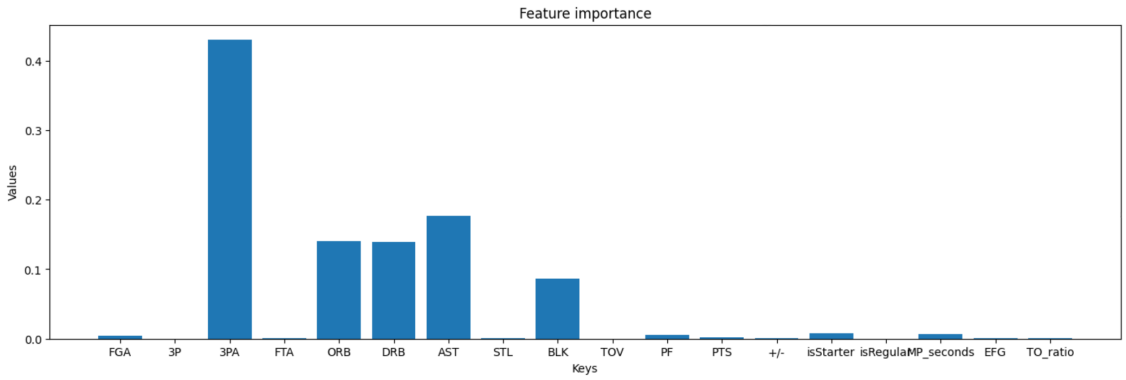


Figure 3.6: Feature importance for Decision Tree (only statistics, 5 classes)

3.3.2 Random Forest

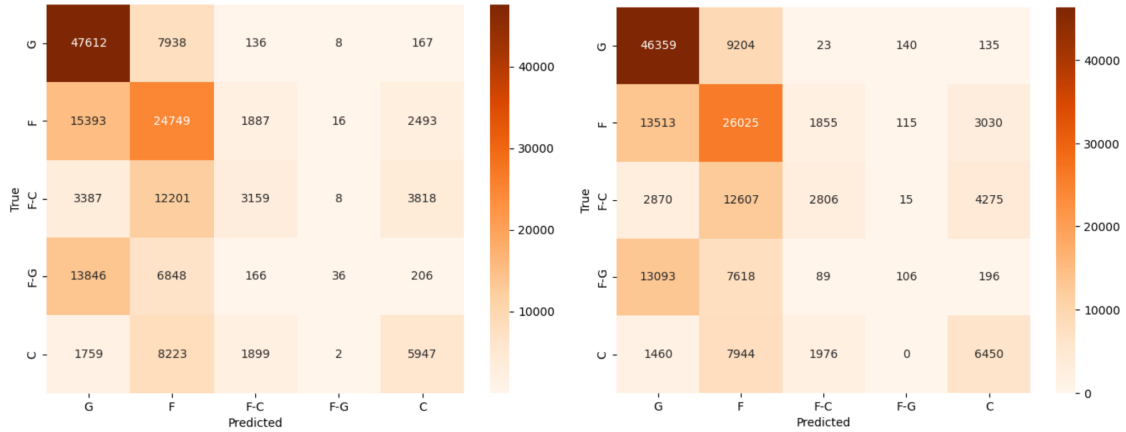
Here the best parameters are numTrees= 50, maxDepth= 9, maxBins= 64, impurity= 'gini', just maxBins number is different from the 3 classes case. Accuracy is 50.3403%. Detailed results can be found in Figure 3.7(a).

3.3.3 Multilayer perceptron classifier

The test accuracy is 50.4904%, using a maximum number of iterations of 80 and a bigger stepSize (0.03) with respect to the 3 classes problem. Apparently, results for the two hybrid positions are worse with respect to the Decision tree. It's interesting to see that G predictions very often involve actual F and F-G classes, and we observe this pattern through all the models.

3 classes			**5 classes**		
D.T.	**R. F.**	**M. Perceptron**	**D.T.**	**R. F.**	**M. Perceptron**
64.1806%	64.9447%	65.3869%	49.6745%	50.3403%	50.4904%
Benchmark: 85.5705%			Benchmark: 71.0841%		

Table 3.1: Results summary



(a) Random Forest (b) Multilayer perceptron

Figure 3.7: Confusion matrices (5 classes)

3.4 Conclusions

We discovered that predicting the role exploiting NBA game statistics is a feasible task. It becomes way easier if we also consider personal data, such as height, since this feature in particular determines the role of the player in an NBA team, in most cases. In conclusion, game statistics can tell us some interesting details about a player's performance, and more importantly, they encode some behaviours on the court. Double roles are difficult to detect, since our simple and freely available statistics could not gather enough information. Possessing more advanced stats (we think about court zones activity), we could obtain some higher performances.
Teams could use this model to analyze the performance of potential draft picks or new players during scouting in order to identify if they fit in a particular available role in the team or not.