Machine Learning

Data mining in WNBA playoffs qualifications

Emanuel Silva Gestosa - up202005485 José Albano Gaspar - up202008561 Maria Sofia Gonçalves - up202006927

Business Understanding

Requirements and definition of business goals

- Goal: Predict for a given year which teams are going to the playoffs
- How does WNBA work?
  - 2 conferences: Western and Eastern
  - First part of season: Each team plays against each other
  - Second part of season: Best 4 teams of each conference play the playoffs
  - Playoffs use an elimination format, with quarter-finals, semi-finals and finals.
- We are given 10 consecutive years of data on the WNBA league, and want to create a model that will predict which teams will make it playoffs on the 11th year, with only the data available at the start of that season (players, coaches, and any past information)

Business Understanding

Data Mining goals

- Metrics to evaluate success of the model
  - Accuracy
  - Precision
  - Recall
  - AUC
- So, it was defined that the model would be considered successful if:
  - Accuracy ~> 80 %
  - Recall and Precision ~> 75 %
- Why? If this metrics are achieved, it is highly likely that the model classifies wrongly around only 2 teams (considering the average number of teams per season).

Data Understanding

The provided dataset

- The following information was available:
  - Teams – contained information of the teams like its name, conference, the statistics of that team for a given year like wins, losses, points, rebound, if they reached playoff on that year, etc .
  - Player_teams – contained the statistics for a given player
  - Players – contained biological information about the players, namely height and weight
  - Coaches – contained the teams that a given coach trained in a given year and also the number of victories, losses and stint of the coach for that season
  - Teams_post and Series_post – contained information of how a given team performed on their last participations on playoffs. This included results, phase of the playoff reached, wins, losses, etc.

Data Understanding

Players that never played in any team

- We realized that some players, even though that they are present in the *players dataset*, don't have any row in the *player_teams* dataset, meaning they never played in any WNBA team, at least for the years we have data on.

```
# Select rows where 'bioID' is NOT in 'playerID'
players[~players['bioID'].isin(players_teams['playerID'])]
```

	bioID	pos	height	weight	college	collegeOther	birthDate	isAlive
0	abrahta01w	C	74.0	190	George Washington	NaN	1975-09-27	Y
2	adairje01w	C	76.0	197	George Washington	NaN	1986-12-19	Y
3	adamsda01w	F-C	73.0	239	Texas A&M	Jefferson College (JC)	1989-02-19	Y
5	adamsmi01w	NaN	0.0	0	NaN	NaN	0000-00-00	Y
6	adubari99w	NaN	0.0	0	NaN	NaN	0000-00-00	Y
...	...	...	...	...	...	...	...	...
874	wrighfa01w	G	66.0	130	San Diego State	NaN	1973-01-28	Y
875	wrighmo01w	G	71.0	178	Virginia	NaN	1988-07-15	Y
881	yasenco01w	G	72.0	160	Purdue	NaN	1973-12-05	N
889	zhengha01w	C	80.0	254	NaN	NaN	1967-03-07	Y
890	zierddo99w	NaN	0.0	0	NaN	NaN	0000-00-00	Y

338 rows × 8 columns

Figure 1: Players that never played in any team

- We retrieved 338 rows of players that never played for any team.
- We also noticed that a lot of players have missing information for the columns *college* and *collegeOther*, and have a value of '0' for the *height* and *weight* fields.

Data Understanding

Players' height and weight

- Since we found some weird values of *height* and *weight* on the *players* table, we decided to plot this information using boxplots. Note that this plots where done only for the players that played in any team at some point, since only these players are relevant for us.

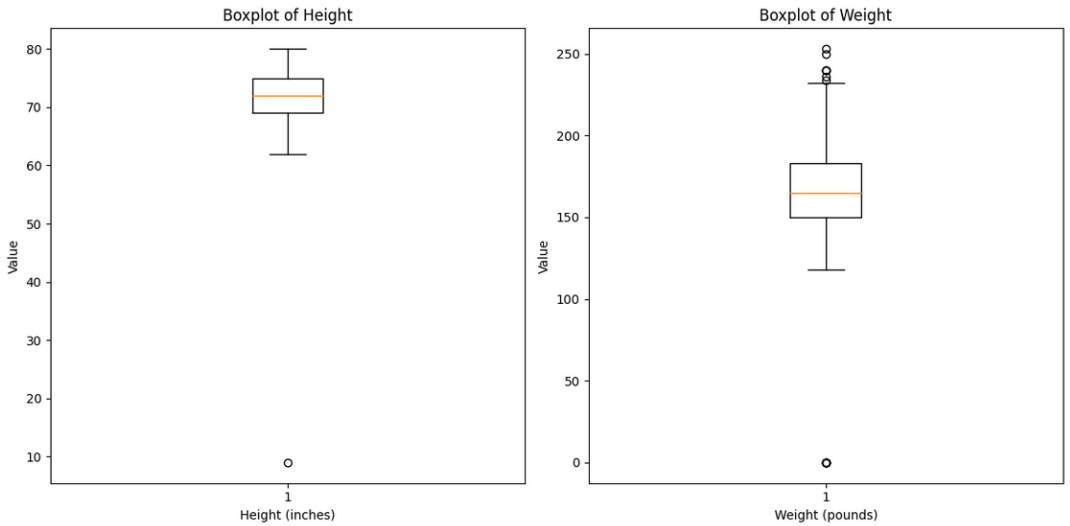

Figure 2: Players' height and weight boxplots

- We can clearly see that we still have players with height or weight at '0', and a few outliers that are heavier than normal, but they are probably just heavier players and not errors in the dataset, unlike the '0's.
- We also noted that all the players are in a rather short range of heights, which was to be expected since basketball is a sport where a player usually needs to be tall to be successful.

Data Understanding

Awards

- To better understand what the awards dataset represented, some operations were performed:
  - Using *pandas,* it was noticed there were several types of awards, some more important than others. As an example, there was a sportsmanship award that is not very useful to accomplish our goals.
  - A plot was made to see the distribution of the awards for the players. The result shows that there is a small group of awarded players, while the average player has no awards or very few.

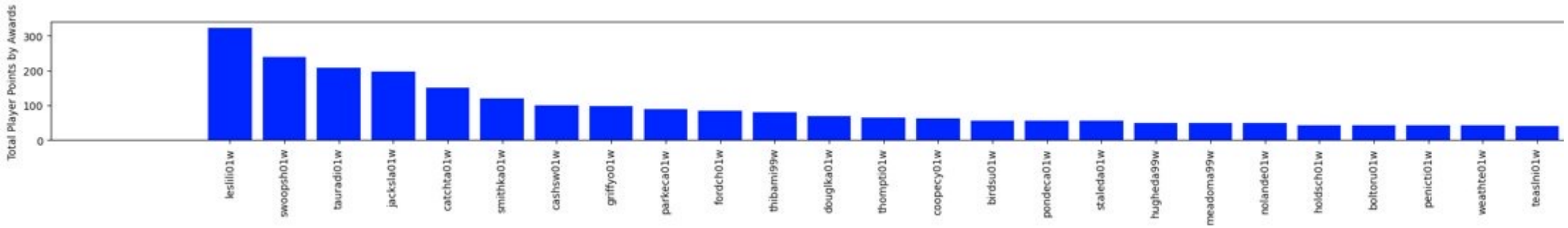

Figure 3: Topmost awarded players

Data Understanding

Coaches

- For coaches, we decided to analyze their win ratios for all the years combined, to try and see if there were any coaches that consistently got better performance than the others.

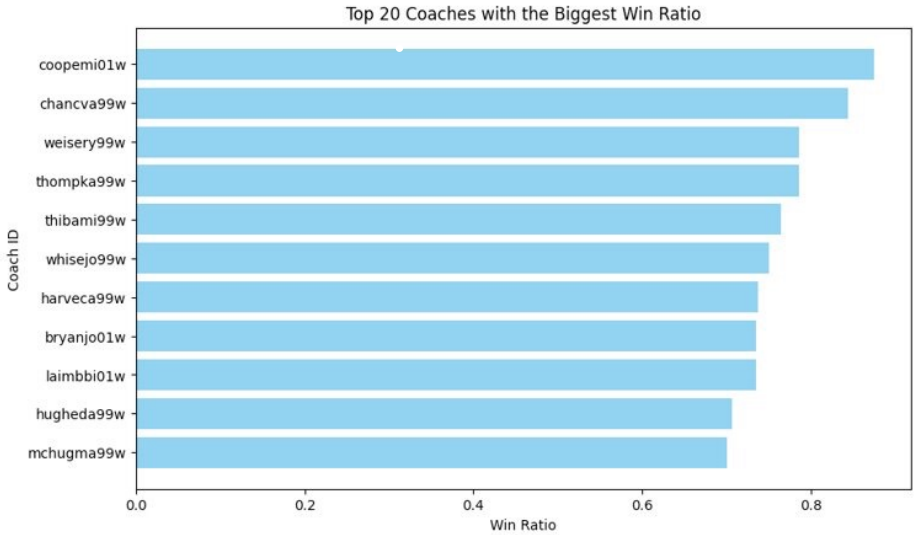

Figure 4: Top coaches by win ratio

- We can clearly understand that the coaches *coopemi01w* and *chancva99w* have a very high win ratio throughout the years, and any team that is coached by them should have a much greater chance of making it to the playoffs.

Data Understanding

Coaches

- Still on the coaches' dataset, their wins and losses on post season (playoffs) were explored. Here is an example of some of this plots:

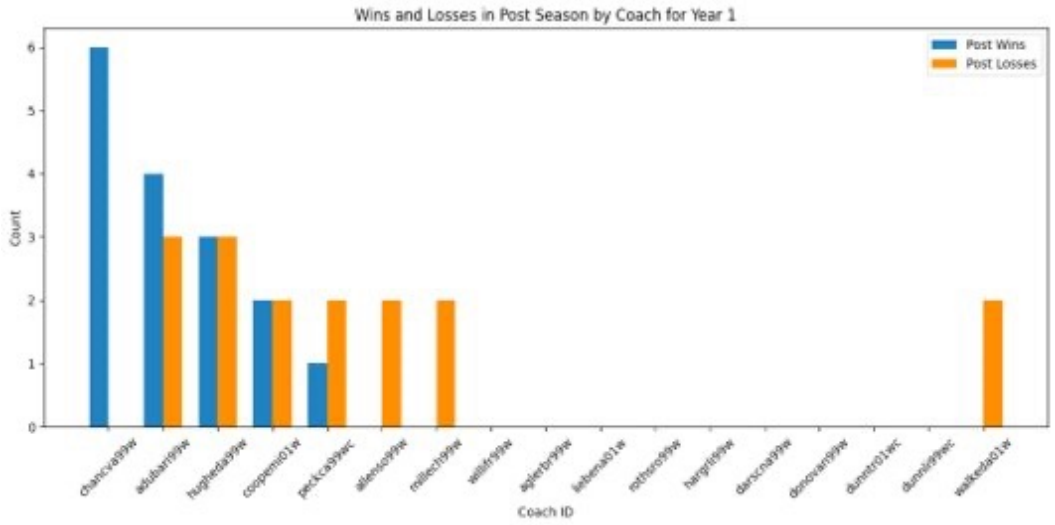

Figure 5: Wins / losses of coaches on playoff for season 1

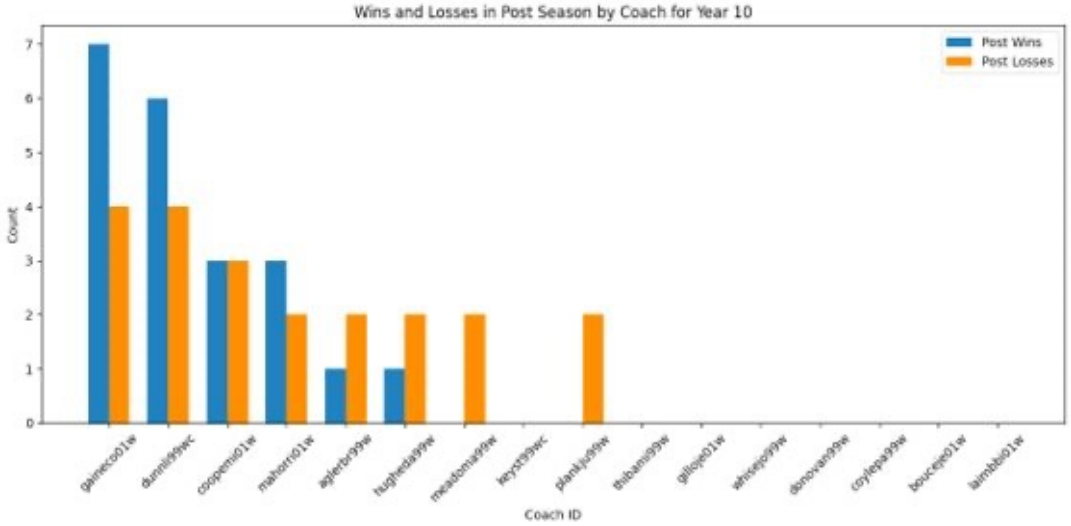

Figure 6: Wins / losses of coaches on playoff for season 10

Data Understanding

Team appearances, wins and losses

- We wanted to know which teams, throughout the years, went the most to the playoffs and which had more wins and losses in the playoffs.

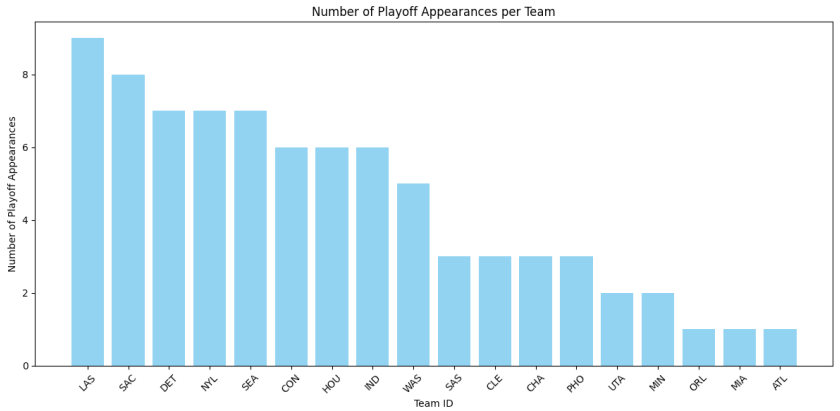

Figure 7: Number of playoffs appearaces by teams

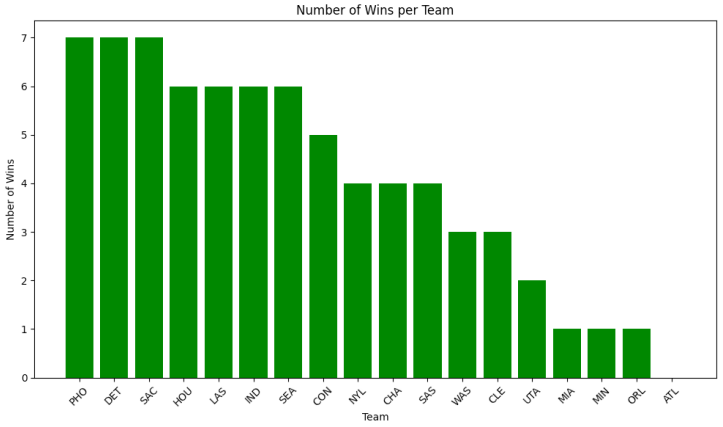

Figure 8 (bottom): Number of wins by teams in the playoffs

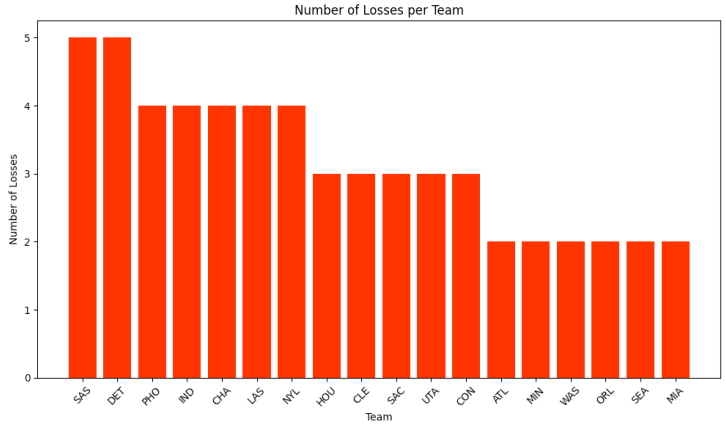

Figure 9: Number of losses by teams in the playoffs

Data Understanding

Correlation matrix for a subset of features of the Teams' table

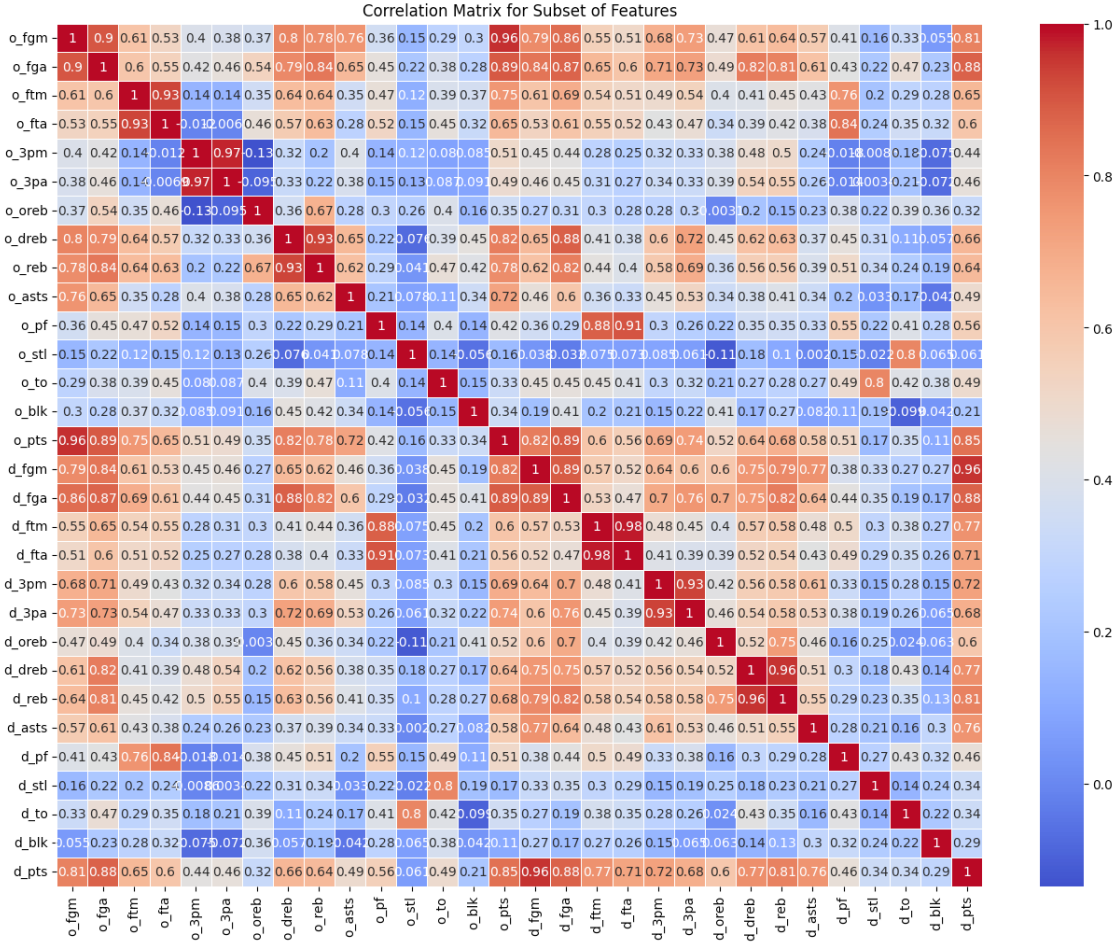

Figure 10: Correlation matrix for a subset of features of the teams' table

- A correlation matrix for a subset of the features of the teams' table was done to verify if any features where highly correlated with each other.
- We verified that there is a high correlation for features that end with 'm' with the ones that end in 'a' (e.g., 'o_fgm' with 'o_fga').
- Another kind of correlations we found, though not in this matrix, is that the 'o/d_pts' is based on the 'o/d_fgm', 'ftm' and '3pm'; and the 'o/d_reb' is the sum of 'o/d_oreb' and 'o/d_dreb'.

Data Understanding

Correlation matrix for a subset of features of the Player_Teams' table

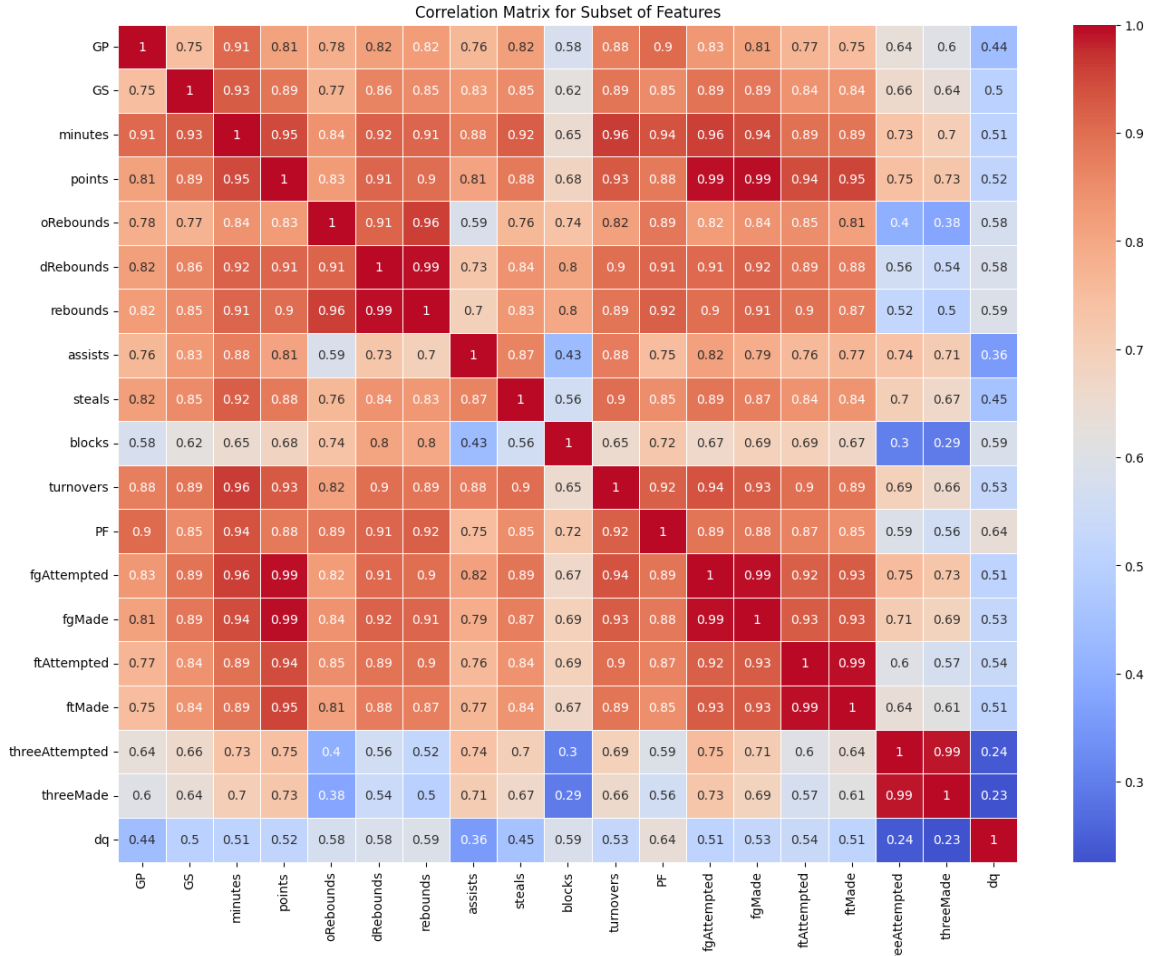

Figure 11: Correlation matrix for a subset of features of the player_teams' table

- A correlation matrix was also created using the player_teams' table. Our aim was to understand the correlations between defensive and offensive moves done by the players.
- We can conclude that most of these features are correlated with each other.

Data Preparation

Data integration

- Our approach was to create a unique table that had representative features of each individual table that was provided to us.
- So, the main table was the teams' table since we want to predict which of them are going to the playoffs.
- To represent the other tables, several features were computed and then integrated on this main dataset.

  Some of those features: *teamYearOverallScore, teamYearAwardPoints, topCoaches, mean_height, previous_playoffs, ...*

Data Preparation

Assessment of dimensions of data quality

- Data accuracy: The .csv files given to us represent real seasons of the WNBA, so the data can be considered very accurate.
- Completeness: The data had some outlier players, combined with some missing values. In the following slides what was done to deal with those cases is thoroughly explained.
- Consistency: The information across the different .csv was uniform. For example, a player only appeared on one team for a given year (not on a certain team on one row for a year and, for the same year, on another team on a different row).
- Reliability: The data is as reliable as possible, since it reflected the information of the official WNBA teams.
- Integrity: Overall, the data can be considered completely integral.

Data Preparation

Redundancy

- When exploring the dataset, several features were found that had the same value for all rows. Those were redundant and if left on the final dataset, would contribute to a higher noise on the dataset, resulting on worse results when testing the models.
- What was done?
  - Simply remove all these features in all given datasets
- Examples: *lgID* (always equal to 'WNBA'), *divID* (always NaN), etc
- Based on our correlation matrixes, we found a lot of correlated features. To avoid having redundant data, we did the following:
  - Replaced features like 'field goals made' and 'field goals attempted' with a 'field goals accuracy'.
  - Calculated for each player an overall score, to unite all their stats, which were heavily correlated.
  - Deleted ofensive and defensive rebounds features, since there is also a 'rebounds' feature that summarizes those two.

Data Preparation

Missing values and Outliers

- As mentioned in the Data Preparation part of this presentation, we noticed some missing values and outliers, mainly concerning the player's height and weight.
- We temporarily removed these outliers and used the values from the other players to calculate an average BMI value.
- Then, in the players who had a reasonable height value, but their weight was equal to 0, we used the average BMI value obtained to calculate this missing value.
- We adopted a similar approach for players without height but with a weight value.
- For players with weight and height equal to 0, we replaced them with the average values of the other players.

Data Preparation

Data transformation for compatibility of algorithms

- To achieve our goals, we tried a couple of algorithms.
- Among those, we tried K-NN and SVM algorithms.
- These two models in particular work based on distances and vectors, so, to get a better performance on those models, it is essential that the dataset is normalized, this is, all the features share the same scale. Only with that done, these algorithms can give their best results.
- To perform this operation, it was used the *MinMaxScaler()* from *sklearn* library.

Data Preparation

Feature Engineering from tabular data

- Several features were created:
- *TeamYearOverallScore* – gives a score representing how well the players of a team, on average, played on the previous season. Used the official formula :

  Game Score = (Points + 0.4 × Field Goals Made - 0.7 × Field Goals Attempted - 0.4 × (Free Throws Attempted - Free Throws Made) +

  + 0.7 × Offensive Rebounds + 0.3 × Defensive Rebounds + Steals + 0.7 × Assists + 0.7 × Blocks - 0.4 × Personal Fouls - Turnovers) / Minutes Played
- *Previous_playoffs* and *previous_playoffs_wins* – represent the number of times a team went to / won the playoff.
- *Top_coaches* – as previously identified on data understanding, there were very successful coaches. This feature is a binary feature that tells the model what teams were trained by one of those coaches.
- *Win_ratio* and accuracies – creation of ratios for the number of wins and to compute the accuracy over some statistics like the 3 points, fields goals and free throws.
- *TeamYearAwardPoints* – this feature computes the mean of award points per player on a given team. These points were attributed based on the authors' s perspective of how important each type of award was.
- *Mean_Height* – computed the mean height of a team.

Data Preparation

Feature selection

- After the entire dataset was computed, we used feature selection to try to tune our models.
- For this, we created a script that removed one feature at a time.
- Then we ran the algorithms again (testing on years 6-10) to see if we were having improvements.
- If the feature removal improved the accuracy in more than 50% of the cases (improvements on 3 or more seasons), then the feature was removed, otherwise kept.
- We also did a lot of manual feature selection, where we would remove certain groups of features, and check if it improved the performance.

Data Preparation

Data leakage and sliding window

- Based on our analysis of the data, we found some features that were data leakage, this is, things that were not meant to be known at the beginning of the season.
- Almost all features on our final dataset were on this category. Example: the features indicating if the team reached firstRound, semifinals or finals of a playoff as well as all the statistics regarding scored points, rebounds, etc.
- To solve this issue, we created a sliding window technique to recompute these values. This strategy consists in computing the value for a given year based on the last N years. We have also tried to give more importance on more recent years (of those N) by adding different weights to them.
- On the process, it was noticed some teams did not play in all provided seasons, so we had to add rows for those teams in those years. To populate these rows with numbers, we applied 2 strategies, one containing all zeros, and other containing the quantile (0.25) of the other teams, originating 2 different final datasets.

Data Preparation

Final Datasets

- We ended up having 2 different final datasets.
- They have the same features and differ on the construction process as explained in the last slide.
- These are the final features:
  - *TmID, year, confID, playoff, teamYearOverallScore, previous_playoffs, previous_playoff_wins, top_coaches, rank, o_fga, o_fta, o_3pa, d_fga, d_fta, d_3pa, winRatio, teamYearAwardPoints, confW, confL, attend, mean_height*

Predictive

Tasks and algorithms

- To solve this classification problem, we used the following models.

```
# Initialize the models with hyperparameter tuning options
models = [
    (DecisionTreeClassifier(), {'max_depth': [None, 5, 10, 15]}, data),
    (RandomForestClassifier(random_state=20),
        {
            'n_estimators': [50, 100, 200],
            'max_depth': [5, 10, 15],
            'min_samples_split': [2, 3, 5],
            'min_samples_leaf': [1, 2, 3],
            'max_features': ['sqrt', 'log2'],
            'bootstrap': [True, False],
            'criterion': ['gini', 'entropy'],
        }, data),  # Random Forest
    (SVC(probability=True), {'C': [0.1, 1, 10], 'kernel': ['linear', 'rbf']}, data_norm),
    (KNeighborsClassifier(), {'n_neighbors': [3, 5, 7]}, data_norm),
    (MultinomialNB(), {
    'alpha': [0.1, 0.5, 1.0, 1.5, 2.0],
    'fit_prior': [True, False],
    'class_prior': [None, [0.1, 0.9], [0.3, 0.7], [0.5, 0.5]]
    }, data),
    (BernoulliNB(), {'alpha': [0.1, 0.5, 1.0]}, data),
    (GaussianNB(), {'var_smoothing': [1e-9, 1e-7, 1e-5, 1e-3, 1e-1, 1.0]}, data)
]
```

Figure 12: Used algorithms and parameter grid for hyperparameterization

- Decision tree
- Random Forest
- SVM
- KNN
- Naïve Bayes
  - MultinomialNB
  - BernoulliNB
  - GaussianNB

Predictive

Parameter tuning

- As it could be seen on the image of the last slide, all the models have a parameter grid associated to them, as well as the dataset associated to that model (normalized or non-normalized).
- The parameter grid was used to tune our models with the best parameters possible.
- Through a small python script, we have tested all those parameters combinations on each model on each year.
- At the end, based on the given results, we chose the best parameter combination for the best model and used it to predict season 11.

Predictive

Understanding algorithm behavior

- When comparing decision tree algorithms with random forest, it is expected that random forest performs better since it is an improvement of the decision tree. The decision tree, in a short way, just splits data based on features. Random forest uses multiple decision trees with random splits. Then it is generated a probability for each case it wants to predict based on the prediction given by each internal decision tree.
- The K-NN and SVM models are based on distances and vectors. To help these models to give us the best results, we have normalized the data as previously mentioned.
- Lastly, the Naïve Bayes uses statistical methods to compute the probability of a given event to happen.
- To our problem, without even perform any test, it is expected that the best results will be probably from Random Forest or Naïve Bayes. This happens because the final dataset has many different scales, like ratios, values in cm for height, "counter" values, for example with the number of points, rebounds, etc, binary features like the *top_coaches,* and we believe that even with the normalization of the dataset, the results from KNN and SVM won't be as good as the Random Forest's or Naïve Bayes's.

Predictive

Training vs Test

- Once we got the final datasets ready (one with 0's for auxiliar rows created on sliding window and the other with the quantile(0.25)), it was time to train some models.

- Since we are dealing with a temporal problem, using cross-validation (CV) is not recommended. Cross-validation splits data in k-folds and uses 1-fold to test and others to train, and then it repeats k times, using a different fold for test in each iteration. This approach does not work, since those folds contain data of multiple years, so, on those CV iterations, we would be training for a season N with information of season N+1, for example.

- To simulate the cross validation, we have created this folds based on the season of data and ensured we are only using data from seasons N - k to train a model that predicts results for season N. k stands for the window size on the previously mentioned sliding window technique.

- Example: For testing season 10 with a window size of 2 we would train it based on seasons 9 and 8

Predictive

Constraints

- The tuning of the models, as it could be seen, did not contain a very large number of hyperparameters. This is mainly due to time constraints.
- Since we are testing all the combinations, adding one more value for one parameter or even add new parameters would result on an exponential increase of execution time.
- On our modeling script, we kept this in mind and allow any possible user to disable the hyperparameterization of the models.
- On the final version of the script, the hyperparameterization could still take 1h long.

Predictive

Performance estimation

- To measure the performance of our models, the following metrics were used:
  - Accuracy
  - Precision
  - Recall
  - AUC
  - Mean Accuracy
  - Standard deviation

Table 1 – results for decision tree classifier

Decision Tree		Accuracy	Precision	Recall	AUC	Params*
Years	6	0.69	0.75	0.75	0.81	Max_depth = None
	7	0.86	0.88	0.88	0.91	Max_depth = 10
	8	0.38	0.5	0.5	0.53	Max_depth = 5
	9	0.43	0.5	0.5	0.52	Max_depth = None
	10	0.54	0.64	0.64	0.51	Max_depth = 15

Mean Accuracy	Std. Deviation
0.58	0.2

*Params = used parameters (hyperparameterization result)

Predictive

Performance estimation

Table 2 – results for random forest

Random Forest		Accuracy	Precision	Recall	AUC	Params*
Years	6	0.69	0.75	0.75	0.88	{'bootstrap': True, 'criterion': 'gini', 'max_depth': 5, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 3, 'n_estimators': 100}
	7	1	1	1	1	{'bootstrap': False, 'criterion': 'entropy', 'max_depth': 5, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 100}
	8	0.54	0.63	0.63	0.55	{'bootstrap': True, 'criterion': 'entropy', 'max_depth': 5, 'max_features': 'sqrt', 'min_samples_leaf': 3, 'min_samples_split': 2, 'n_estimators': 100}
	9	0.86	0.88	0.88	0.92	{'bootstrap': True, 'criterion': 'entropy', 'max_depth': 5, 'max_features': 'sqrt', 'min_samples_leaf': 3, 'min_samples_split': 2, 'n_estimators': 200}
	10	0.39	0.5	0.5	0.55	{'bootstrap': False, 'criterion': 'entropy', 'max_depth': 5, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 50}

Mean Accuracy	Std. Deviation
0.69	0.24

*Params = used parameters (hyperparameterization result)

Predictive

Performance estimation

Table 3 and 4 – Results for SVM and KNN

SVM		Accuracy	Precision	Recall	AUC	Params*
Years	6	0.23	0.38	0.38	0.13	{'C': 0.1, 'kernel': 'rbf'}
	7	0.14	0.25	0.25	0	{'C': 0.1, 'kernel': 'linear'}
	8	0.54	0.63	0.63	0.775	{'C': 10, 'kernel': 'rbf'}
	9	0.29	0.38	0.38	0.29	{'C': 10, 'kernel': 'linear'}
	10	0.54	0.63	0.63	0.5	{'C': 1, 'kernel': 'linear'}

Mean Accuracy
0.35
Std. Deviation
0.18

KNN		Accuracy	Precision	Recall	AUC	Params*
Years	6	0.69	0.75	0.75	0.85	{'n_neighbors': 5}
	7	0.86	0.88	0.88	0.86	{'n_neighbors': 5}
	8	0.69	0.75	0.75	0.64	{'n_neighbors': 5}
	9	0.71	0.75	0.75	0.73	{'n_neighbors': 7}
	10	0.54	0.63	0.63	0.6	{'n_neighbors': 5}

Mean Accuracy
0.70
Std. Deviation
0.11

*Params = used parameters (hyperparameterization result)

Predictive

Performance estimation

Table 5 and 6 – results for MultinomialNB and BernoulliNB

MultinomialNB		Accuracy	Precision	Recall	AUC	Params*
Years	6	0.85	0.88	0.88	0.9	{'alpha': 0.1, 'class_prior': [0.3, 0.7], 'fit_prior': True}
	7	0.86	0.88	0.88	0.94	{'alpha': 0.1, 'class_prior': [0.1, 0.9], 'fit_prior': True}
	8	0.69	0.75	0.75	0.58	{'alpha': 0.1, 'class_prior': None, 'fit_prior': True}
	9	0.86	0.88	0.88	0.94	{'alpha': 2.0, 'class_prior': [0.3, 0.7], 'fit_prior': True}
	10	0.69	0.75	0.75	0.67	{'alpha': 0.1, 'class_prior': None, 'fit_prior': True}

Mean Accuracy
0.79

Std. Deviation
0.09

BernoulliNB		Accuracy	Precision	Recall	AUC	Params*
Years	6	0.85	0.88	0.88	0.89	{'alpha': 0.1}
	7	1	1	1	0.97	{'alpha': 0.1}
	8	0.54	0.63	0.63	0.45	{'alpha': 0.1}
	9	0.71	0.75	0.75	0.71	{'alpha': 0.1}
	10	0.54	0.63	0.63	0.68	{'alpha': 0.5}

Mean Accuracy
0.73

Std. Deviation
0.19

*Params = used parameters (hyperparameterization result)

Predictive

Performance estimation

Table 7 – results for GuassianNB

GaussianNB		Accuracy	Precision	Recall	AUC	Params*
Years	6	0.54	0.63	0.63	0.48	{'var_smoothing': 1e-05}
	7	0.57	0.63	0.63	0.54	{'var_smoothing': 1.0}
	8	0.54	0.63	0.63	0.57	{'var_smoothing': 1e-09}
	9	0.71	0.75	0.75	0.85	{'var_smoothing': 1e-09}
	10	0.54	0.63	0.63	0.5	{'var_smoothing': 1e-07}

Mean Accuracy	Std. Deviation
0.58	0.07

*Params = used parameters (hyperparameterization result)

Predictive

Analysis of results

- Based on the obtained results, we can see, based on the mean average, that the 4 best models are:
  - MultinomialNB (Mean Average = 0.79)
  - BernoulliNB (Mean Average = 0.72)
  - KNN (Mean Average = 0.699)
  - RandomForest (Mean Average = 0.694)

- We can see that almost everything that was expected to happen actually happened, apart from the unexpected performance of KNN.
- The fact that KNN performed better than RandomForest is one good indicator that the work that we had to normalize our dataset actually worked better than the expected.
- Regarding the MultinomialNB, it's important to notice the low standard deviation (0.09) that proves a certain consistency on results. This means that we are very likely to achieve good results predicting season 11 with this model.

Predictive

Model improvement and feature importance

- A great effort was put into making sure that we had chosen the best model possible, considered the type of data given and the project's goal. We tested several models, such as the Decision Tree Classifier, the Random Forest Classifier, the Support Vector Machine, the Multinomial Naïve Bayes, K-Nearest Neighbors Classifier, the Bernoulli Naïve Bayes and the Gaussian Naïve Bayes. For each of this models, we changed their parameters multiple times, because we wanted to find out the model+parameters combination that gave us the best accuracy value. In the end, it turned out to be the Multinomial Naïve Bayes the one to do exactly that.
- We also made a small script to see if dropping certain features would actually increase our models' accuracy. We did this by comparing each model's accuracy and AUC values for each year with the ones obtained by dropping each feature. If dropping the feature increased the values (the amount of values increased were bigger that the ones that decreased), the dataset would save that change and, in the next feature dropped and tested, that first feature dropped wouldn't be present already. We understood that all features that our final dataset had were crucial, because our script concluded that not a single feature should be removed.
- Overall, the models' performance, after hyperparameterization, was slightly improved. However, it did not improve the performance of the Multinomial Naïve Bayes, which is our best model.

Project management

Plan and tools

- During the semester, we increased, week by week, the work developed for this project. We started by ensuring that we had a good understanding of what was intended, as well as the datasets we would have to deal with, mainly creating graphs to understand the data and the presence of outliers.
- There was, over the weeks, a balanced development of the project.
- The distribution of the group's tasks was done using the management tool Trello (a web-based, kanban-style, list-making application).

Tools

Database, analytics and tools used

- The programming language used was Python, since it allowed the utilization of the library Python Pandas, which was crucial for this project because it is a powerful and flexible library that provides extensive tools for data manipulation and analysis in a tabular format.
- The entire Project was developed in Jupiter Notebooks.
- There was no specific database used, since all the data needed was given at the beggining, in the .csv format.

Conclusion

- To sum up, throughout this semester we learned a lot about different Machine Learning techniques, namely which steps should be taken (Business Understanding, Data Understanding,...) and how to do them.
- Since our model has an accuracy around 80%, we can safely say that we concluded our project with success.