WNBA Playoff Qualifier Prediction Model

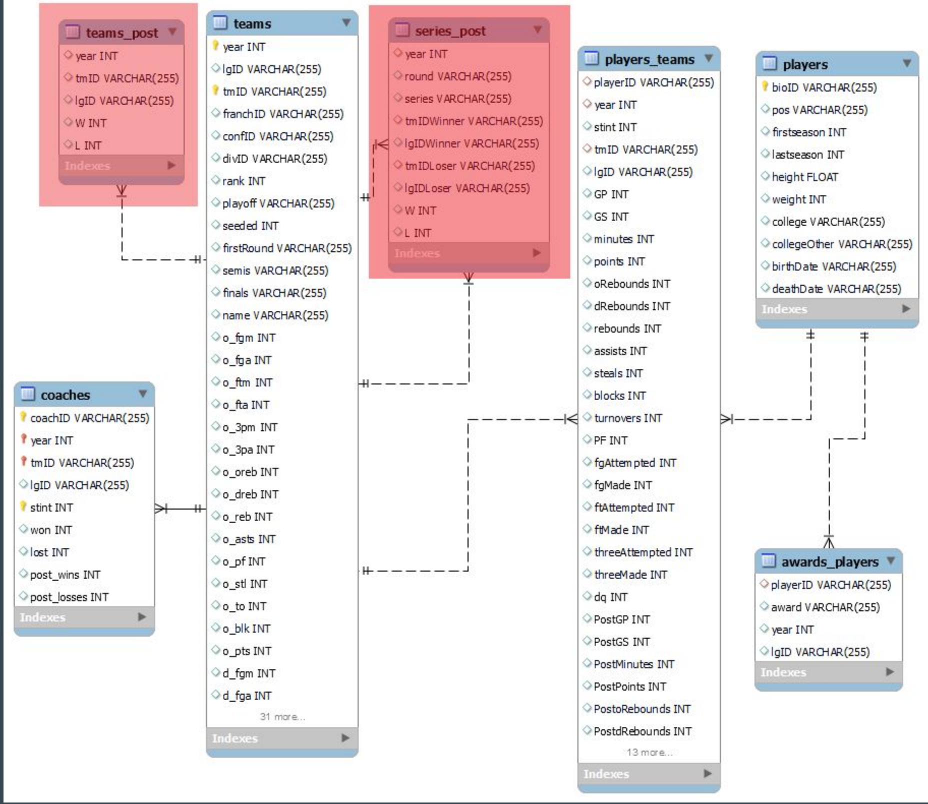
Work done by:

- Afonso Osório - up202108700 - 0.33
- Tiago Cruz - up202108810 - 0.33
- Tomás Xavier - up2021008759 - 0.33

Domain Description

In the WNBA, the regular season involves all teams competing against each other to accumulate wins. The primary goal for each team during this phase is to secure a spot in the playoffs by achieving one of the highest win records and consequently a spot among the top 8 in the standings. Only the top teams of the season make it to the playoffs and then are seeded based on their regular season records, with the top seeded teams winning home-court advantage in the postseason.

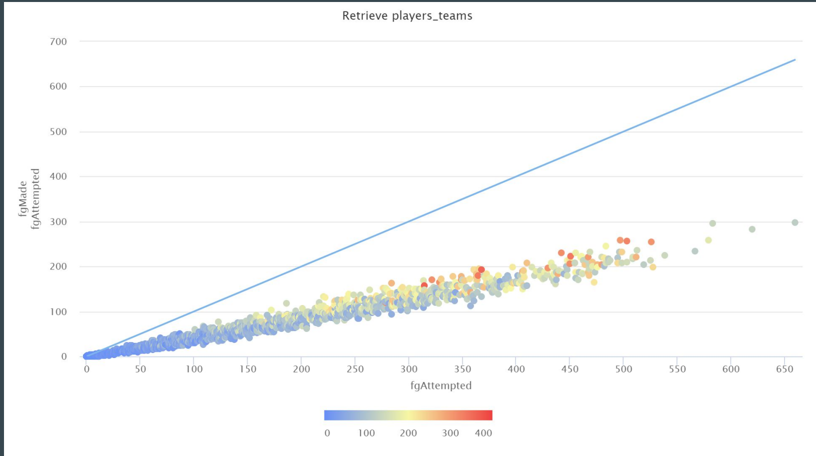
Along the years, the WNBA also suffered a loss in the amount of teams playing.

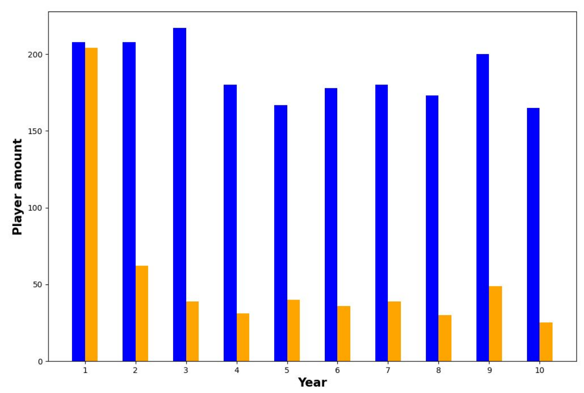


Exploratory Data Analysis

During our exploratory data analysis we noticed the following:

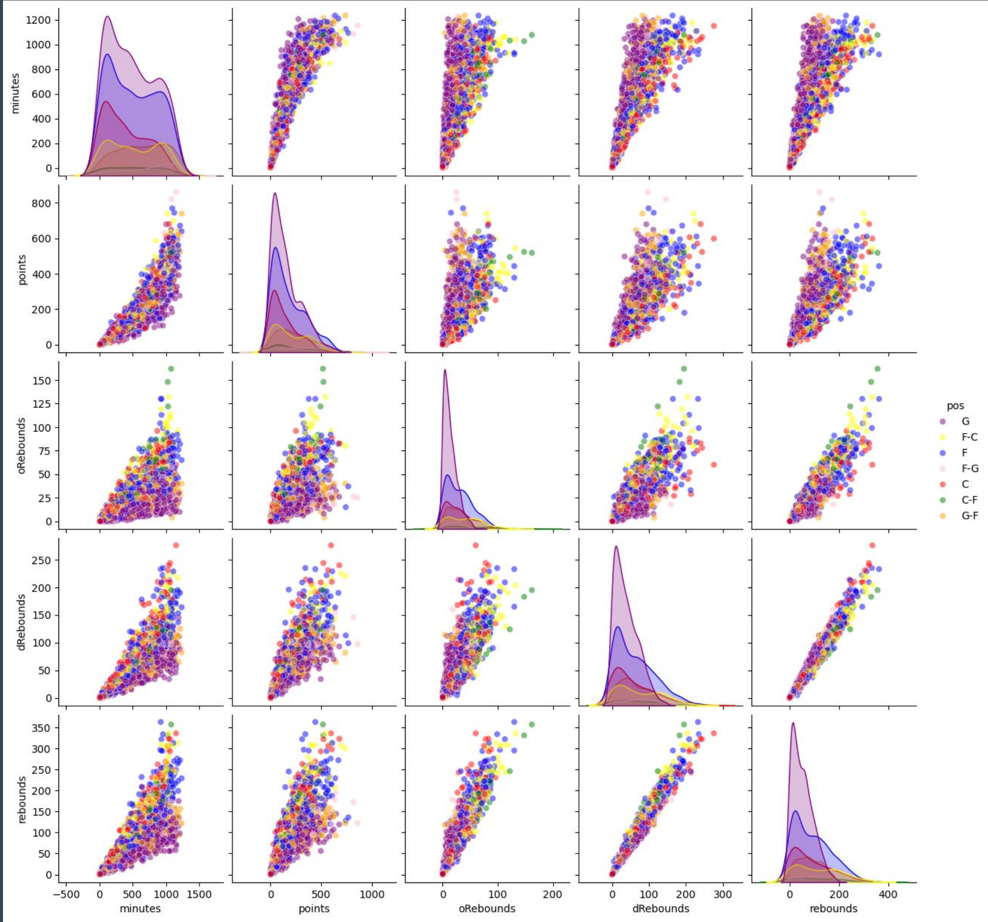
- While there are more teams in the ‘Y’ class than in the ‘N’ class, the imbalance isn’t large
- Circa 30 new players each year (aside from year 1)
- The number of teams isn’t constant over the years
- In cases of metrics for which there are both columns for attempts and successes, since the attempts always have a larger value, we assumed that they include the successes.





Exploratory Data Analysis (cont.)

- We verified that, for each year, 4 teams of each conference are qualified for the playoffs.
- We found that, while different player positions present different values for their stats, the distribution is similar. This, along with the fact that each team will always have players in different positions, means that the differentiation of players through their position shouldn't be a big concern.



Problem Definition

The goal of this problem is to predict future results in the WNBA league. Based on data about the performance of different teams, their players and their coaches, spanning a decade, we want to predict which teams will make it to the playoffs next year.

This is a binary classification problem, and for each team, our model will output the probability that it will make the playoffs. The evaluation metric corresponds to the sum of the absolute difference between said probabilities and the real value (0 or 1). A lower value corresponds to a better prediction.

Data Cleaning

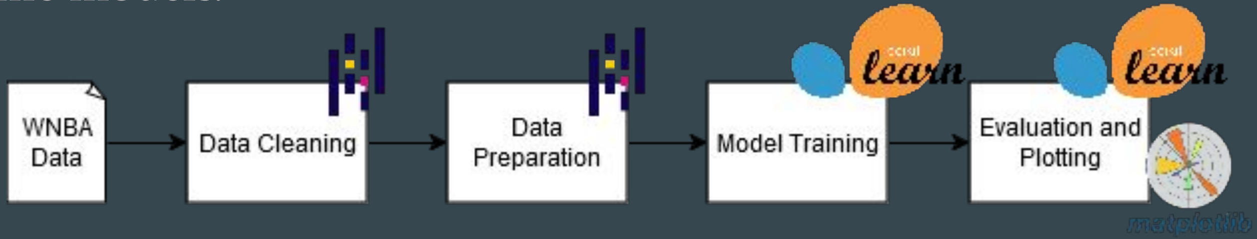
- Multiple attributes had only one or no value, so they were not included in the final dataset (ex: divID).
- We did not identify any missing values regarding the players' performance. However, missing values are inserted in the final dataset due to the way we perform our calculations. We replaced those values with 0. This happens for the team-based values in a new team, and for the "Post" values for a player/coach who hasn't yet been to playoffs in any previous season. For this last case in particular, we think it makes sense to replace them with 0, since it expresses their lack of experience in playoffs.
- A player's stats can be misleading when they play for a short period of time in a given year (in extreme cases, this may give the impression that a player can, for example, score 3 points per minute). To combat this, instead of weighing stats only by minutes played, we divide them by the nº of games played (less volatile than minutes) and we multiply by minutes played.

Data Preparation

- Before training our models, we transformed the data in order to aggregate information from multiple entities: Players, Teams, Awards, and Coaches.
- To condense past information into each row of data, we used exponentially-weighted expanding windows in order to calculate the average of multiple attributes up to the row's year, giving more importance to recent observations. As such, categorical features (such as the players' college) were not used. During the competition, we implemented a grid search (and later a more sophisticated search using Optuna) over those windows' decay values to find optimal values.
- We also replaced some features with ratios that we think are more informative with regards to the quality of a team/player (win/loss ratio instead of wins and losses, for example).

Experimental Setup

- In order to train our models, we split the data in a way that allows us to make predictions for a given year. The chosen classifier is used along with a grid search (evaluated with accuracy) to be trained on all previous data. It is then tested on the year's teams, and the problem description evaluation metric is used to display a result.
- Before fitting a model, our data is passed through a StandardScaler so that we can apply PCA. Beyond reducing training time, the use of PCA improved performance for some models.

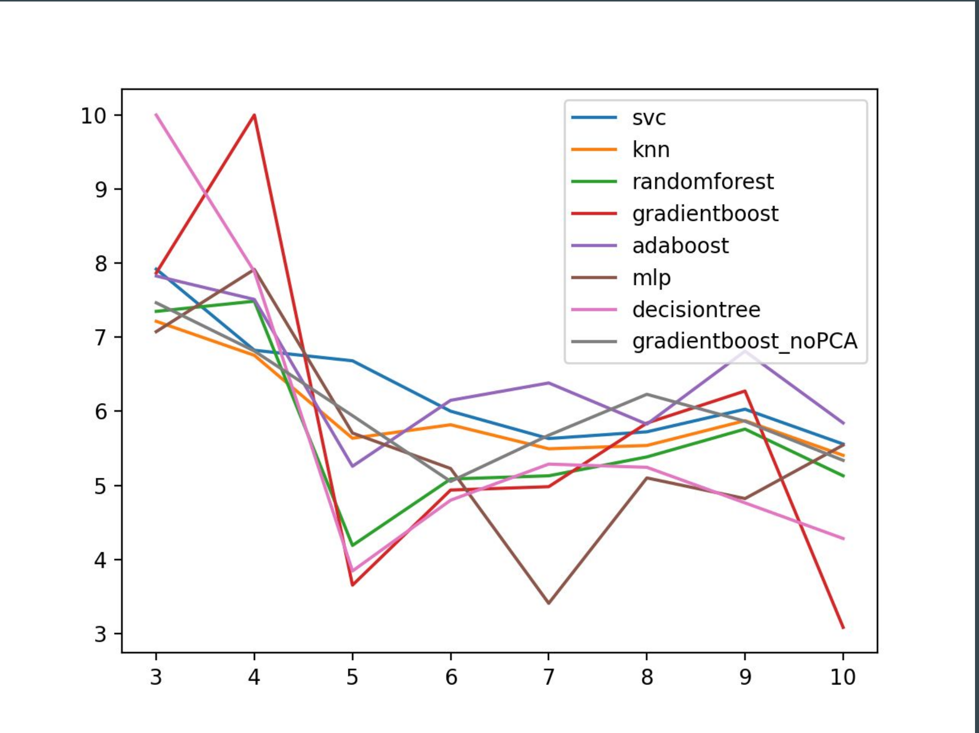


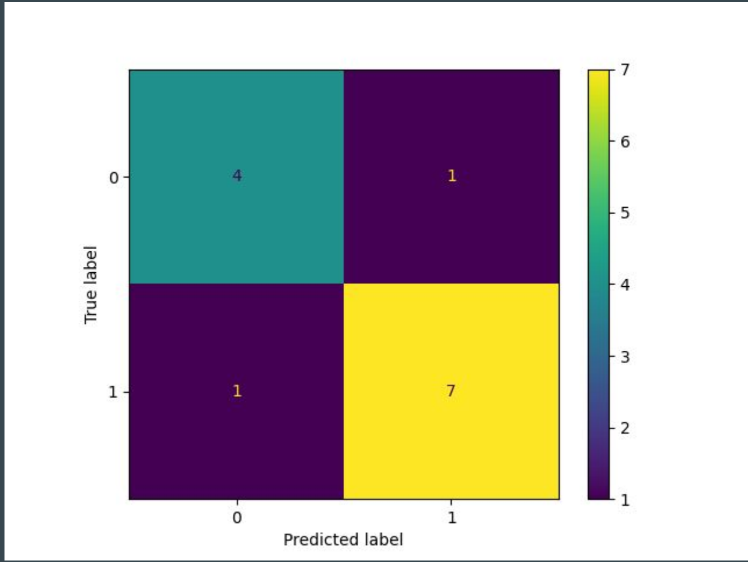
Results

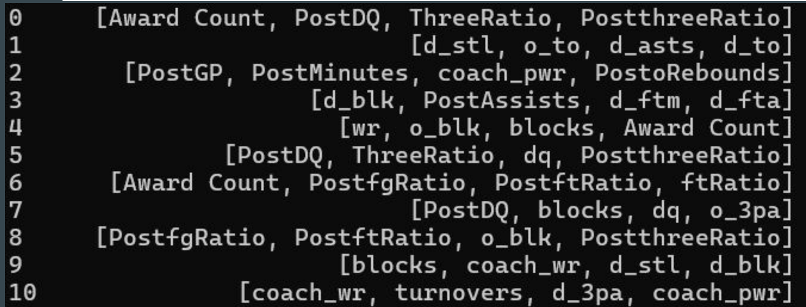
For predictions based on year 10 (the one that includes the most data), Gradient Boosting yielded the best results. Here, we can find the confusion matrix corresponding to that prediction, and the four most important features for each principal component.

Closer look at the results

Feature Importance for Gradient Boosting







```
0     [Award Count, PostDQ, ThreeRatio, PostthreeRatio]
1                          [d_stl, o_to, d_asts, d_to]
2       [PostGP, PostMinutes, coach_pwr, PostoRebounds]
3                     [d_blk, PostAssists, d_ftm, d_fta]
4                      [wr, o_blk, blocks, Award Count]
5                [PostDQ, ThreeRatio, dq, PostthreeRatio]
6       [Award Count, PostfgRatio, PostftRatio, ftRatio]
7                           [PostDQ, blocks, dq, o_3pa]
8     [PostfgRatio, PostftRatio, o_blk, PostthreeRatio]
9                      [blocks, coach_wr, d_stl, d_blk]
10                [coach_wr, turnovers, d_3pa, coach_pwr]
```

Results (cont.)

- We used decision trees so that we could have a better look at how a "simple" white-box model is interpreting our data. These trees were trained without PCA, so that we can more clearly see which features were used in each used. As we can see, as we add data from more and more years, the decision tree grows deeper and more complex since the model has more room to infer meaningful patterns in the data.
- In order to ensure that the perceived difference between the results of our Gradient Boosting model and our baseline (Gradient Boosting but with simpler data) when predicting year 10 was not due to chance, we used a t-test to confirm that the difference is statistically significant with 95% confidence, using 15 samples for each model.

Competition Modifications

After the intermediate presentation, we made some improvements to our model in order to try to achieve better results. During the competition, our focus was mainly in fine-tuning results for the XGBoost classifier.

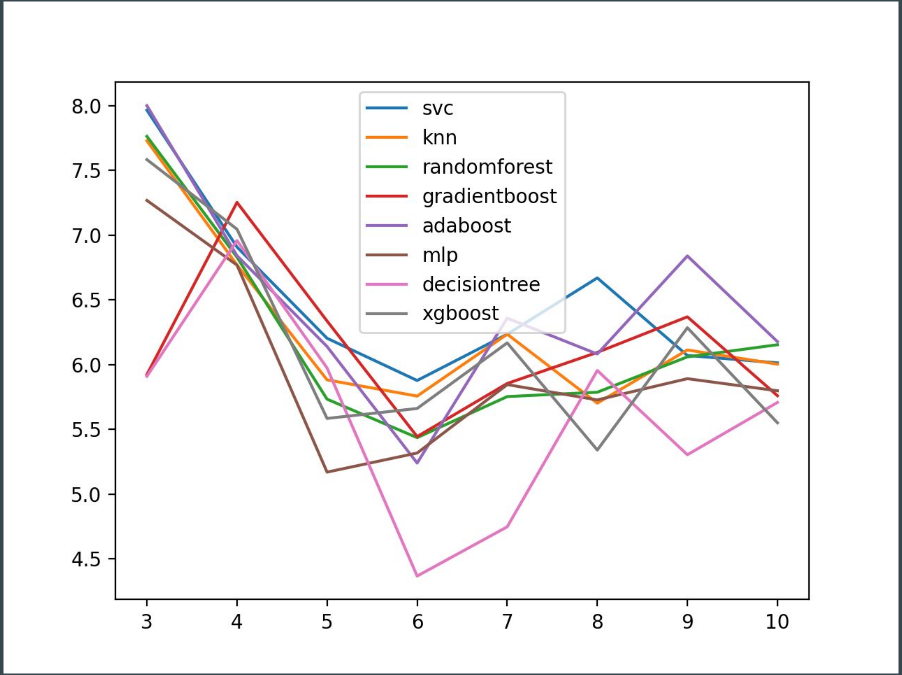
Firstly, we made some slight changes in the way we prepare our data in order to be more consistent with the meaning of the values in the initial domain. A change was made on the way we ordered players and coaches before using our expanding window average, where instead of ordering only by year, we added a second sorting criteria with stint. Despite yielding worse results, we believe it was needed so that the final dataset is consistent with the domain. As such, our top result before this modification may have been partially due to luck in the way the data was arranged internally by the libraries.

Other than this, our changes throughout the competition were:

- Day 2: No changes (slight differences due to model randomness).
- Day 3: Implementation of a “grid search” to find decay values for the exponentially-weighted expanding windows.
- Day 4: Prediction post-processing: set the four highest predictions per conference to 1, and the others to 0. Use of Optuna for hyperparameter tuning (including exponential window decays).
- Day 5: No changes.

Competition Modifications (cont.)

Using the optimal parameters, here are the final comparison between all results



As we can see in the confusion matrix corresponding to year 10's prediction using XGBoost, there are many false positives, meaning too many predictions have values that are higher than what we want. This was a major motivation for our post-processing introduced in day 4 of the competition: by forcing the top ones to be 1, and the rest to 0, we ensure that the score won't be harmed by the bottom teams having high predictions, despite being lower than the top ones.

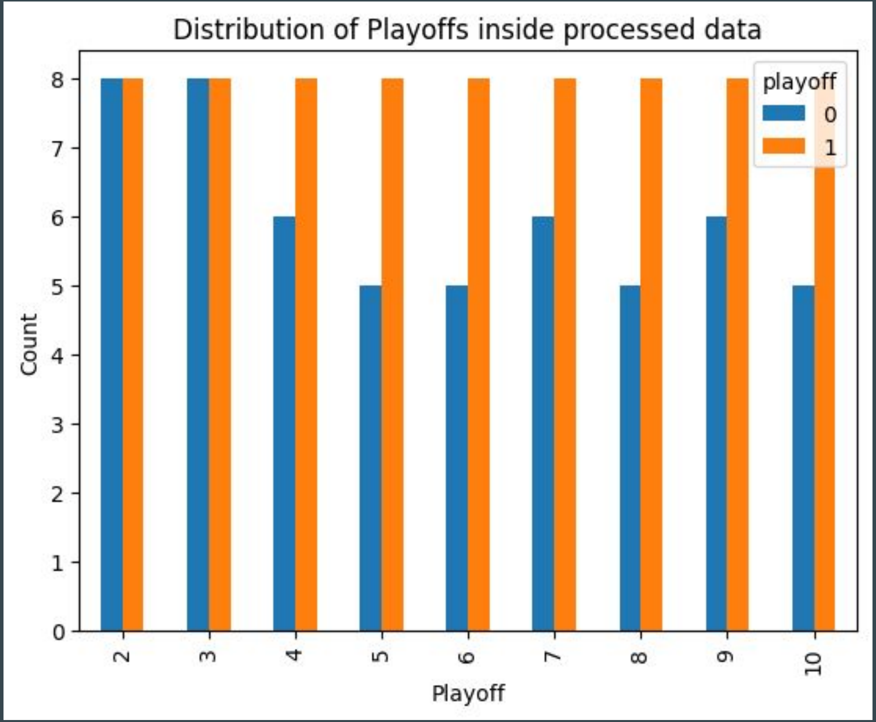
In the last two days of the competition, we used Optuna to find new hyperparameter values (and decays for the exponentially-weighted windows) for XGBoost, based on all of our data. These hyperparameters were then applied before making our final predictions.

XGBoost Results

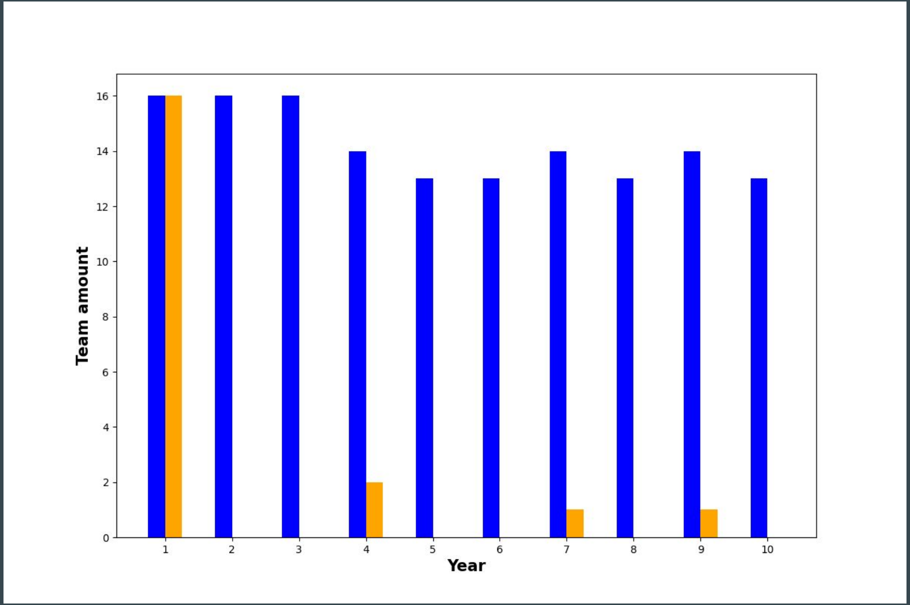
Conclusions, Limitations and Future Work

- With our setup, we are able to test different classifiers and evaluate their performance by using a well-defined pipeline, from data preparation to evaluation.
- While our simple replacement of missing values doesn't seem to be causing major issues with the models, a better strategy could potentially yield better results.
- The model is limited if a large amount of new players is introduced. However, the trend in our training data shows that this isn't likely to be a big problem, as the number of new players each year is fairly consistent.
- The post-processing approach of turning the predictions into 1s and 0s is somewhat "all or nothing". However, it served us well in this case, getting only two teams wrong in the competition (we can deduce this due to us being placed between the top 10% and 25% of groups, with a score between 2.00 and 3.28; since our approach can only lead to values such as 0, 2, 4,..., our score must be 2.00).
- During the competition, more models could have been tried out in more depth, but due to the limited number of submissions and time constraints, we chose to explore what seemed to be the most promising model in depth.

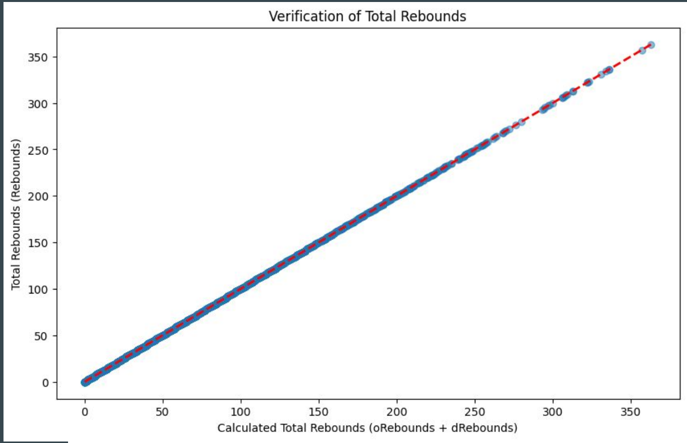
Annexes

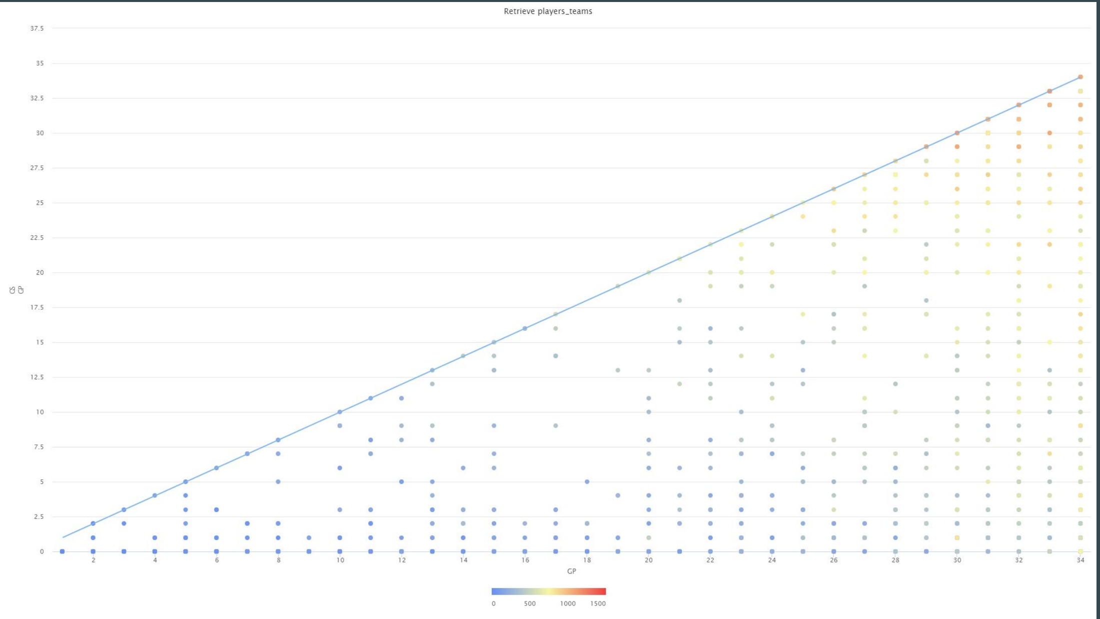


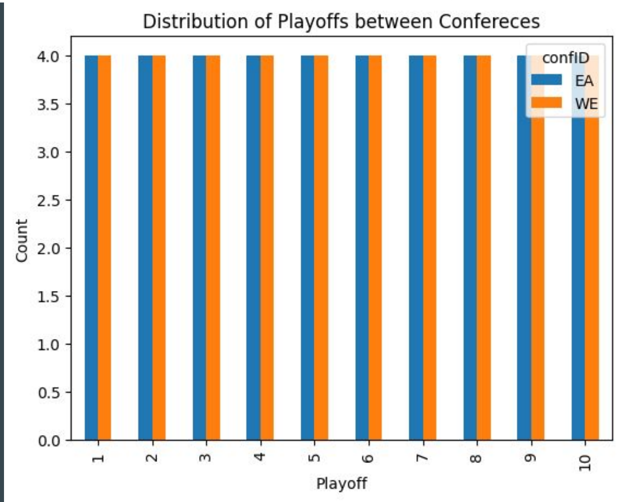
- Amount of teams over the years (blue: total teams, orange: new teams)



- Additional data verification plots

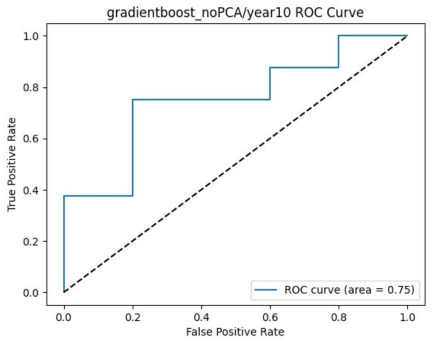


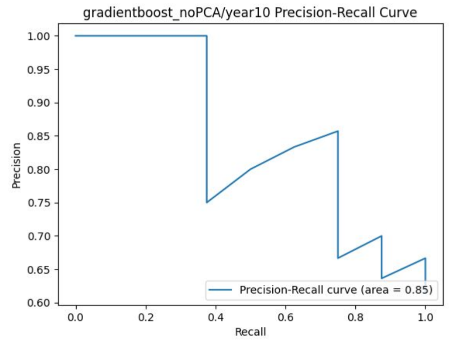


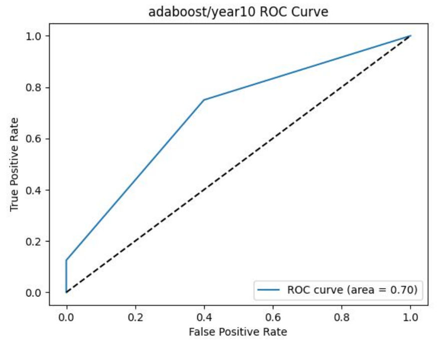


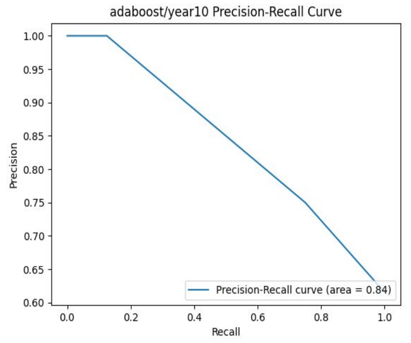
- Results for the multiple classifiers, according to the provided metric (pre-competition)

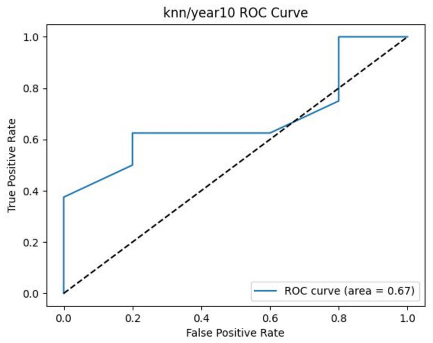
Year	SVC	KNN	RF	AdaB	MLP	DTree	GB	GB noPCA	GB baseline
3	7.919	7.215	7.349	7.826	7.075	10.000	**7.867**	7.465	7.727
4	6.824	6.756	7.486	7.510	7.915	7.886	**4.000**	6.813	8.227
5	6.682	5.636	4.189	5.258	5.707	3.844	**3.653**	5.942	5.899
6	6.001	5.818	5.086	6.148	5.228	4.800	**4.937**	5.053	5.732
7	5.632	5.494	5.129	6.383	3.408	5.287	**4.982**	5.677	4.110
8	5.723	5.538	5.384	5.826	5.101	5.244	**5.841**	6.229	3.985
9	6.028	5.873	5.760	6.811	4.822	4.764	**6.274**	5.866	5.531
10	5.558	5.405	5.129	5.843	5.544	4.283	**3.0836**	5.338	6.979

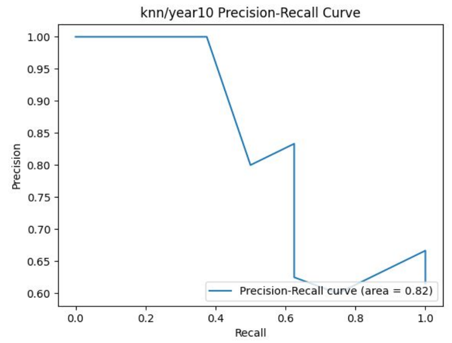

gradientboost_noPCA/year10 ROC Curve
True Positive Rate
False Positive Rate
ROC curve (area = 0.75)

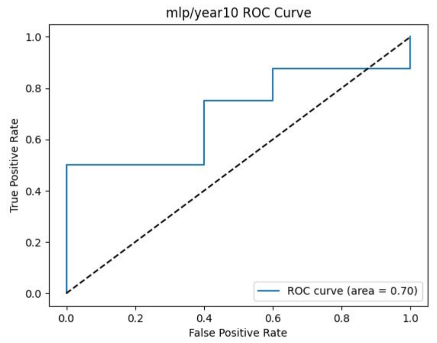

gradientboost_noPCA/year10 Precision-Recall Curve
Precision
Recall
Precision-Recall curve (area = 0.85)

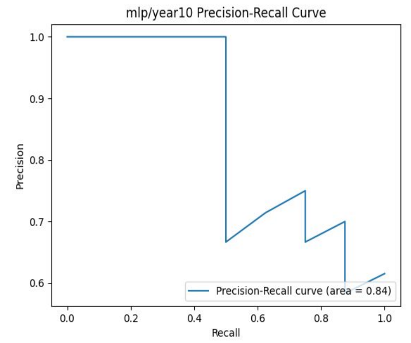

adaboost/year10 ROC Curve
True Positive Rate
False Positive Rate
ROC curve (area = 0.70)

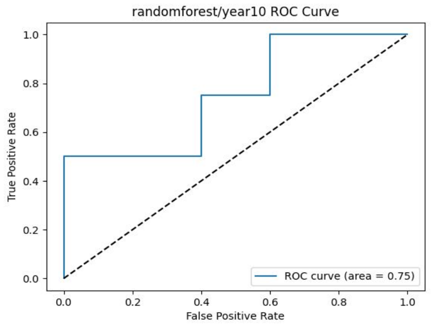

adaboost/year10 Precision-Recall Curve
Precision
Recall
Precision-Recall curve (area = 0.84)

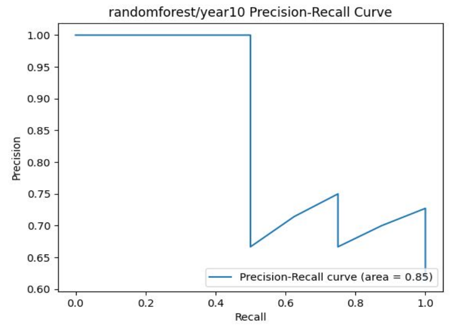

knn/year10 ROC Curve
True Positive Rate
False Positive Rate
ROC curve (area = 0.67)

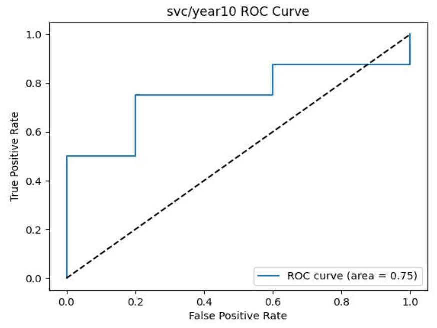

knn/year10 Precision-Recall Curve
Precision
Recall
Precision-Recall curve (area = 0.82)

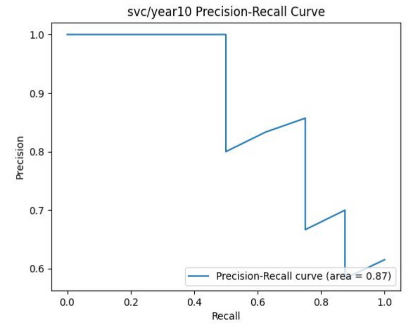

mlp/year10 ROC Curve
True Positive Rate
False Positive Rate
ROC curve (area = 0.70)


mlp/year10 Precision-Recall Curve
Precision
Recall
Precision-Recall curve (area = 0.84)

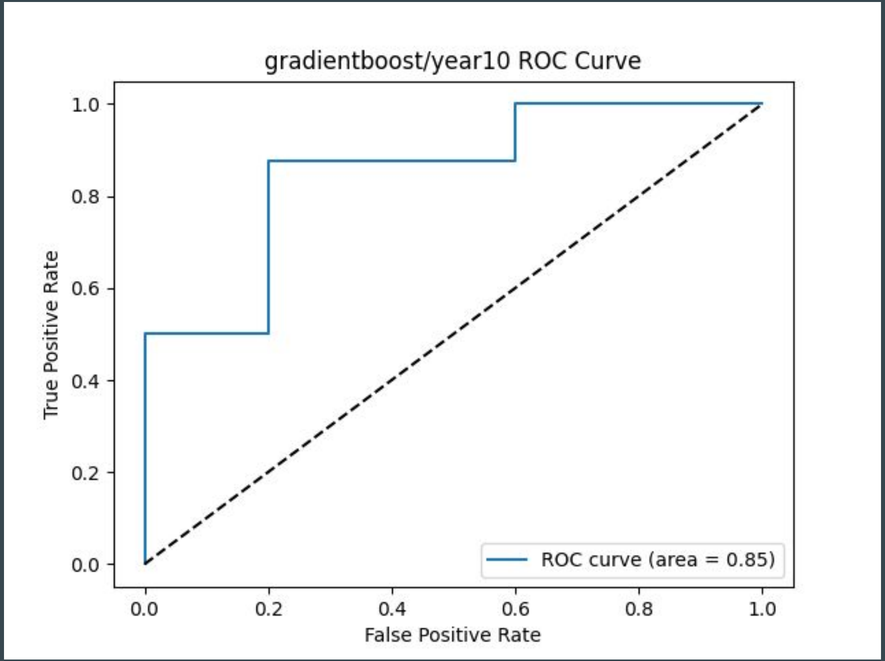

randomforest/year10 ROC Curve
True Positive Rate
False Positive Rate
ROC curve (area = 0.75)

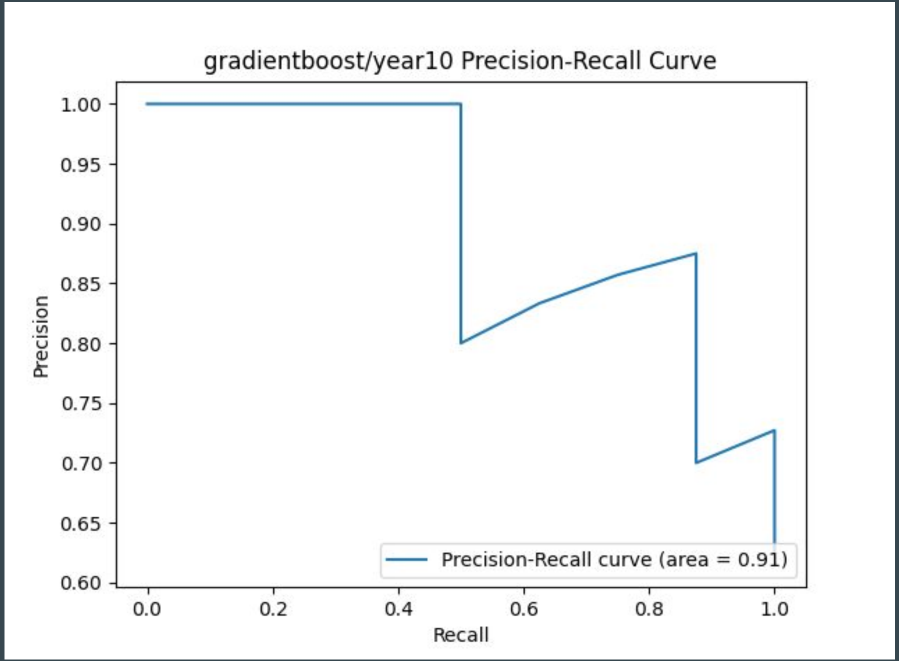

randomforest/year10 Precision-Recall Curve
Precision
Recall
Precision-Recall curve (area = 0.85)


svc/year10 ROC Curve
True Positive Rate
False Positive Rate
ROC curve (area = 0.75)

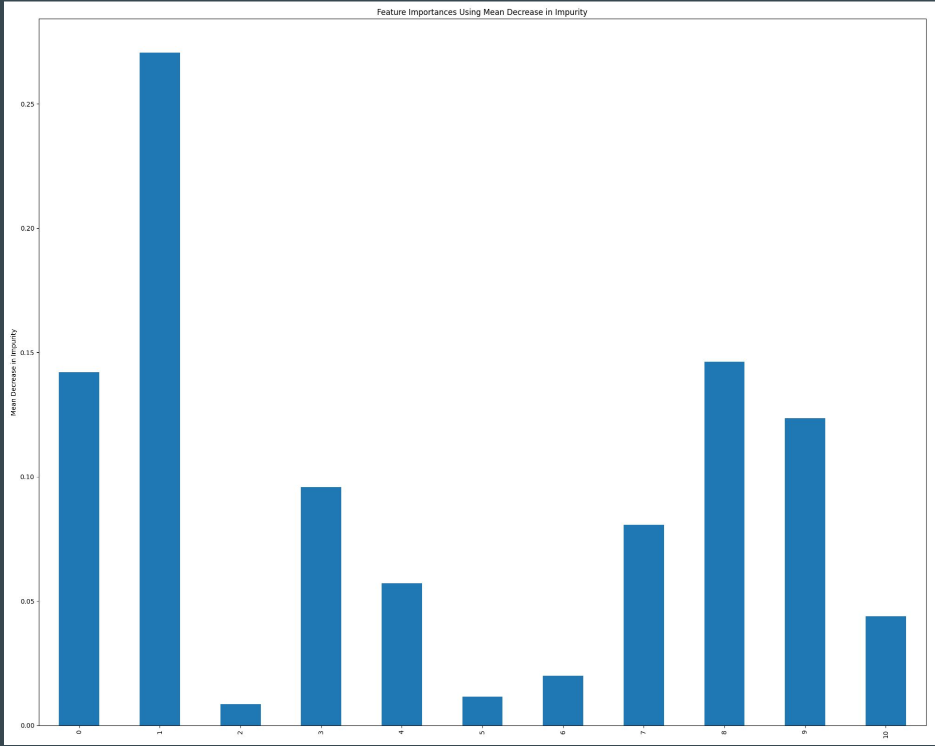

svc/year10 Precision-Recall Curve
Precision
Recall
Precision-Recall curve (area = 0.87)

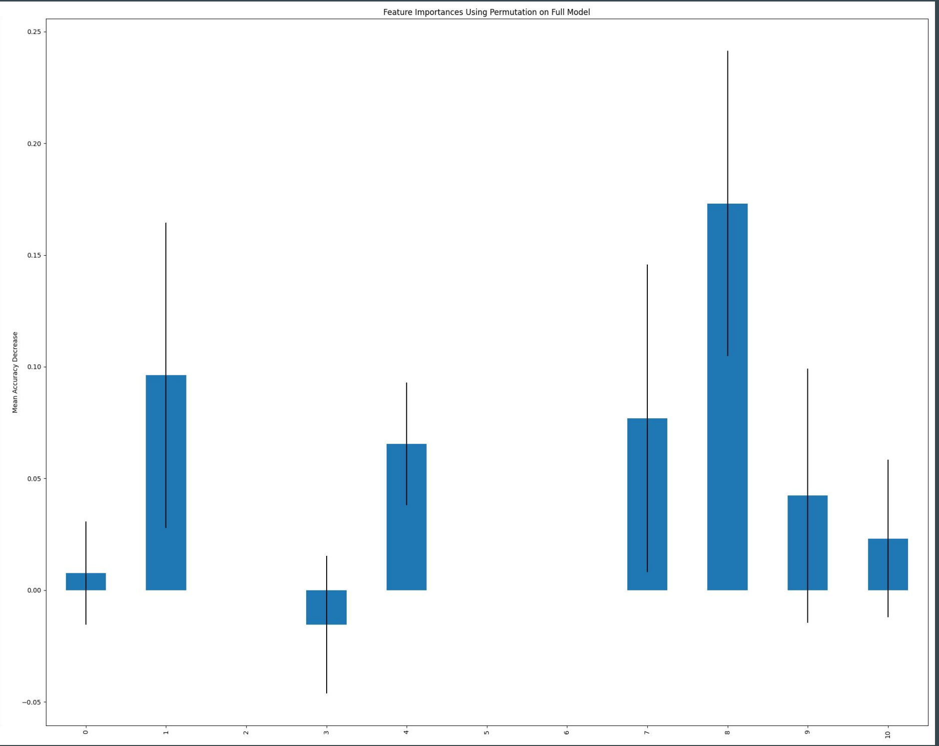
- Gradient boosting results (pre-competition)





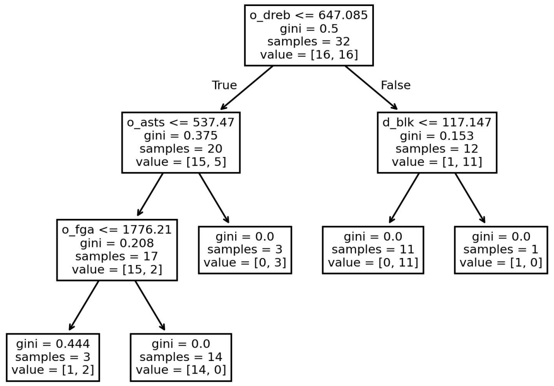
- Gradient boosting feature importances (pre-competition)

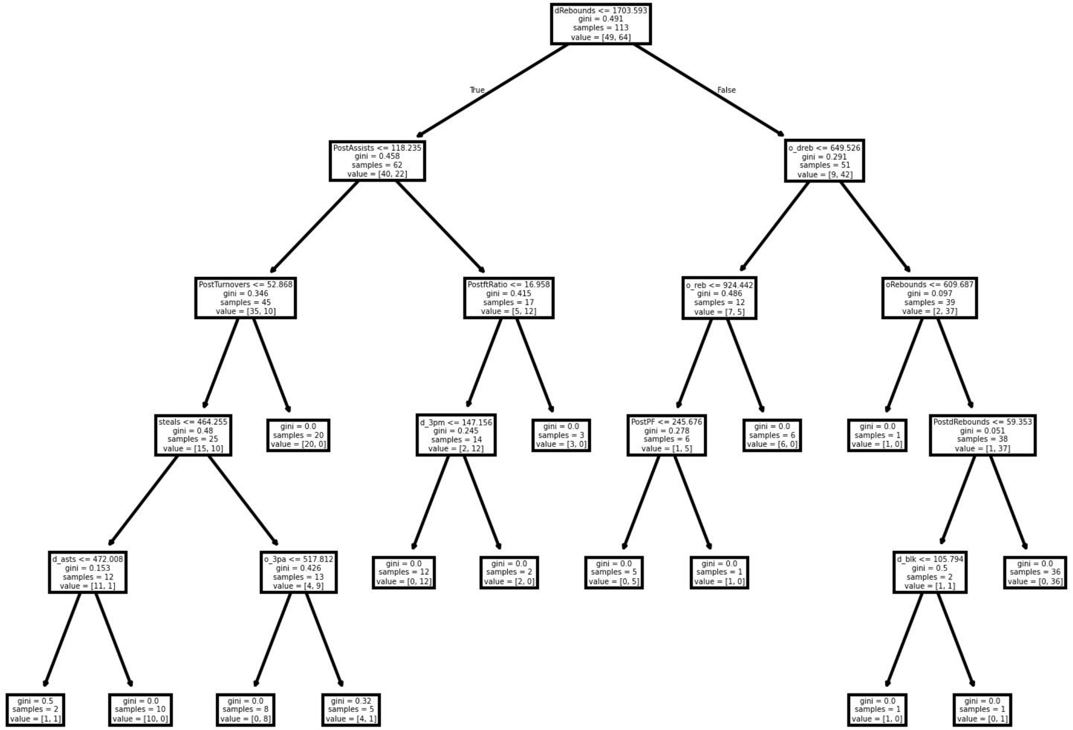




Year 10 Decision Tree

Year 4 Decision Tree





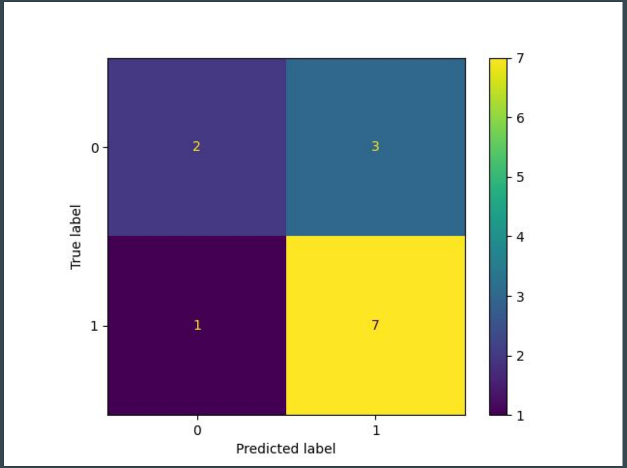
Comparison between Gradient Boost and Baseline model (year 10) (pre-competition)

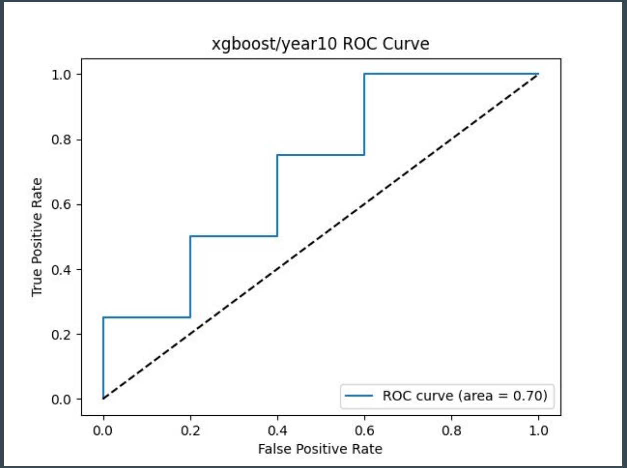
Gradient Boost	Baseline
3.084	6.979
3.178	7.024
3.186	6.598
3.198	6.784
3.138	6.597
3.085	6.174
3.038	7.019
3.312	6.021

Gradient Boost	Baseline
3.199	6.027
3.082	6.214
3.269	6.661
3.081	6.319
3.177	6.547
3.034	7.028
3.273	7.117
3.052	5.985

- XGBoost results (during competition)

Year	XGBoost
3	7.584434211
4	7.046708345
5	5.584490895
6	5.661508292
7	6.168384403
8	5.340533972
9	6.285372168
10	5.550856233


True label
Predicted label


xgboost/year10 ROC Curve
True Positive Rate
False Positive Rate
ROC curve (area = 0.70)