2023

Machine Learning - FEUP

G54

João Alves - up202007614 (Contribution: 33%)

Marco André - up202004891 (Contribution: 33%)

Ricardo de Matos - up202007962 (Contribution: 33%)

Introduction

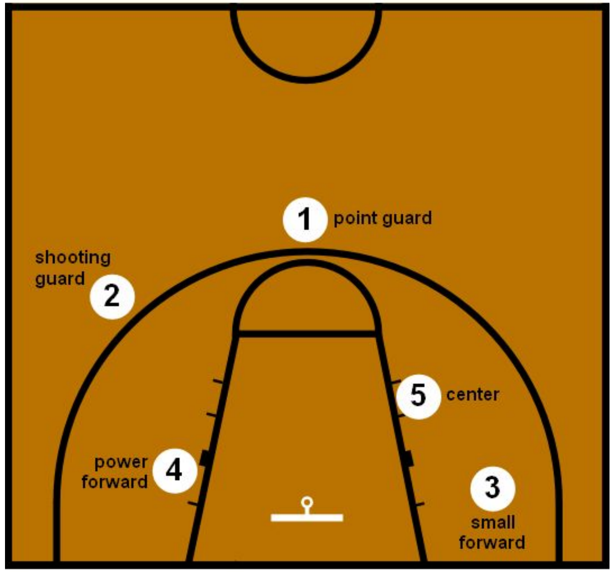
- The Women's National Basketball Association (**WNBA**) is the USA's women's basketball league.
- The recent **growth of women's sports on a global scale** has brought demand for sophisticated data-driven approaches for statistical analysis and performance calculations.
- This **machine learning project** aims to contribute to this landscape by leveraging learning models to **predict WNBA playoff qualification**, providing valuable insights about team performance expectations based on 10 years worth of player, team and coach data.
- This model could be used for **early insights into potential playoff contenders** based on historical data and also help fans more accurately **bet on team's performance**

Domain Description

- **Entities:**
    - Players
    - Teams
    - Coaches
    - Games
    - Awards
    - Metrics (e.g., points, rebounds, assists, etc.)
    - Playoff Qualification (binary: qualified or not qualified)
- **Relationships:**
    - Players belong to Teams
    - Coaches coach Teams
    - Games involve two Teams
    - Metrics are associated with Players, Teams, and Games
    - Teams qualify for the playoffs or not based on scores
- **Qualification Criteria:** 4 best teams of each conference qualify for the playoffs

Problem Definition

- **Problem definition:** Create a predictive model using a decade of historical data to anticipate playoff qualification for basketball teams in the forthcoming season.
- **Objective**: Accurately predict the teams that will classify for the next playoff season
- **Basketball Rules:**
  - The top 4 teams from each of the 2 conferences, determined by their win percentages, advance to the next stage.
  - A roster of 11-12 players.
  - Multiple player positions (both offensive and defensive)

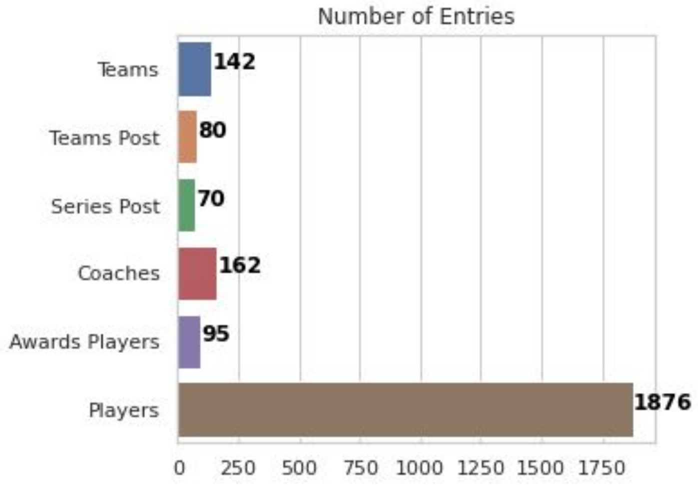


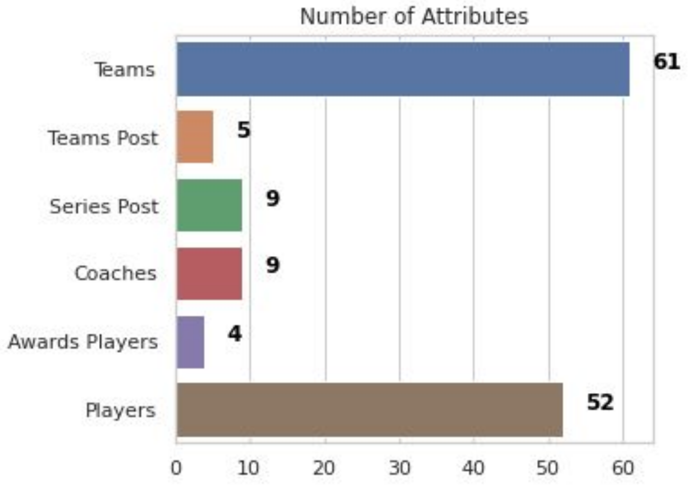
Exploratory Data Analysis

We were provided with 7 CSV files with 10 years worth of data.

After a brief data exploration, we quickly found out that the **Teams** and **Players** CSVs would provide us with the most useful information based on the number of entries, feature relevance and count.

We only present **6 data frames** instead of 7 because we joined the *players.csv* and *players_teams.csv* from the start based on player ID.

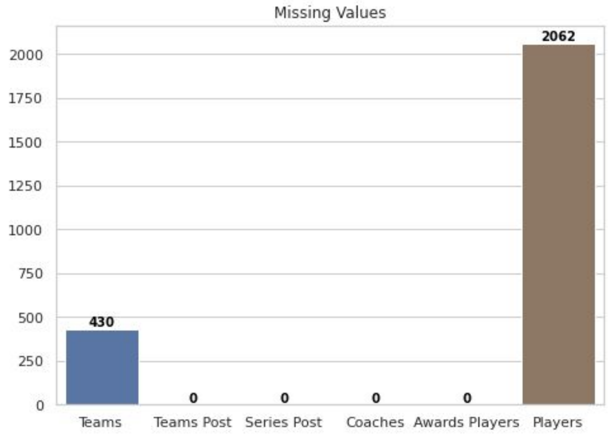


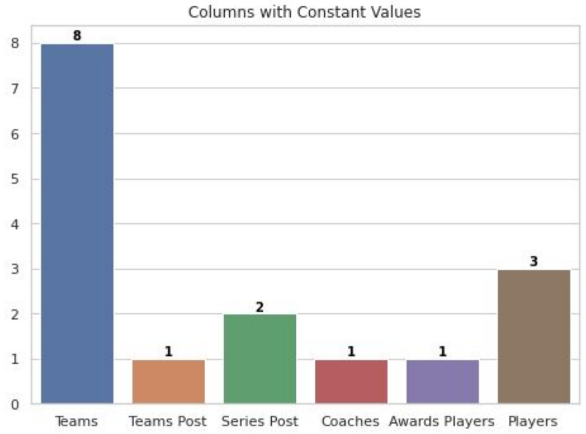


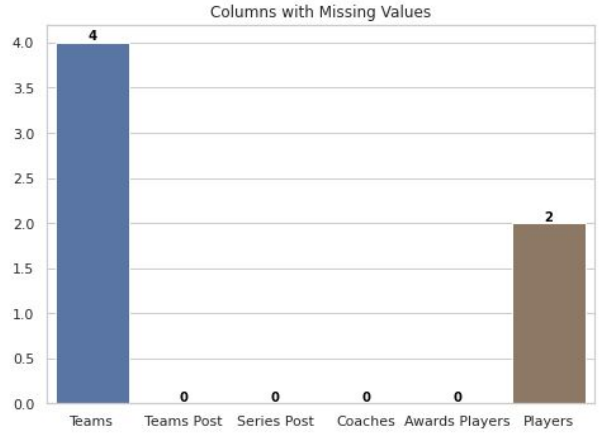
Exploratory Data Analysis

In our analysis, we identified numerous **missing values** in two dataframes, primarily concentrated within a few specific columns.

Additionally, we observed columns with **constant values** (redundant information) dispersed throughout the dataframes. We subsequently addressed these cleaning candidates in our operations.







Exploratory Data Analysis

The team inspected the features of each dataset by using official web sources for nomenclature interpretation in order to get a better grasp of what data we had to work with.

In a nutshell, the CSV datasets that we had access were:

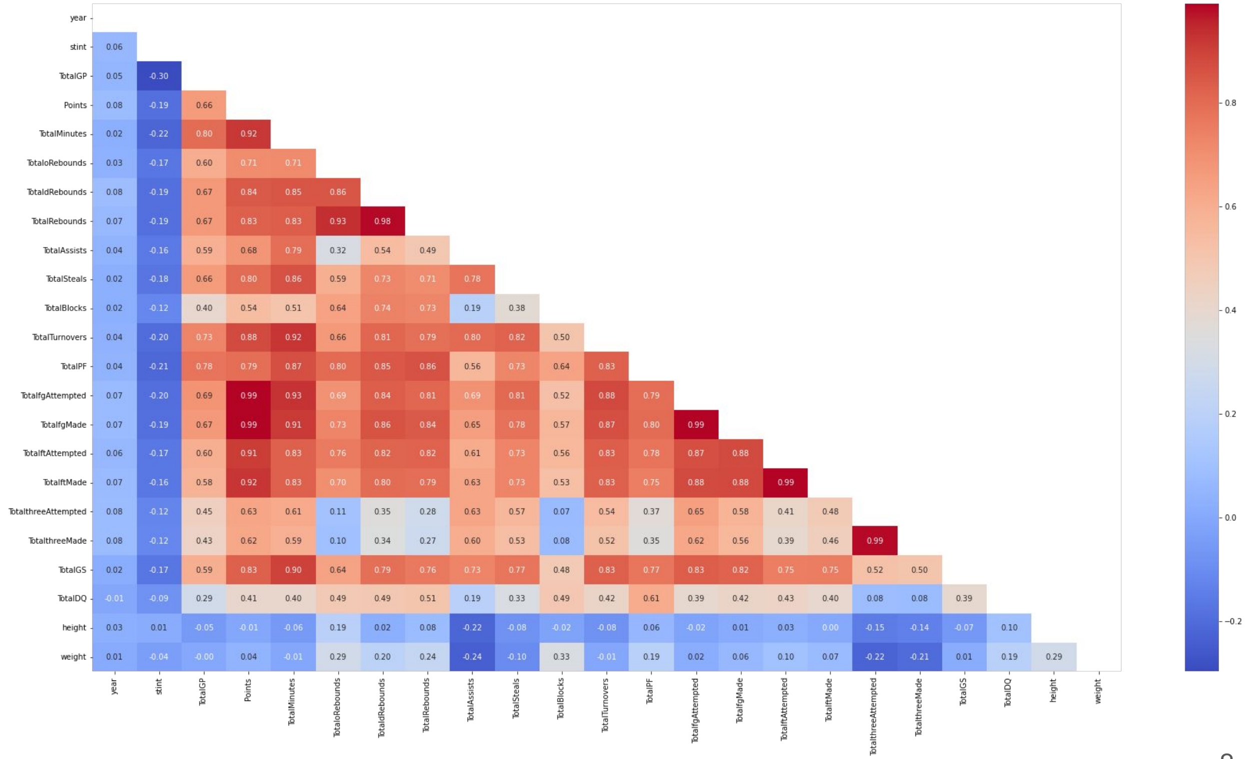
- **Teams -** Information about teams and their metrics
- **Players -** Information about players, metrics and teams they belong to over the years
- **Coaches -** Basic metrics and teams they trained
- **Awards Players -** Includes both player and coach award names and year for the award

The following datasets were not used as the data they contain was not considered relevant and, from the few tests made, didn't influentiate the prediction results: **Series Post** and **Teams Post**.

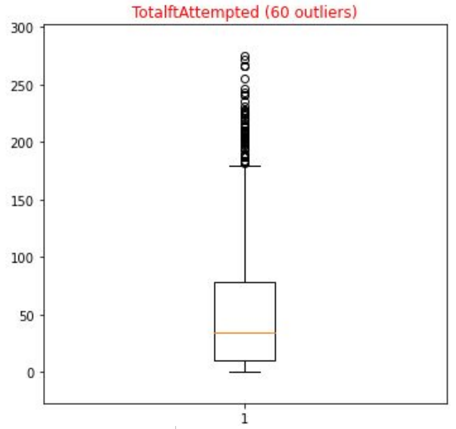
Exploratory Data Analysis

Using **correlation matrix**, we found many attributes (specially in player/team metrics) that, as expected, had very high correlation, and thus can be removed without losing much information.

By pruning redundant features from our dataset, we ensure that our subsequent analyses are more focused and efficient.



Exploratory Data Analysis

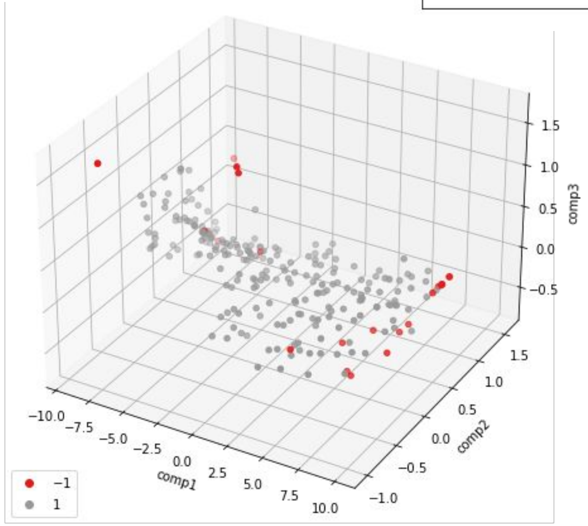


Taking a closer look at **outliers** with **box plots** for multiple features, specifically for players, we quickly found that those were the worst and best players in their specific areas and not data errors.

For example, defensive players will clearly have higher defensive metrics than other offensive players and between offensive players, the best ones will have above the average metrics.

With multiple **scatter plots** like the one to the right where the red dots are the outliers, we find the best/worst overall players (as was successfully manually verified by taking those player IDs and identifying that indeed they had awards for example).

The outliers, in the context being worked on, instead of being removed, should actually be what we should focus on as they are highly determinants of team winning in general.

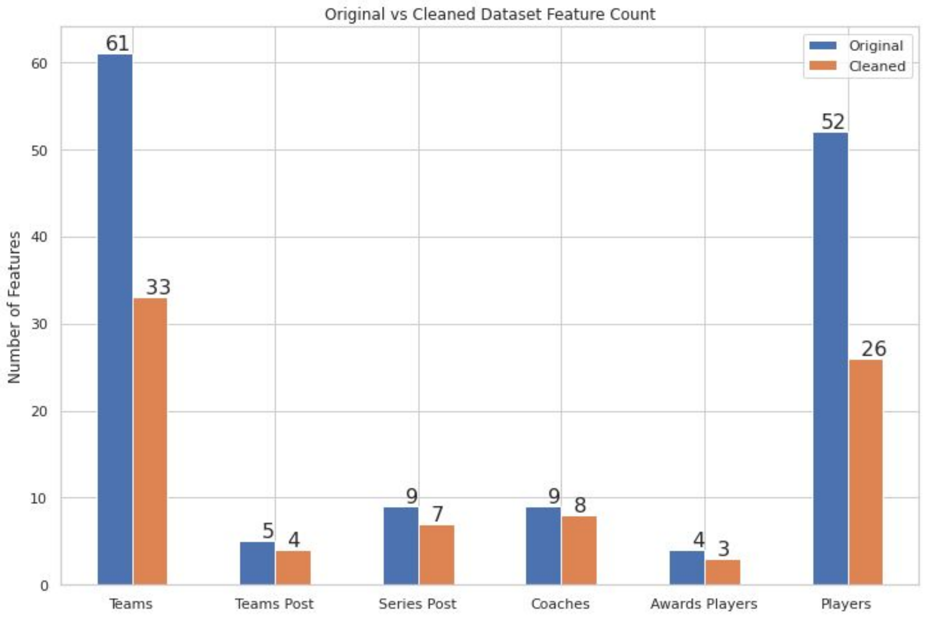


Data Preparation

As earlier stated, there were a lot of cleaning operations to be performed. The main ones were:

- **removing attributes** with **constant values**
- **manual attribute filtering phase** with **PCA** to support our decisions
- **feature engineering** on metrics attributes

Managed to substantially reduce the dataset's dimensionality as seen in the image on the right, facilitating the work with the data.



Data Preparation

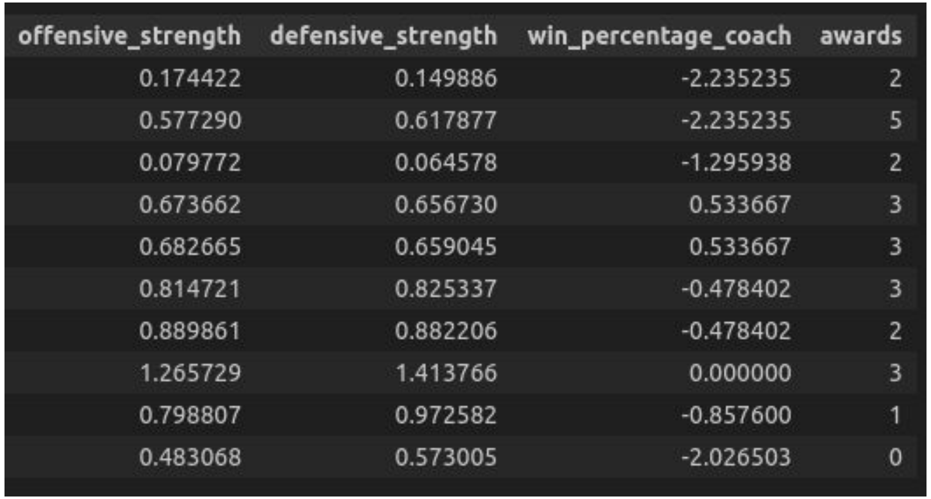
As the work being performed is rather tricky and the targeted goal is influenced by many factors apart from the collected data we have access to, we have made some **educated guesses** and **deduction leaps** based on the more in-depth data analysis we've performed in order to greatly simplify the datasets for an easier and guided data digestion by the models being trained by performing **feature engineering**.

- Teams composed mainly of **above-average players** have a higher likelihood of success in a competition.
- **Coach performance** serves as a key indicator or influence on a team's success, as even talented teams can face difficulties with poor management.
- As most **team metrics** are not known at the beginning of each predicted season, the next greatest thing to feed the models is the players that make up the team metrics from previous years.
- **Awards** for both players and coaches function as **boosts to team metrics** as a whole.

Data Preparation

With all that being said, we **condense all player information** belonging to each team **to just a few standardized player performance variables describing the overall team strength**. **Coach performance** and **awards** also affect the way we do the calculations and may **boost the team ratings**.

This simplistic approach while not perfect, in principle gives us a very good approximation of real life experiences and got us good results.



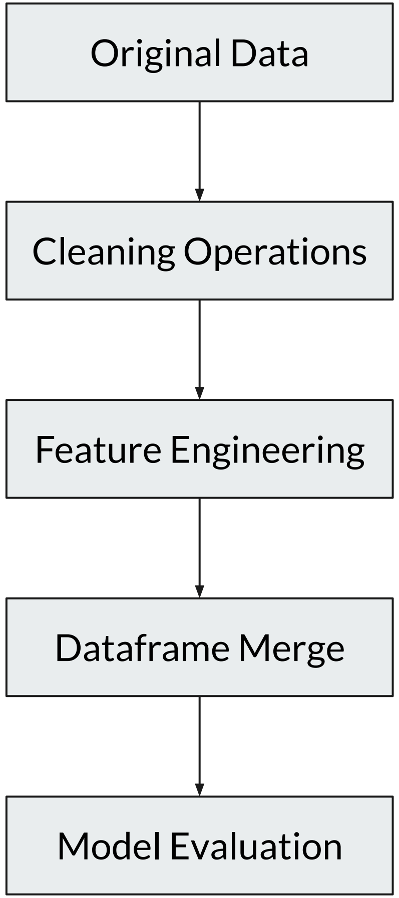
offensive_strength	defensive_strength	win_percentage_coach	awards
0.174422	0.149886	-2.235235	2
0.577290	0.617877	-2.235235	5
0.079772	0.064578	-1.295938	2
0.673662	0.656730	0.533667	3
0.682665	0.659045	0.533667	3
0.814721	0.825337	-0.478402	3
0.889861	0.882206	-0.478402	2
1.265729	1.413766	0.000000	3
0.798807	0.972582	-0.857600	1
0.483068	0.573005	-2.026503	0

Experimental Setup

To streamline and simplify the data processing, we developed a **pipeline** that autonomously cleans the original datasets, performs dataframe merges and is capable of training and applying models to give us predictions results.

We have dedicated **Python Jupyter notebooks** for **statistical analysis** and **predictive modeling**, along with supplementary files with complementary functions.

The pipeline's efficiency grants flexibility to test various combinations and models swiftly, allowing us to access results with ease, allowing for experimentation with multiple different strategies in order to achieve better predictive results.



Evaluated Metrics

To evaluate the performance of our models, we employed various metrics, including **accuracy**, **recall**, **precision**, and **F1 score**.

It's worth noting that we used precision at 4 (**P@4**), for each conference, to evaluate our performance, since it specifically aligns with the nature of our problem as we need to decide which four teams from each conference make it to the playoffs. Unlike regular precision, which measures the overall correctness of positive predictions, P@4 ensures a more contextually relevant evaluation to our particular problem.

Accuracy percentage is always very similar to precision, although consistently smaller due to the variation on the number of teams participating on a given year. It can be misleading because the model can get 7/8 teams correct 2 years straight but still see a change in accuracy due to variation in the team count, translating into slight variations when plotting the data that can wrongly make us think the model variated.

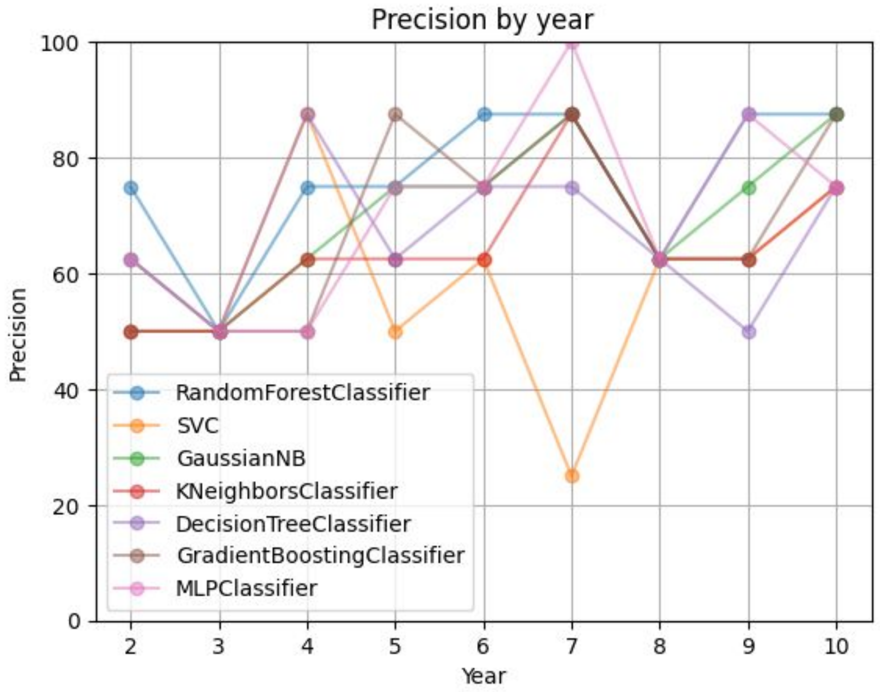
We took close attention to recall to provide further guidance into the precision results we got and used F1 score in order to conciliate both metrics into a single value descriptor. ROC and AUC were also collected and are present in the annexes as another form to evaluate the models.

With all these accounted metrics, we can confidently say whether or not a model is a good one based on a concrete systematic approach.

Results

We **trained models of different types** with different cross validations techniques (e.g.: **sliding** and **expanding window** with and without devaluation of earlier data) and have reached good results without much hyperparameter configuration due to the previous work done on feature engineering.

Note: This results are without hyperparameter tuning. Please refer to annexes for those improved values after the initial presentation.

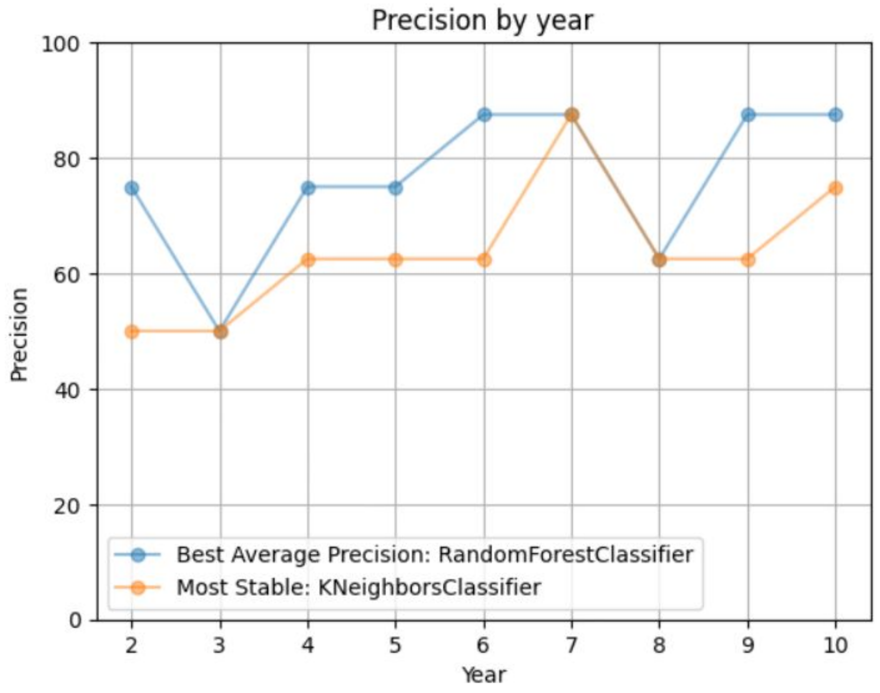


Results

From the previously shown models, we now focus on the two best ones, concerning **biggest overall precision**, but also the most **stable predictions** over time.

As presented to the right, the results achieved are very promising.

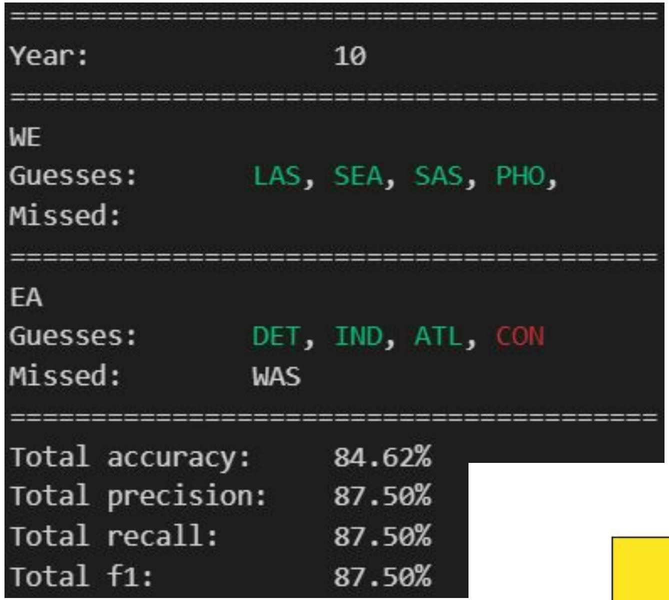
A more stable model could be more interesting for more secure bets on team performance while the other could bring better outcomes but being more risky (even though that in this case, the best model matches or outperformed the stable model every time).



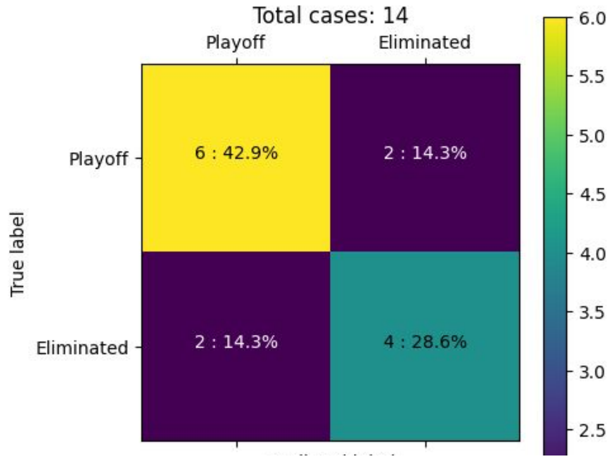
Results

In sports predictions, **precision is often favored** because it's more important to correctly identify the 8 playoff teams even if you miss some, rather than making many false predictions that lead to inaccurate conclusions.

It's better to have a shortlist of highly probable playoff teams, even if you miss a few, than to have a long list that includes many false positives. More information can be found in the annexes.



```
========================================
Year:                   10
========================================
WE
Guesses:       LAS, SEA, SAS, PHO,
Missed:
========================================
EA
Guesses:       DET, IND, ATL, CON
Missed:        WAS
========================================
Total accuracy:     84.62%
Total precision:    87.50%
Total recall:       87.50%
Total f1:           87.50%
```



Development Challenges

During development, **there were many challenges**, mostly related to **how to treat the data during data cleaning and feature engineering** steps in order to achieve satisfactory results while also taking care of the **high dimensionality of the data** that was provided.

With the help of the experimental setup pipeline, the team was able to test multiple data operations combinations until founding the one which results were deemed acceptable and iterated from there.

Another **challenge** that proved **more difficult to solve** was the **small dataset size**, as being limited to 10 years, **many interesting paths taken by the team were fruitless** (and thus won't be covered due to lack of space and not being particularly interesting). For example, many models and hyper-parameters didn't produce a big impact in the results as there were models like neural networks that heavily benefit from larger datasets to work with, **stopping some models from reaching their greatest potential**.

Another example was the attempt of trying different cross validation techniques (expanding and sliding window with different levels of decay for older years) that resulted in **very subtle changes or none at all** for the results.

Work carried out after Initial Presentation

After the initial presentation, the team focused on refining the already good results in order to try maximize metrics with careful actions to avoid overfitting. The following actions were performed:

- Grid search for **hyperparameter tuning** leading to modest model improvements after much tinkering and prolonged computations
- LightGBM model (**ensemble learning**) exploration with mixed results
- **Experimented with adding features** related to team tendencies over the last years to attempt to fix a recurrent identified problem of the models that were very local to each year and didn't take player/team progress into consideration, but results didn't match expectations
- Invested more time into **different cross validation configurations** for sliding and expanding window but with little to no success to show for it

Conclusions

Our journey to predict basketball playoff qualification has **involved problem and data understanding**, **meticulous data preparation**, and **feature engineering**.

We have successfully created a **streamlined pipeline** for model training and achieved promising results with room for improvement with further model configuration.

It's important to note that basketball performance predictions are **conditioned by external factors** from the data we have and that there is also **luck involved**, thus making ~100% precision near impossible to achieve with every-year consistency and making our 87.5% precision even more impressive.

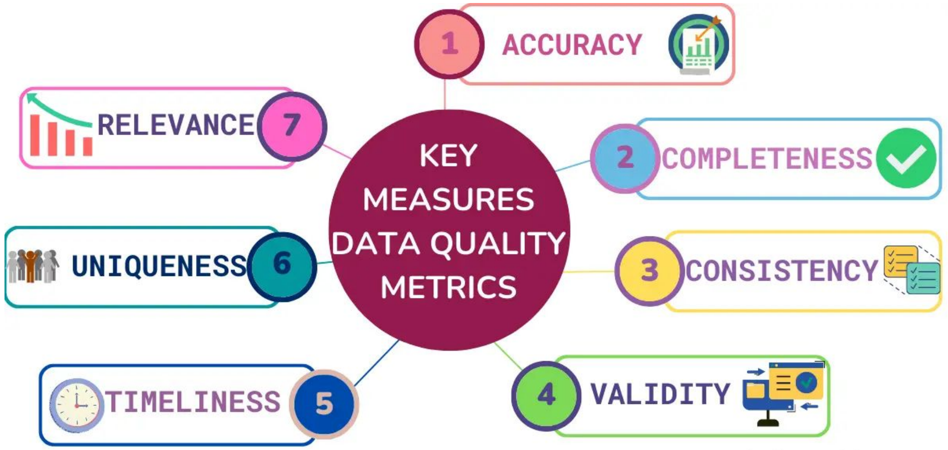
Nonetheless our work exemplifies the potential of data analysis and **predictive modeling** in the realm of basketball playoff prediction, reaching satisfactory results in most later dataset years by just getting one of the 8 playoff teams wrong for many of the models.

Annexes

Data Quality

The data cleaning process guaranteed the quality of data:

- **Accuracy**: Data captures real-world data based on facts (statistics) from previous seasons
- **Completeness**: Missing and invalid values were dropped to ensure completeness.
- **Consistency**: Difficult to access as names of players and coaches are changed.
- **Validity**: Outliers were removed, when appropriate, to ensure validity of data
- **Timeliness**: Data from year 1 to 10 ensures a up-to-date dataset to try to predict year 11
- **Uniqueness and Relevance**: Dataset without duplicated data and useful for predicting the season outcome.



Feature Selection

Necessary to reduce the dimensionality of data, to avoid falling in the **curse of dimensionality**:

- **Hughes Phenomenon**: With only 155 <team, year> pairs it's crucial to have a small set of features to avoid overfitting.
- **Manual attribute filtering phase** with **PCA** to support our decisions. Consequently, reducing the dataset.
- **Correlation matrices** to understand the correlation between features and support decisions about their removal.

Feature Engineering

To also reduce the dimensionality of the data, we aggregated features into other subset of features that would describe them:

- The defensive and offensive features presented in slide 12 were an aggregation of the best 12 (the roster of a team) **defensive/offensive players rankings** which were calculated using the **mean of the normalized** features per player.
- The awards feature is also an aggregation of the awards of players in the team.
- After engineering new features we addressed different data transformation to match models requirements:
    - **Data normalization/Standardization:** models like **KNN, NN** perform better after normalizing/standardizing the data.
    - **Converting Strings to Integers:** The playoff classification was converted to an boolean integer (0/1).

Missing Data, Data Balancing and Sampling

- Features effectively used in the training of the model had no missing data.
- The small amount of data over 10 years makes it difficult to train models like Neural Networks which require larger amount of data and hinder their results.
- Data sampling/balancing is not a problem in this project context:
    - There is **no bias** towards the majority class. The classification ensures that **4 teams per division** are classified as going to the playoffs by taking the greatest probability/confidence given by the model
    - Normally 4 teams in a division go to the next phase and 2/3 don't, so it's reasonably balanced.
    - Although, evaluation metrics like **accuracy** can be **misleading**. If we classify every team as going to the playoffs we get around 66% accuracy.

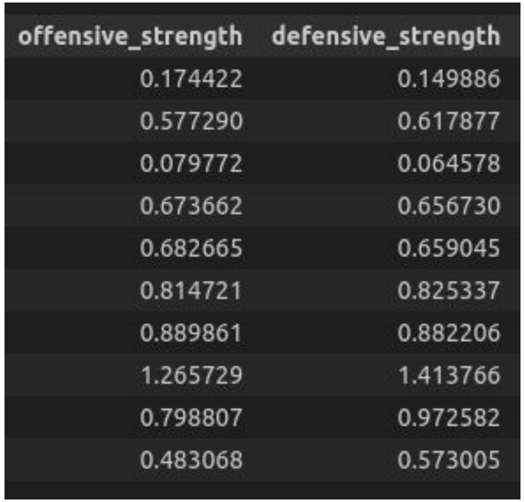
Data sampling is in itself something that doesn't make much sense in the context of the problem, so strategies to increase data quantity like SMOTE wouldn't be effective since we'd lose the factual data we originally had and would fabricate results that being faked, would deviate from the truth we want our models to follow.

Data Integration

As mentioned before, to reduce the data dimensionality we condensed the teams information into a small set of features that describe how the team players are expected to perform.

From a simple mean of the teams player attributes and **without any algorithm** being employed and just relying on crude statistics, we notice that we could predict with **around 75% precision most of the seasons.**

Following this approach means that **new teams** will have valid data to be used in the prediction. An important part of basketball it's the **draft**, although new player differ in quality depending from the **colleagues** that they came from, we opt to assign **a mean value to them**, preventing false assumptions (specially for new players).



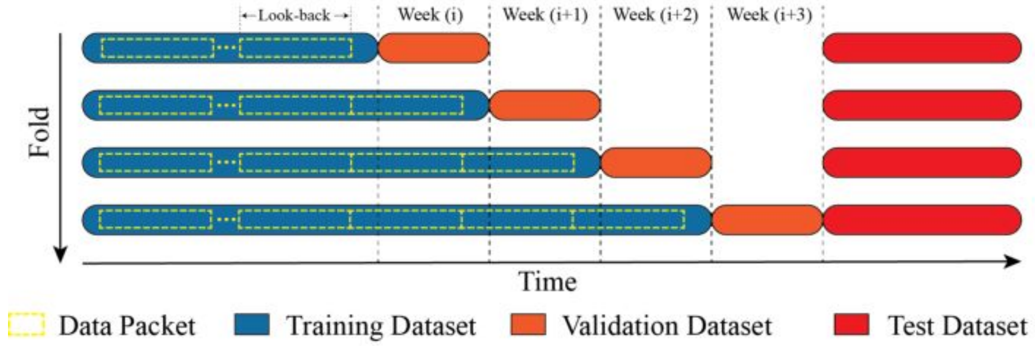
offensive_strength	defensive_strength
0.174422	0.149886
0.577290	0.617877
0.079772	0.064578
0.673662	0.656730
0.682665	0.659045
0.814721	0.825337
0.889861	0.882206
1.265729	1.413766
0.798807	0.972582
0.483068	0.573005

Cross Validation

In **Time series** cross validation is not trivial, as the data is sequential it makes no sense to choose random years to the train and test sets, which would mean training with future data and invalidating model results. We tried different approaches to solve this:

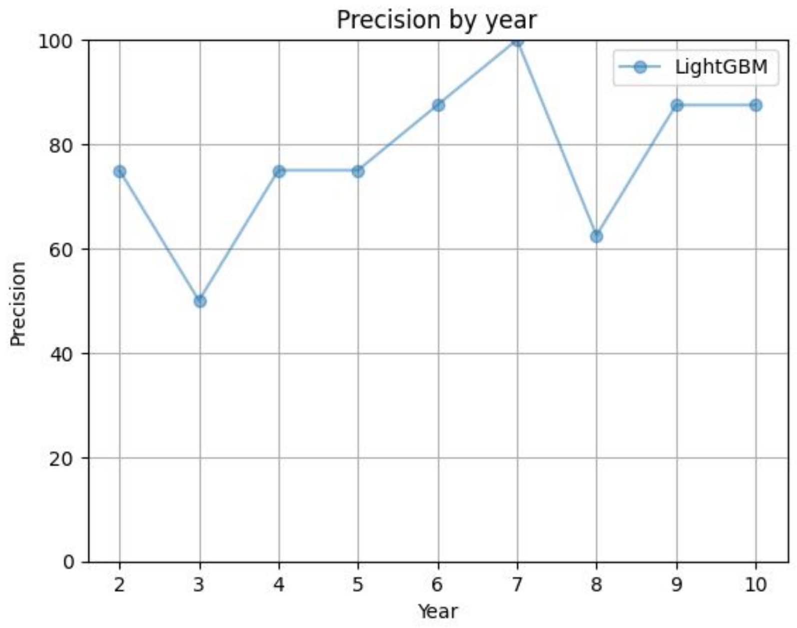
- **Sliding window**
- **Expanding Window**
- **Expanding Window with decay**

Sliding window was the one that got us worst results while the expanding windows got us the best results, but the difference was very mild for all of them.

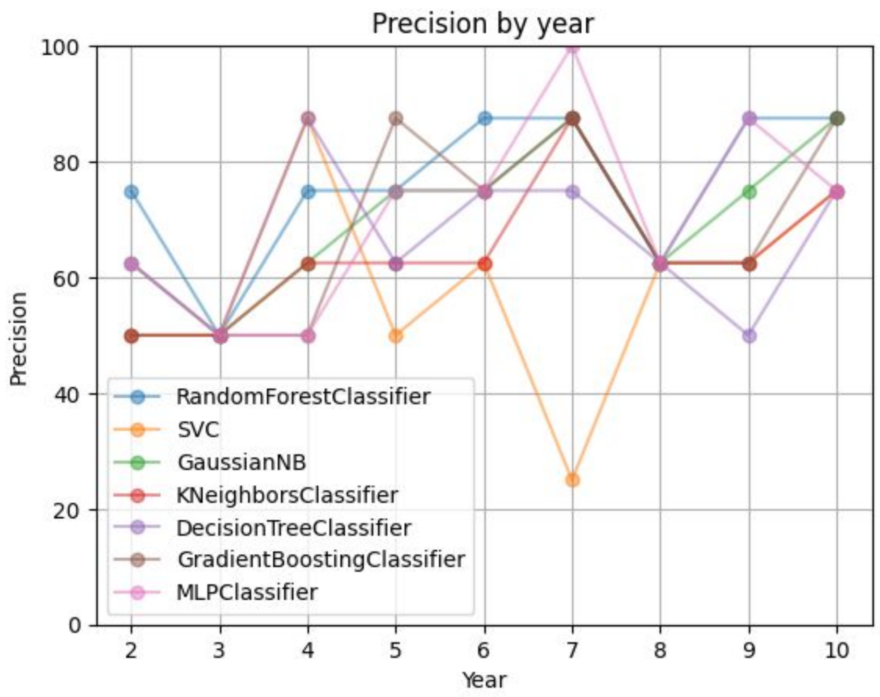


New Ensemble Classifier - LightGBM

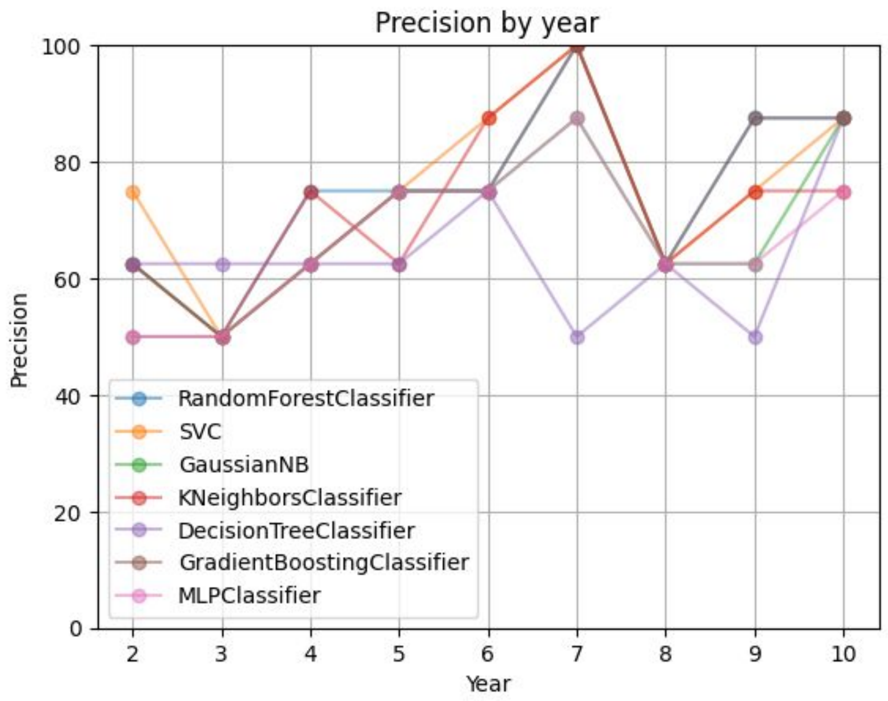
- **LightGBM** is a gradient boosting framework that uses tree based learning algorithms.
- This model is considerably faster and more efficient on training than other models
- The precision results, shown to the right, were considered slightly above average when compared to the other models after grid search



Grid Search - Parameter Tuning



Before grid search hyper parameter tuning

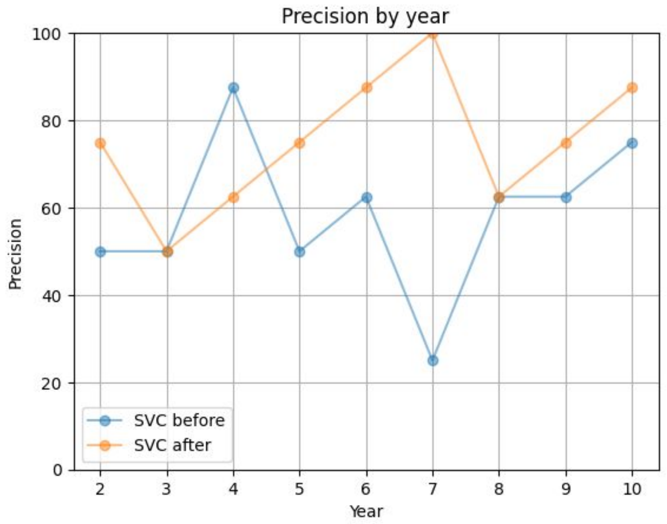


After grid search hyper parameter tuning

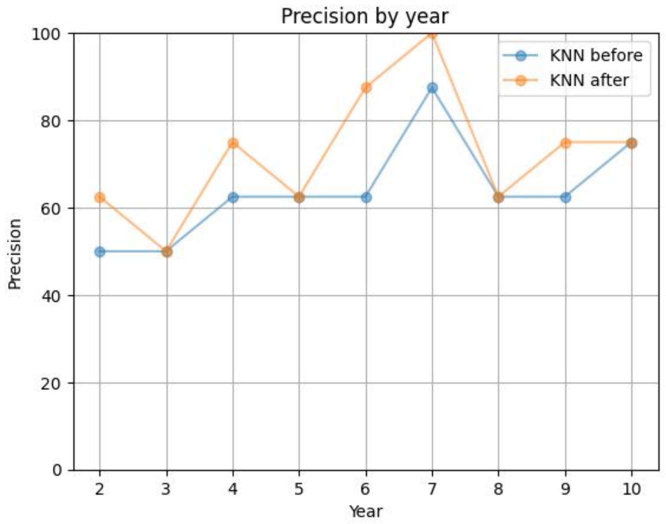
Grid Search - Improvements

The SVC model was initially our worst model. This was surprising, since it's generally a good model for smaller datasets like ours, but after using grid search, the model drastically improved, becoming one of the best.

Most of the models didn't improve much when grid search was used. There were some fluctuations throughout the years, but the average precision remained overall the same. However, KNN was an exception, as it obtained equal or superior precision in all the years.



SVC hyper parameter tuning improvements

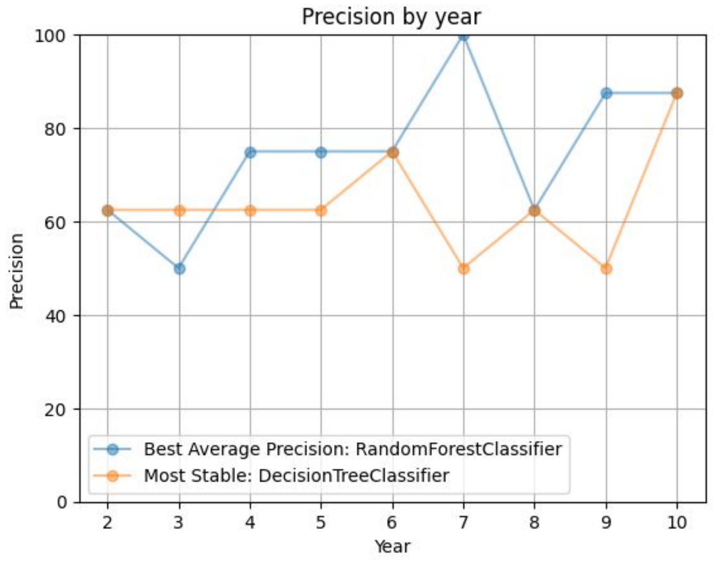


KNN hyper parameter tuning improvements

Grid Search - Results analysis

Just like before, we will consider the two best models, concerning **biggest overall precision** and the most **stable predictions** (based on standard deviation) over time.

This time, after using grid search, the best model remained the **RandomForestClassifier** that didn't improve, and the most stable was the **Decision Tree** instead of **KNN**.

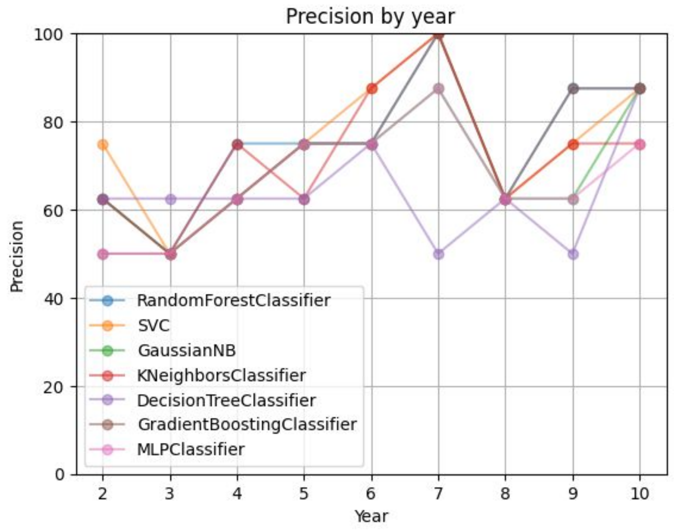


Grid Search - Results analysis

- Besides the 2 previous explored models (SVC and KNN), the rest of the models didn't improve much their overall precision.

- Years 3 and 7 still have a very uniform and slightly lower precision score for almost all the models. By further analysing these years, it's possible to conclude that some of the teams that reached the playoffs were, indeed, not anticipated to achieve such a feat based on their stats

- Some of the models could have benefited from a larger dataset, such as the Neural Network, that was not possible in this scenario.



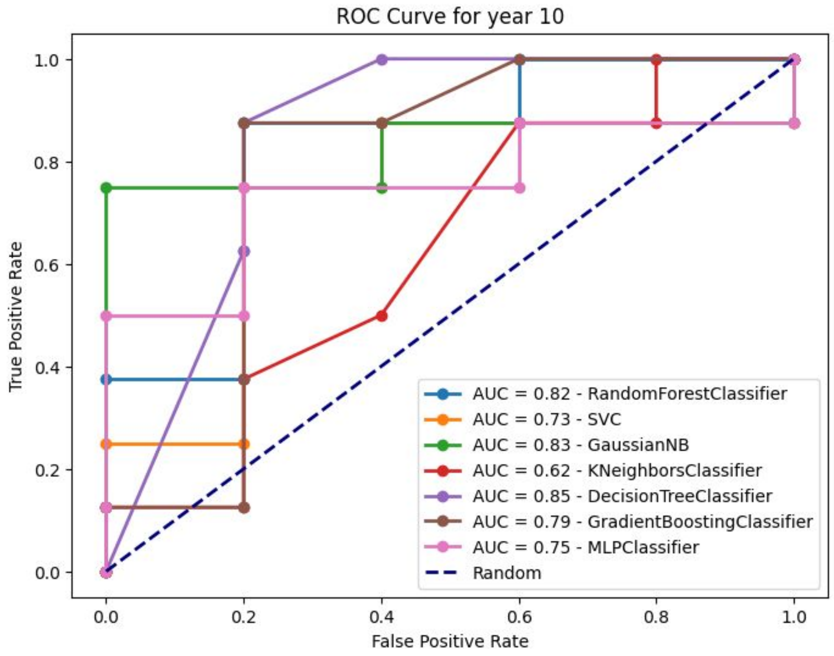
After grid search hyper parameter tuning

ROC and AUC

ROC (Receiver operating characteristic) curves provided us with a more advanced way to explore how well our models could distinguish between classes.

The faster the model reaches 1, the bigger the area under the curve and thus, the best the model performs.

The decision tree classifier was the one with the highest Area Under the Curve (**AUC**) during this season, but both RandomForest and GaussianNB obtained good similar results.

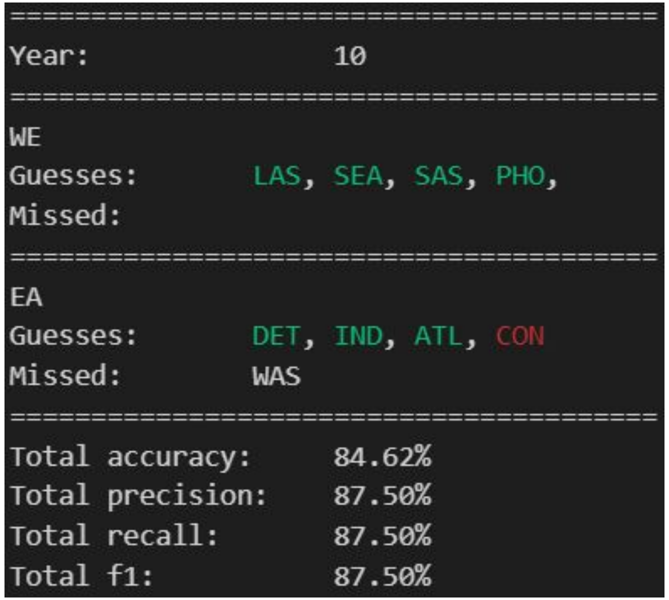


Output analysis

During the development of our project we made continuous experiments and changes to our project in order to improve our models. Many times, it was necessary to manually investigate why the model failed on a certain season prediction.

In order to facilitate the comprehension of our results, we developed a simple, yet effective way to output our results, as you can see in the image.

This was beneficial during the development stage, but it can also assist regular users in comprehending the outcomes of our model by converting raw statistics into a comprehensible visual representation of a model's output.



```
========================================
Year:                   10
========================================
WE
Guesses:       LAS, SEA, SAS, PHO,
Missed:
========================================
EA
Guesses:       DET, IND, ATL, CON
Missed:        WAS
========================================
Total accuracy:         84.62%
Total precision:        87.50%
Total recall:           87.50%
Total f1:               87.50%
```

Development Methodology

During the development of this project, the team adhered to a well-thought-out methodology, with a **strong emphasis on collaboration**.

The Cross-Industry Standard Process for Data Mining framework (**CRISP-DM**) was embraced, guiding the project through well-defined stages – Business Understanding, Data Understanding, Data Preparation, Modeling, and finally, Model Evaluation.

The team worked in close collaboration for tight development and quick feedback loops for **swift project evolution** while ensuring quality and allowing for further direction planning.

The work performed was done mostly asynchronously but with heavy emphasis on live collaboration during presencial practical classes and multiple remote working sessions on **Discord**.

Technologies, Code Organization and Execution

The team used **Python** for the project with heavy use of **Jupyter Notebooks** and extensive use of **external Python Libraries** (more about that on the next slide).

The **Visual Studio Code** code editor was used by all team members and **Git/Github** for version/source control.

The code developed during this project is organized into **2 Jupyter Notebooks** with **many auxiliar script files** that are **fully documented**. There are the following notebooks, organized by performed steps:

- Data Exploration & Data Processing
- Model Predictions and Evaluation

To run the project one might need to run “pip install -r requirements.txt” and then, just run all notebook cells to produce the code outputs. Finally, there is also a markdown readme with more information.

Used Libraries

Below presented are the most important external libraries used during this project:

- Most models used were from **scikit-learn**, with exception of **lightgbm**
- For insightful data visualization **matplotlib** was used
- For seamless data manipulation, **pandas** and **numpy** were used

This combination of tools that were already familiar to all team members ensured effective handling of large datasets and streamlined the workflow, enhancing both model performance and project manageability.

End