PREDICTIVE DATA MINING

BASKETBALL PLAYOFFS QUALIFICATION

ANDRÉ SOUSA (UP202005277) IF:I

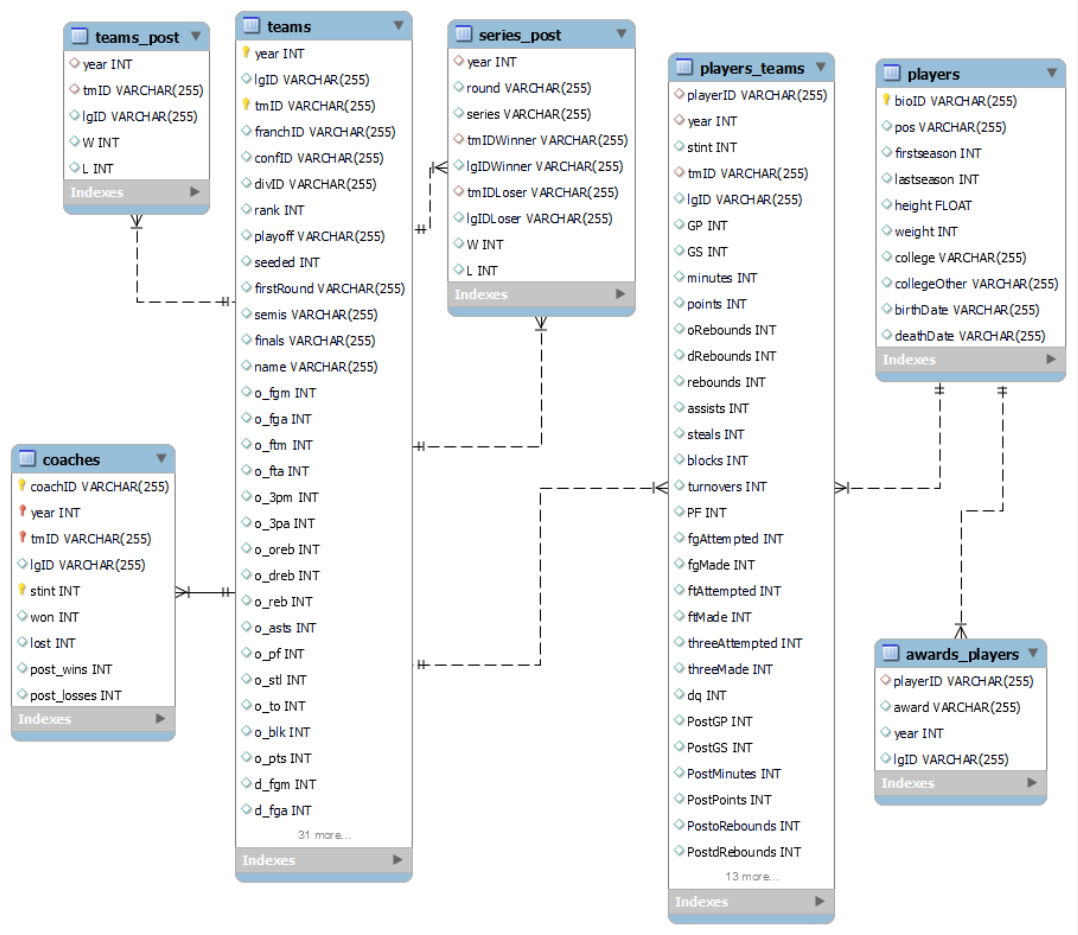
PEDRO FONSECA (UP202008307) IF:I

TOMÁS MACIEL (UP202006845) IF:I

AC – MACHINE LEARNING 2023/2024

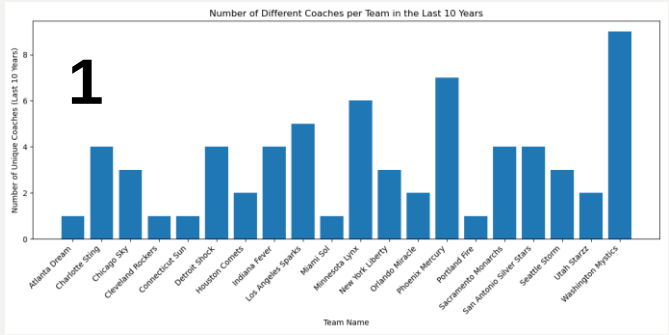
DOMAIN DESCRIPTION

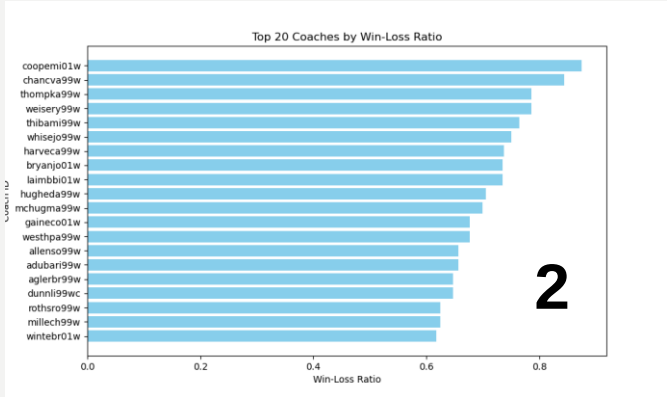
- For 10 seasons (years) data from players, teams, coaches, games, and several other metrics were gathered and arranged on this dataset.
- Our **main goal** is to develop a model using this data that can predict which teams go to the playoffs (8 best teams from the standard season).

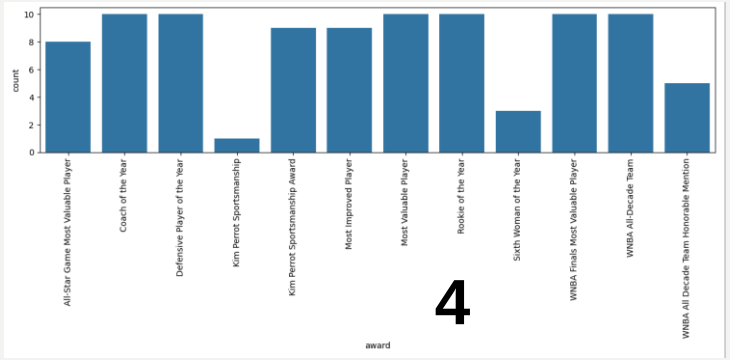


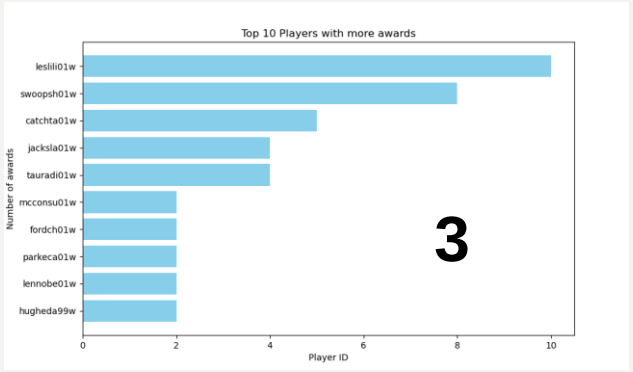
EXPLORATORY DATA ANALYSIS 1/2

1. There are teams with different coaches over the 10 years, however, a few kept the same coach. There are also teams that switched coaches during the season.
2. The 20 coaches with the higher Win-Loss Ratio.
3. The top 10 players with the most awards won, which suggests us the best players.
4. The number of awards given to players.





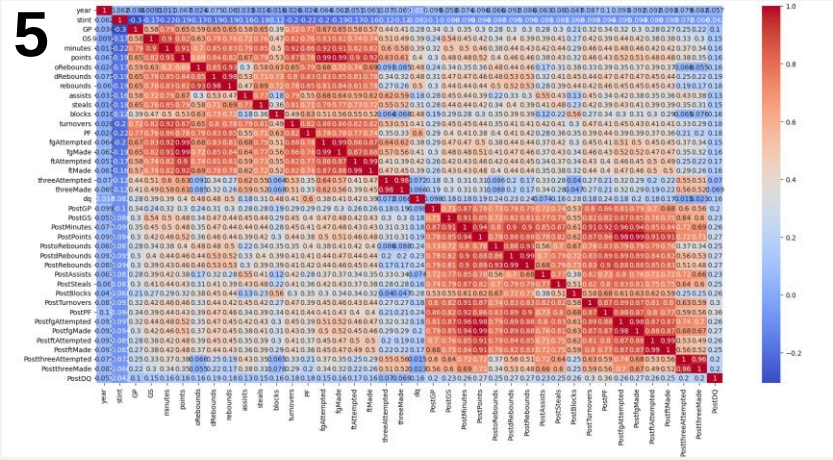




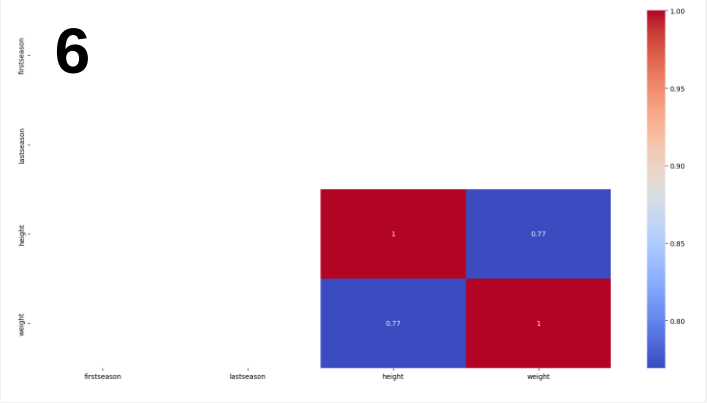
EXPLORATORY DATA ANALYSIS 2/2

5. The correlation matrix of players_teams allows us to see that many individual stats are correlated.
6. Correlation matrix, that shows us the high correlation between weight and height of a player.
7. Playoff appearances per team, show us the teams that are statistically more predictable to advance to playoffs again.
8. Playoff Wins per team that shows us the last winners of the playoffs.

5



6



7

Team	Playoff Appearances
0 LAS	9
1 SAC	8
2 DET	7
3 NYK	7
4 SEA	7
5 CON	6
6 HOU	6
7 IND	6
8 WAS	5
9 SAS	3
10 CLE	3
11 CHA	3
12 PHO	3
13 UTA	2
14 MIN	2
15 ORL	1
16 MIA	1
17 ATL	1

8

Team	PlayOff Wins
0 DET	3
1 LAS	2
2 PHO	2
3 HOU	1
4 SEA	1
5 SAC	1

PROBLEM DEFINITION

This project delves into the domain of basketball tournaments, specifically focusing on the **prediction of playoff qualification for teams**. Basketball tournaments are traditionally divided into two phases: the standard season, where teams compete to accumulate the highest number of wins and the playoffs, featuring knockout matches for the championship. The objective of this project is to **utilize a decade's worth of comprehensive data**, encompassing players, teams, coaches, games, and various performance metrics, to forecast which teams will qualify for the playoffs in the forthcoming season.

DATASET OVERVIEW:

- The dataset at hand **spans ten years**, providing a rich and extensive repository of information relevant to basketball tournaments. It includes detailed player statistics and respective awards, team performance metrics, coaching data, and game-specific information. Furthermore, the dataset also provides data on the team's performance both in the season and in the post-season.
- Moreover, the dataset is not provided as a typical one-table dataset. Instead, we were provided with data that was not previously cleaned and it was scattered through seven distinct tables.

MAIN STEPS FOR PLAYOFF PREDICTION

- **Feature Engineering** in the different tables: creation, removal, and transformation of features.
- **Prediction** of well-known player evaluation metrics, such as, **Efficiency (EFF)** and **Defensive Player Rating (DPR)** that reflect the player's performance in the upcoming season.
- **Prediction of every teams' results**, according to the previously predicted statistics of their players.
- Comparison between teams' results and **choice of the best 8 teams** for qualification.

DATA PREPARATION 1/2

TABLE *player*

- Drop irrelevant features: deathDate, NaN pos, firstSeason, lastSeason;
- Drop players whose birthDate = 0000-00-00;
- Mapping colleges' names and player's positions to numerical indexes;
- Update birthDate to birthYear (day and month are irrelevant for our goal);
- Merge table awards with players, and then add a new attribute awards_count for each player.

TABLE *players_teams*

- Feature Engineering (Creation of 'EFF', 'DPR', 'FG_Percentage,' 'FT_Percentage' and 'PPG' and drop the rows with null values of 'FG_Percentage', 'FT_Percentage', 'PPG');
- Merge *players_teams* with *players* by playerID (playerID matches bioID column present on *player*);
- Creation of lagged features to be used with the prediction models.

DATA PREPARATION 2/2

TABLE *series_post*

- Drop lgIDLoser and lgIDWinner (not relevant) and update 'W' and 'L' to percentages.

TABLE *coaches*

- Creation of new columns on table *coaches* ('WLRatio' and 'WLRatio_Post').

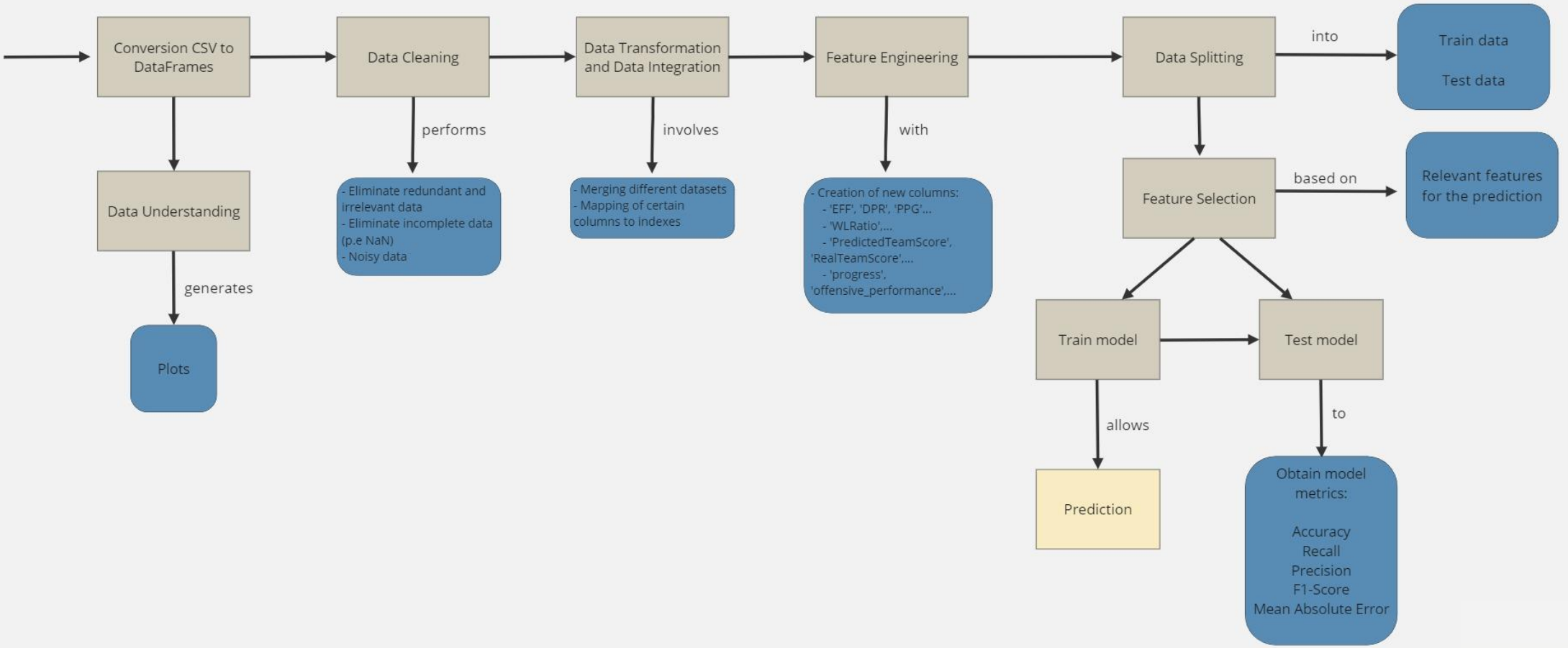
TABLE *teams*

- Feature Engineering related to score (creation of 'PredictedTeamScore' and 'RealTeamScore');
- More Feature Engineering ('progress', 'offensive_performance' and 'defensive_performance', …);
- Merge of table *coaches* with *teams;*
- Mapping of the playoff qualification to Boolean values ('Y'=1 and 'N'=0);
- Creation of lagged features to be used with the prediction models.

LAGGED FEATURES

- Including historical context in our data allows our models to catch trends and improve accuracy in the forecasting of variables.
- Used in:
  - *players_teams* - 'FG_Percentage', 'FT_Percentage', 'PPG', 'EFF', 'DPR';
  - *teams* - 'RealTeamScore', 'defensive_performance', 'offensive_performance', 'gamesWLRatio', 'homeWLRatio', 'awayWLRatio', 'confWLRatio', 'progress', 'playoff';
- For each year we added the information of the X previous years, being X, the size of the sliding window used for testing (see annex: slide 21).
- In case of non-existent values (happens when there are less than X prior years) we define the values as -1 except for playoff where we define them as 0.5 (represents uncertainty).

EXPERIMENTAL SETUP



RESULTS 1/2

- To achieve the final prediction, we made predictions on the players' EFF and DPR, so that we could forecast their performance.
- To predict players' EFF, we utilized the 'Random Forest Regressor' model, which was selected from our previous testing.
- The 'RobustScaler' was applied to enhance the accuracy of the EFF predictions. 1
- For forecasting players' DPR, we maintained consistency by using the same 'Random Forest Regressor' model and 'RobustScaler' approach. 2

Model	Scaler	Best Score	Mean Absolute Error	R Squared
Random Forest Regressor	RobustScaler	0.708	2.601	0.575
Lasso Regression	None	0.527	2.548	0.590
Gradient Boosting Regressor	RobustScaler	0.744	2.692	0.532
Support Vector Regressor	None	0.726	4.015	0.027

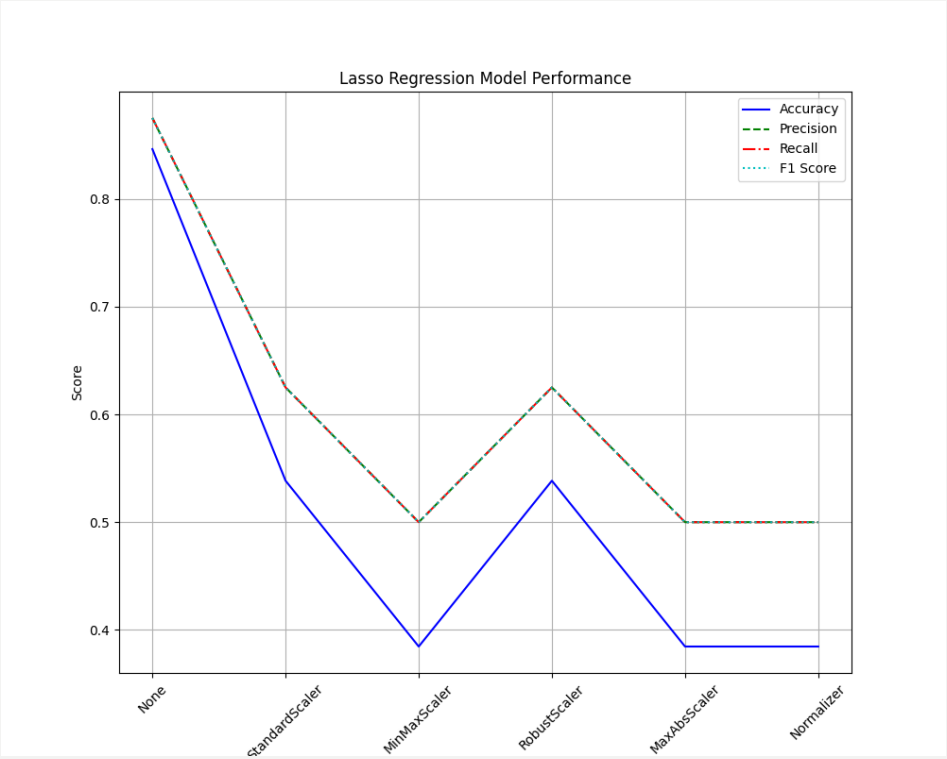
1

Model	Scaler	Best Score	Mean Absolute Error	R Squared
Random Forest Regressor	RobustScaler	0.69358	246.68	0.564
Lasso Regression	None	0.487	237.417	0.5929
Gradient Boosting Regressor	RobustScaler	0.733	262.191	0.489
Support Vector Regressor	None	0.710	367.87	0.0196

2

RESULTS 2/2

- The main goal of our project was to predict which teams go to the playoff, and for that the model that had the highest accuracy, was the 'Lasso Regression Model', with None Scaler.



- As we can see, with None Scaler, it obtained an accuracy of nearly 0.85 and values of Precision, Recall, and F1 Score of 0.875.
- This accuracy is very good, given that we must predict the first 8 teams of 12, and one wrong prediction has a large effect on the final accuracy.

CONCLUSIONS

- At this stage, we believe that the project was a success. We started by making a robust and complete analysis of the data, which allowed us to understand early on what would be useful for our prediction models, which led to efficient data cleaning in the first few weeks of the project.
- As a team, we conducted research to identify and choose specific metrics that formed the foundation for the prediction models we developed. These metrics are commonly utilized by the NBA to assess players' performance.
- We tested a variety of models and chose the most accurate as our 'official' model. By evaluating different machine learning algorithms, we were able to identify the weaknesses and strengths of each.

ANEXES

DATA ANALYSIS 1/2

- Utilized Pandas functions such as describe(), info(), isnull(), etc., during data analysis for comprehensive insights into the provided data.
- Gained a detailed understanding of the data, including its columns, data types, and potential issues.
- Identified columns with a substantial number of null values that didn't contribute relevant information to our problem.
- Discovered that the 'divID' column in the 'teams' dataset was entirely filled with NaN values, leading to its removal for analysis.

DATA ANALYSIS 2/2

- We also observed that in the table '*players*' there were many columns with NaN values.

Column	Sum of Nan values
bioID	0
pos	78
firstseason	0
lastseason	0
height	0
weight	0
college	167
collegeOther	882
birthDate	0
deathDate	0

- Our solution:
  - For the 'pos' column, we dropped the Nan values and kept the others, not Nan.
  - For the columns related to the colleges, the solution we found was to make a mapping, giving each college an index.

FEATURES CHOICE 1/3

- As we presented before, we created 'important' features that were directly used by the models: EFF, DPR, defensive_performance, offensive_performance, and TeamScore.
- **EFF**
  - This is the official NBA individual player efficiency. It is derived by the following formula that we used:

    (PTS + REB+ AST + STL + BLK – Missed FG – Missed FT – TO) / GP
- **DPR (Defensive Player Rating)**
  - This metric shows the Defensive Prowess of a player and its defensive impact on his team. It evaluates the player's efficiency on a defensive level and the formula is:

    100 – (100 * (Def. REB + STL + BLK – PF – TO – PTS)) / GP

FEATURES CHOICE 2/3

- **RealTeamScore**
  - This feature was fully invented by us. Basically, integrates the EFF individual metric of a player on a team level. It is the mean of the EFF of all players:

SUM(EFF) / COUNT(EFF)

- **offensive_perfomance**
  - We got this 'team performance measure' from a basketball forum we found and decided to give it a try to complete the evaluation of the performance of a team:

((O_PTS / (O_FGA + 0.44 * O_FTA)) + ((O_FGM + O_3PM) / (O_FGA + O_3PA)) + (O_ASTS / (O_TO + 1)) + O_REB) / GP

FEATURES CHOICE 3/3

- **defensive_performance**
  - Like the 'offensive_performance', we found this formula in the same source and also decided to use it to measure the team's defensive performance:

    (((D_FGM + D_3PM) / (D_FGA + 0.44 * D_FTA)) + ((D_FGM + D_3PM) / (D_FGA + D_3PA)) + D_REB+ D_STL + D_BLK – D_PTS) / GP

- The use of these new features allowed a complete analysis of several statistics (both individually and team level), which gave us the best chance at creating a robust and efficient model.

EVALUATION METHODOLOGY: PROBLEMS

- In this project, we did not use the Hold-out methodology for evaluating our models since this technique does not provide sufficient data for training and testing the models. For this reason, we applied the Stratified K Folding methodology. However, the standard implementation of this technique is not suitable for the problem at hand since it splits the data regardless of its semantical meaning.
- In this case, a likely scenario would be, for example, training the model with portions of data from seasons 5 and 8, but testing would be done with the complete data from season 7. This is not a realistic scenario; therefore, it cannot be used to train and evaluate our models.

EVALUATION METHODOLOGY: SOLUTION

- To implement a version of the Stratified K Folding technique that pays attention to the semantic meaning of the data, we followed the advice given in the theoretical classes.
- Instead of grouping random data from the main dataset into k folds, we implemented an alternative that mimics the functioning of a sliding window.
- Each fold only contains data from complete seasons.
- The several folds are obtained by adjusting the time delta of the algorithm. For example, if the dataset contains data from 6 years and we are dividing the data into 3 folds, each fold will contain the data corresponding to 2 years.
- Using this approach, we were able to train our models with different window sizes, keeping their semantic meaning and simulating a realistic situation, therefore resulting in more reliable predictions.

LAGGED FEATURES IDEA

- To complement the Sliding Window methodology, we applied Feature Engineering to our data.
- As we mentioned before, we created Lagged Features, so that, for each feature, we added the information of the previous X years, being X the length of the sliding window. We did this because it would improve both the accuracy and efficiency of the model.
- When we tested, we concluded exactly that so, for the other metrics that we created after to be used in the models, we kept doing that and creating the lagged features using the shift function, which made our work easier.

ML MODELS EXPLORED

- In the upcoming slide, we will enumerate the Machine Learning models utilized for predicting the qualification of basketball teams for the playoffs.
- In this project, we explored both regression and classifier models, depending on our target variable.
- The regression models were used to predict each player's future performance, by computing the value of the player's EFF and DPR performance metrics in the upcoming season. Furthermore, we also used this type of model to predict the likelihood of the team's qualification to the playoff.
- We opted for regression models over classifiers due to our specific objective of computing a score ranging from 0 to 1, a metric that would determine the top 8 teams eligible for playoff qualification. This approach allowed us to scrutinize and compare the likelihoods of qualification, enabling us to select the most suitable ones, rather than blindly designating 8 teams with a "Y" label, devoid of additional criteria.
- However, for pedagogical purposes, we also employed classifier models to forecast playoff qualification, offering a valuable basis for contrasting the outcomes against those produced by the regression models.

ML MODELS EXPLORED

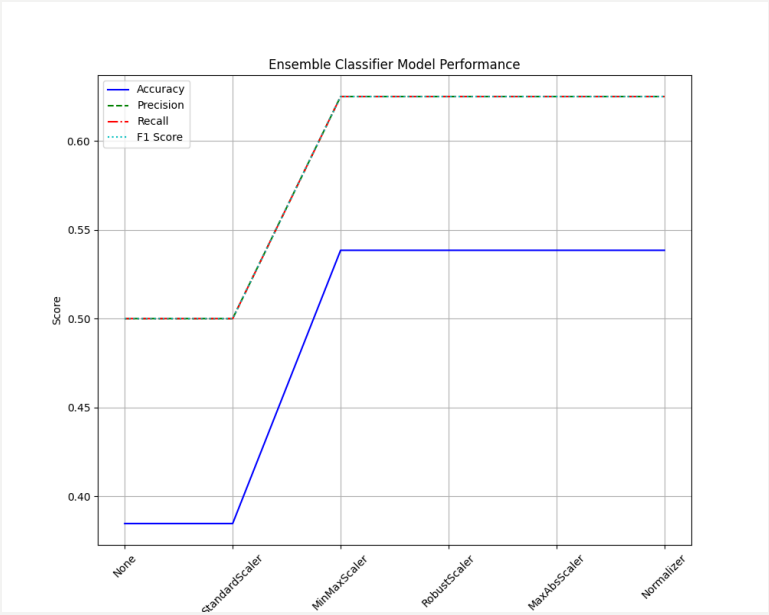
- Regression models:
  - Linear Regression
  - Random Forest Regressor
  - Gradiente Boosting Regressor
  - Support Vector Regressor
  - Ridge Regression
  - Lasso Regression
  - MLP Regressor

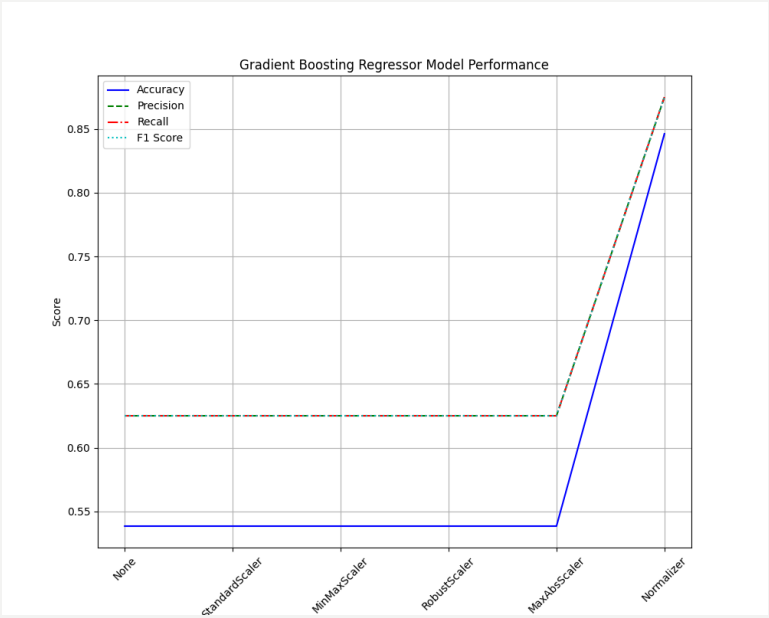
- Classifier models:
  - Linear SVC
  - Naive Bayes
  - K-Nearest Neighbours Classifier
  - Support Vector Classifier
  - Voting Classifier (Ensemble)
    - Logistic Regression
    - Random Forest Classifier
    - Decision Tree Classifier

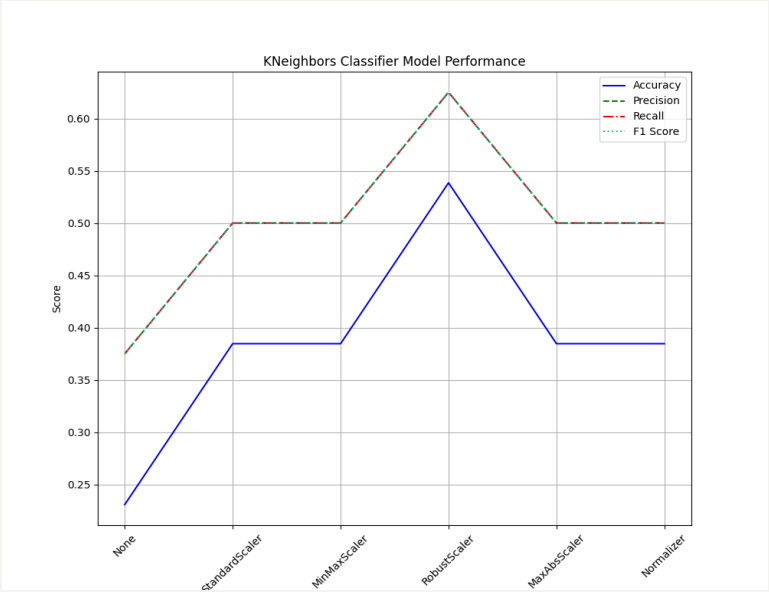
SCALERS EXPLORED

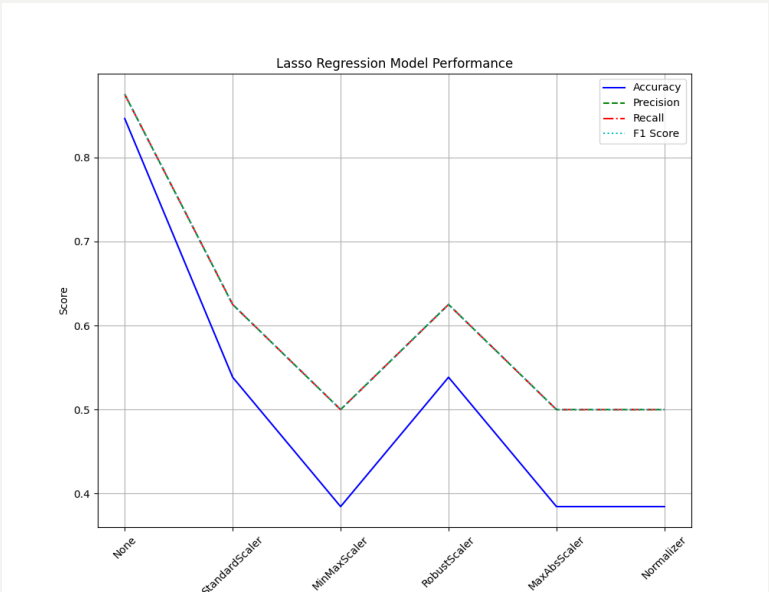
- Given that the output of a Machine Learning algorithm is intricately tied to the input data it receives, we embarked on an exploration of diverse data normalization techniques. Our aim was to assess the impact of these normalization methods on the performance of each model.
- To this end, we conducted a comparative analysis of the following scaling algorithms:
  - Standard Scaler
  - MinMax Scaler
  - Robust Scaler
  - MaxAbs Scaler
  - Normalizer

SCALERS EXPLORATION RESULTS 1/3

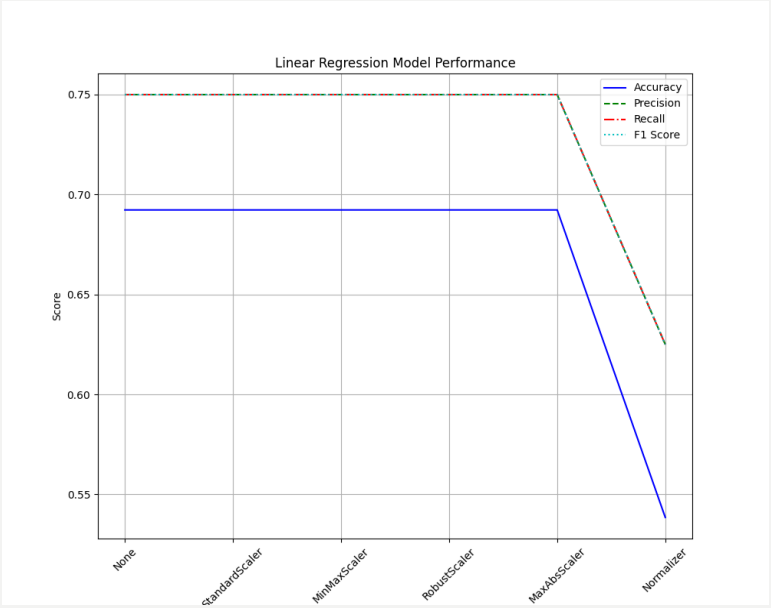


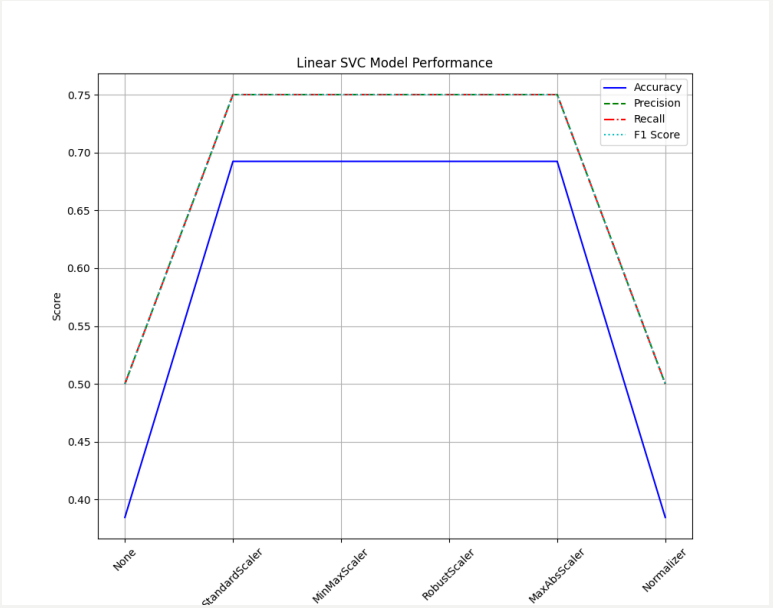


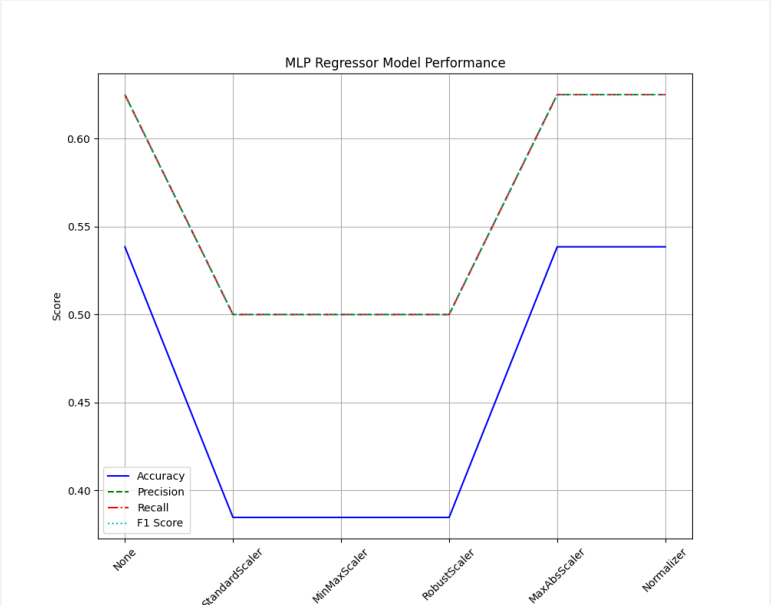


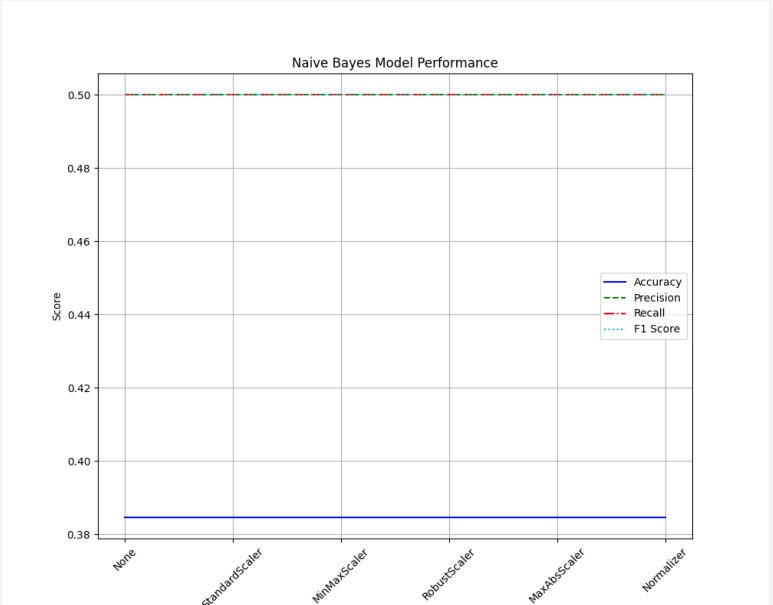


SCALERS EXPLORATION RESULTS 2/3

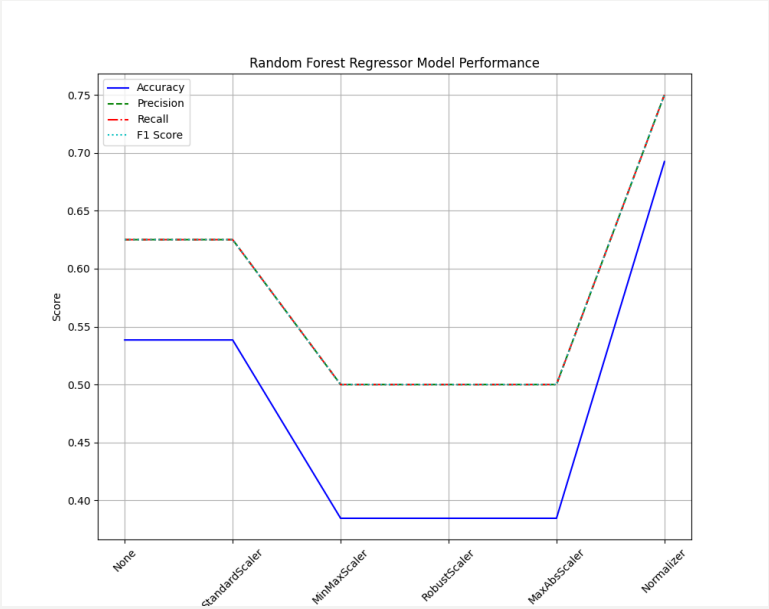


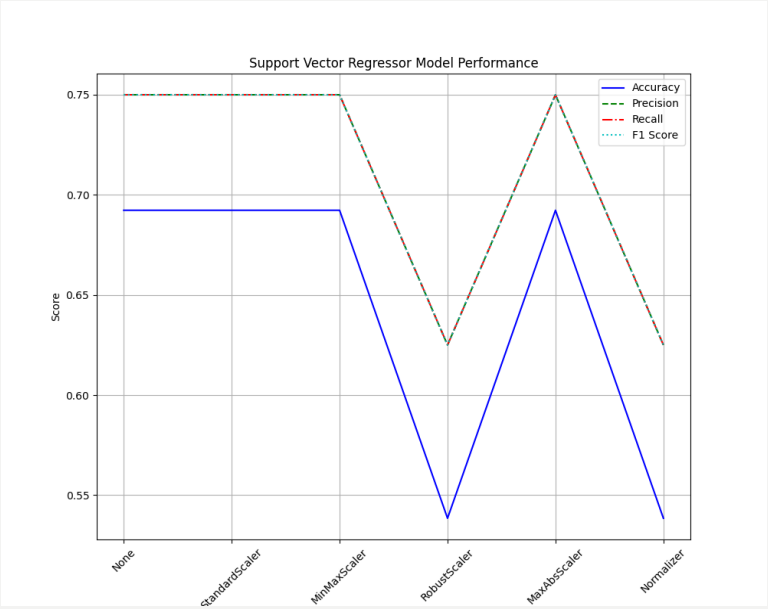


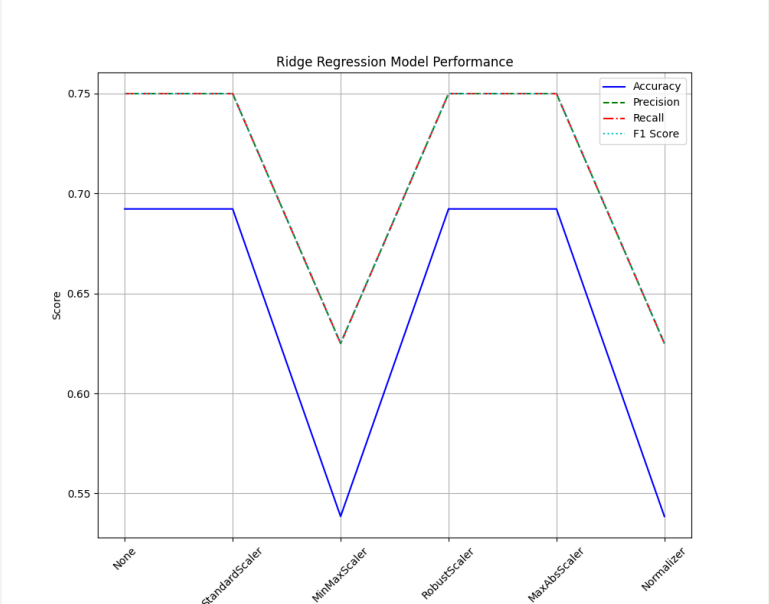


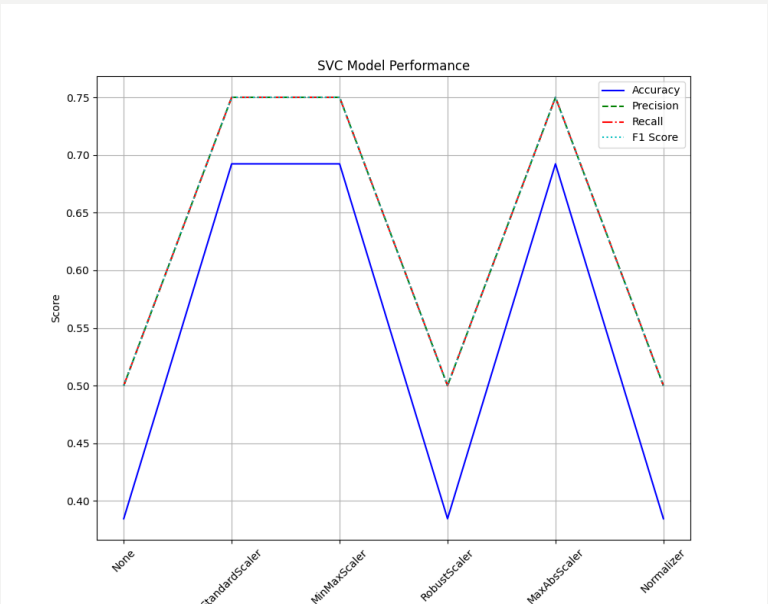


SCALERS EXPLORATION RESULTS 3/3









SCALERS EXPLORED: ANALYSIS

- **Linear Regression, Support Vector Regressor, and Ridge Regression** consistently achieve high scores across all Scalers, with accuracy, recall, precision, and F1 score of 0.625 or higher.

- **Random Forest Regressor and MLP Regressor** exhibit less consistent performance, with scores varying depending on the Scaler used. However, the overall scores are lower than the results mentioned in the previous topic.

- **Lasso Regression** achieves the highest accuracy and F1 score among all regression models, indicating strong overall predictive capability. Furthermore, the best result was obtained without using a Scaler.

- **Gradient Boosting Regressor** demonstrates competitive performance with high accuracy, recall, precision, and an F1 score under the Normalizer Scaler.

- As expected, **Naïve Bayes** did not perform well in this problem. We were counting on this outcome, because this algorithm assumes that all the features are completely independent, however, as displayed in the confusion matrices, the columns have some correlation among them, therefore invalidating the assumption made by Naïve Bayes and leading to poor performance.

SCALERS EXPLORED: ANALYSIS

- The choice of Scaler significantly impacts the performance of most regression models, as scores vary across different Scalers.

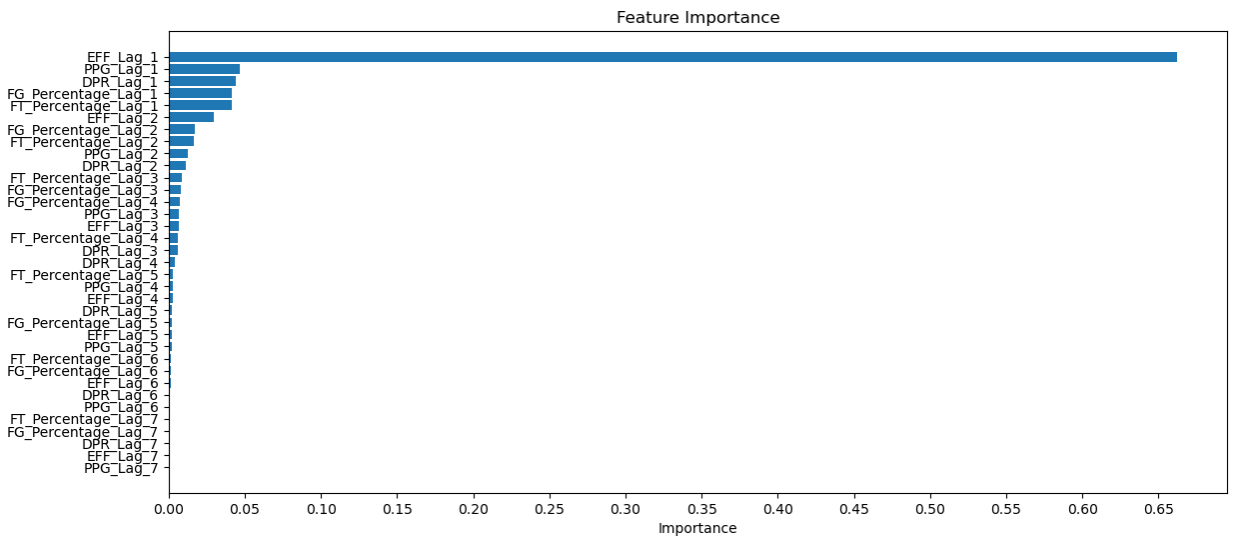
- The "None" Scaler is used for some models, indicating that no data scaling was applied. These models achieve mixed results, highlighting the potential benefit of data scaling in certain cases.

- It is important to highlight that the testing process used to collect this data closely mirrors the conditions of the final Kaggle evaluation. However, this similarity gives rise to a specific challenge: in each season, an average of 13 teams compete, but only 8 secure a spot in the playoffs. Consequently, any misjudgment carries a substantial impact on the evaluation metrics.

- For example, if the model anticipates the qualification of a team that does not make it, this single error accounts for 2/13 or 15,4% of the final accuracy value. This is because an erroneous prediction of a team's qualification implies the incorrect prediction that another team did not qualify when it should have, effectively doubling the impact of the mistake.

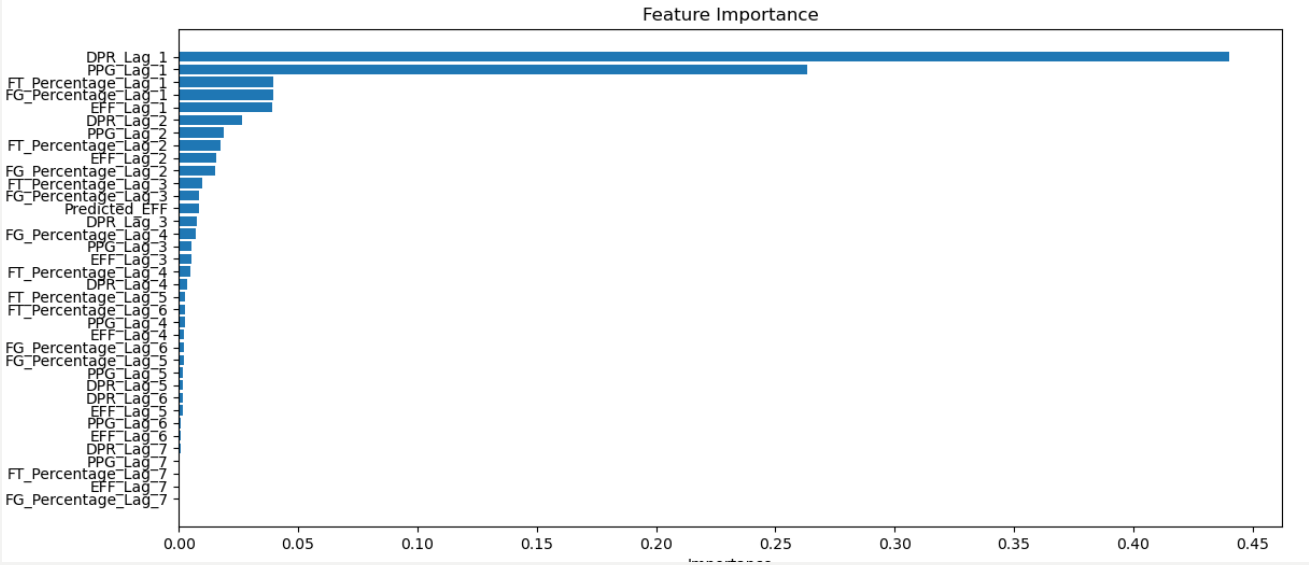
In summary, the data obtained offers valuable insights into the performance of various Machine Learning models across different data scaling techniques. It highlights the importance of selecting appropriate Scalers for different models and tasks. Additionally, it indicates that some models are more robust across different Scalers, while others exhibit sensitivity to the choice of scaling method.

FEATURE IMPORTANCE



- The plot on the left shows the feature's importance by weight plot for predicting the EFF for the players

- The plot on the right shows the feature's importance by weight plot for predicting the DPR for the players



GAME SIMULATION ALTERNATIVE

- Using the *series_post* dataframe we were able to create an alternative method of determining the teams going to the playoffs.
- By using the individual games in the series_post we were able to create a model capable of predicting the odds of a particular team winning.
  - Data passed to the model:
    - Lagged: 'RealTeamScore', 'defensive_performance', 'offensive_performance', 'gamesWLRatio', 'homeWLRatio', 'awayWLRatio', 'confWLRatio', 'progress', 'playoff'
    - Current: 'PredictedTeamScore'
- With this model, we then ran simulations of the games for the standard season to determine the best teams.
- Right from the start, we had doubts about this model due to the lack of individual game data (only 70 games to train and test on)
- The obtained results proved that this approach is not reliable:
  - Accuracy: 69.2 %
  - Precision: 75%
  - Recall: 75%
  - F1 Score: 75%

TECHNOLOGIES/TOOLS

- To make this project, we used different technologies like libraries to draw plots or libraries specific for Machine Learning:
  - **Pandas** – Data manipulation by using Dataframes and analysis
  - **SciKit Learn** – Machine Learning
  - **Numpy** – Numerical computing in some special cases
  - **MatPlotLib** – Plot drawing and visualization
  - **Seaborn** – Data visualization integrated with MatPlotLib
  - **Math** – Mathematical Functions

REFERENCES

- https://en.wikipedia.org/wiki/Efficiency_(basketball)
- https://machinelearningmastery.com/machine-learning-in-python-step-by-step/
- https://realpython.com/python-data-cleaning-numpy-pandas/
- https://www.nba.com/stats/help/glossary
- https://www.datacamp.com/tutorial/matplotlib-tutorial-python