U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Basketball Playoffs Qualification

Data Mining Project
AC - Group 11

Presented By:

Henrique Seabra Ferreira - up202007044

Henrique Santos Ferreira - up202007459

Jorge Costa - up201706518

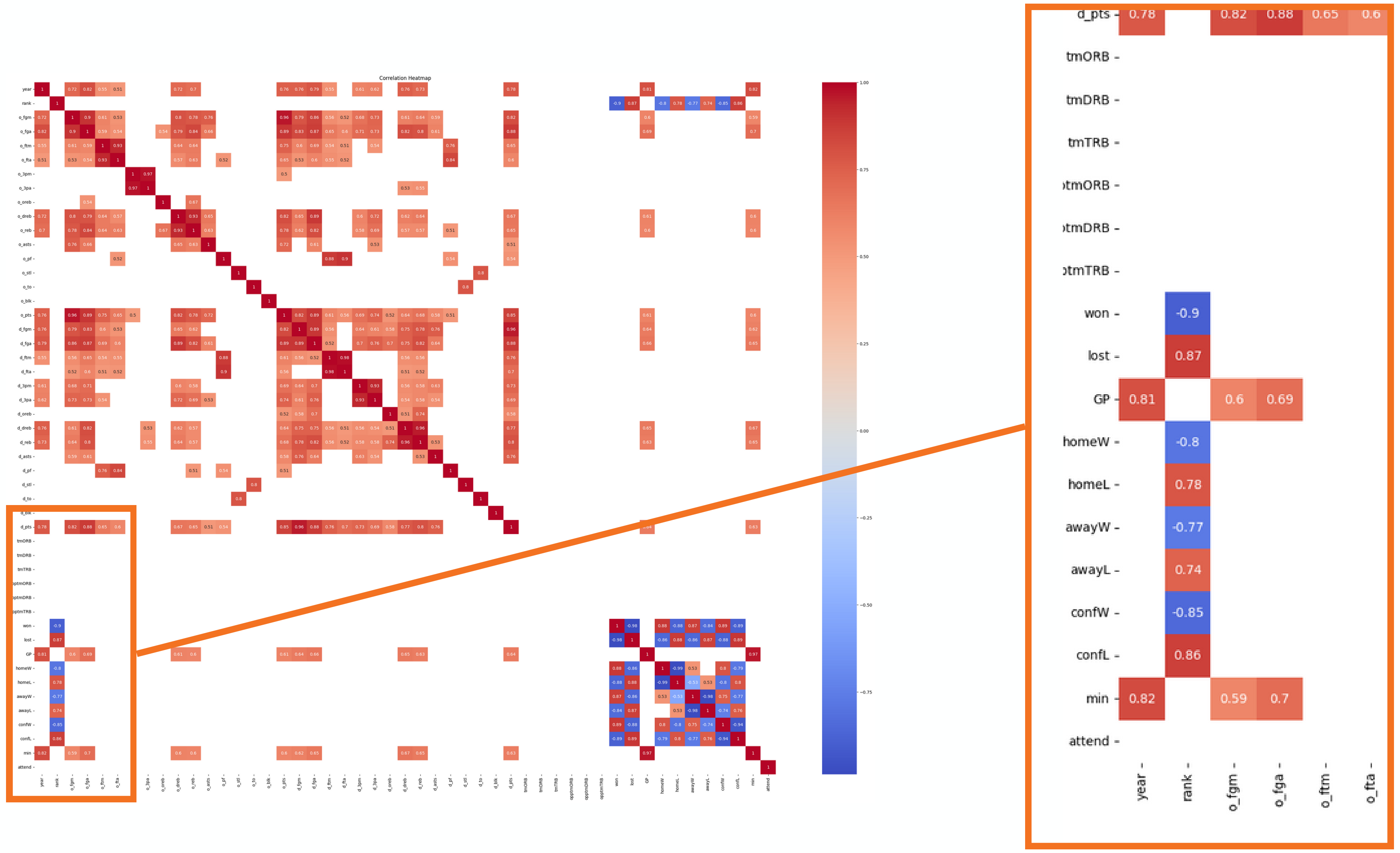
Domain Description

The Women's National Basketball Association (WNBA) is a professional basketball league in the United States featuring some of the most talented female basketball players in the world. Each season, teams compete for the opportunity to qualify for the WNBA playoffs, where a select number of teams advance to compete for the championship title. The data provided about the players, teams and coaches consist of following relations:

- Awards Players: Tracks player awards and honors over ten seasons.
- Coaches: Details coaching staff for different teams during the dataset's timeframe.
- Players: Comprehensive player information.
- Players Teams: Records player performance across various teams.
- Series Post: Documents playoff series results and outcomes.
- Teams: Contains data on team performance across seasons.
- Teams Post: Records playoff performance, offering insights into playoff success.

Exploratory Analysis

One crucial step in understanding the relationships within the dataset is the generation of a correlation heatmap. This technique provides valuable insights into how different variables are related to one another.



There are some obvious correlations that can be observed:

- Strong positive correlations between the number of field goals made (o_fgm) and points scored (o_pts), as well as between the number of field goals attempted (o_fga) and field goals made (o_fgm).
- Negative correlation between "rank" and several offensive statistics, implying that a good team tend to have higher values for these offensive metrics.

Some more interesting informations can be taken:

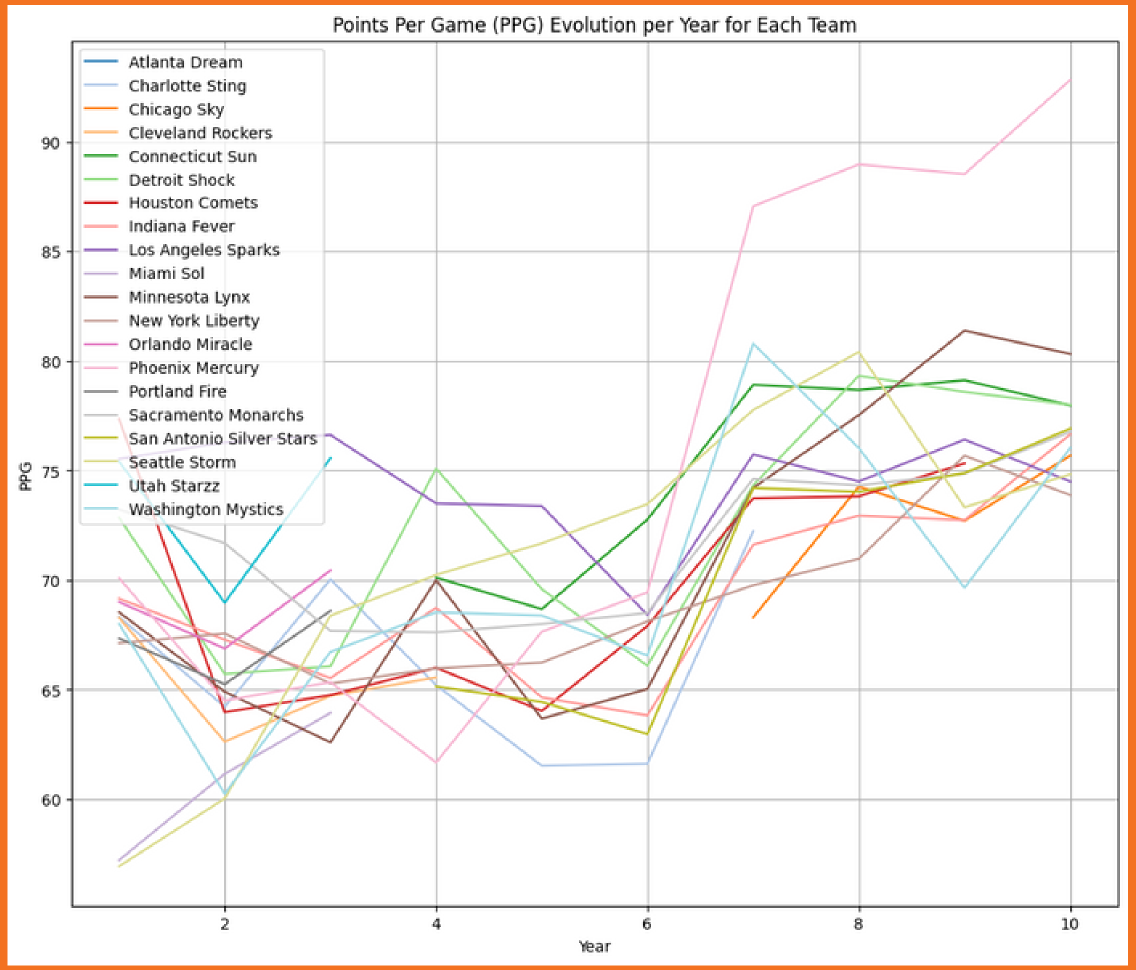
- The "year" variable has a strong positive correlation with various offensive statistics. This suggests that these offensive metrics tend to increase over the years.
- Offensive and defensive rebounding statistics are positively correlated, indicating that teams with more offensive rebounds also tend to have more defensive rebounds and vice versa.

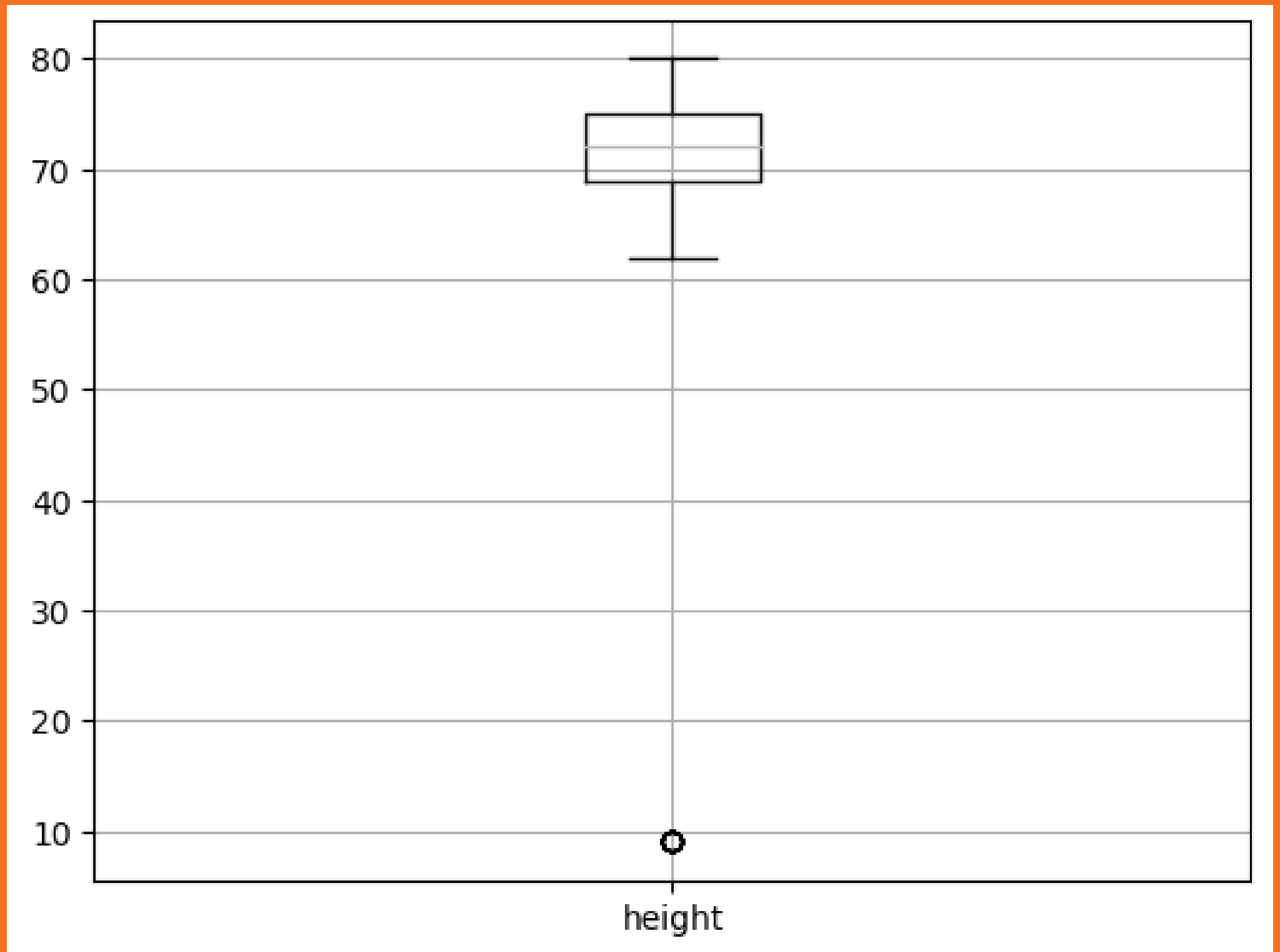
Exploratory Analysis

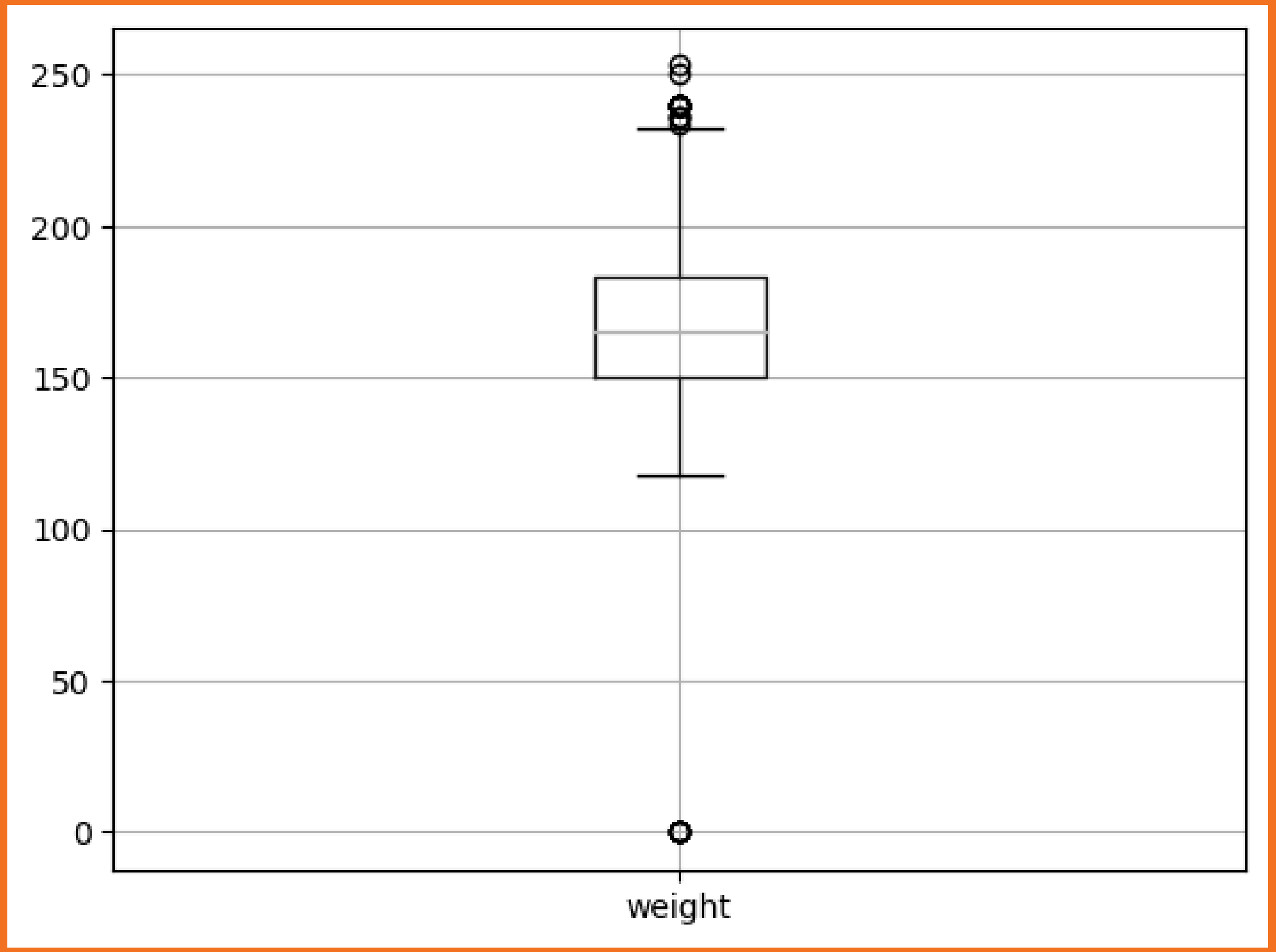
Another step in the process of the exploration of the data was the building of a graphic to analyse the evolution of points made per game from each team.

One of the conclusions taken is that there is a clear rise in PPG after year 6, in all teams.

In terms of outliers, using this boxplots, some outliers are visible in the players data, with height and weight way below possible values. This rows were removed. The one above were mantained, since the values, while out of the ordinary, weren't impossible.







Problem Definition

The primary objective of this project is to develop a predictive model that can accurately identify which teams will qualify for the playoffs in a given season. To achieve this objective, the following key components need to be addressed:

- Data Preparation: Identify and preprocess relevant features and variables that may impact playoff eligibility.
- Model Development: Develop and fine-tune machine learning models that can effectively predict playoff teams.
- Evaluation Metrics: Define appropriate evaluation metrics for assessing the performance of the prediction model.

Data Preparation

01 Deletion of irrelevant and abnormal data about players (players without weight, height or position) and removal of redundant/useless team features

02 Creation of new variables through values obtained from players table (avg. age, weight and height)

03 Creation of variable storing number of awards won by the team players and coaches from awards table

04 Replacing of missing data for first year for players and teams by that year’s medium

Data Preparation

05 Creation of variable based of classification to the playoff in last two years (0 if didn’t, 1 if class. two seasons ago, 2 if classified last season and 3 if in both.

06 Creation of variables based on team players and team performance in the last two years.

07 Creation of variable that assigns weights to the team's playoff qualifications based on the year

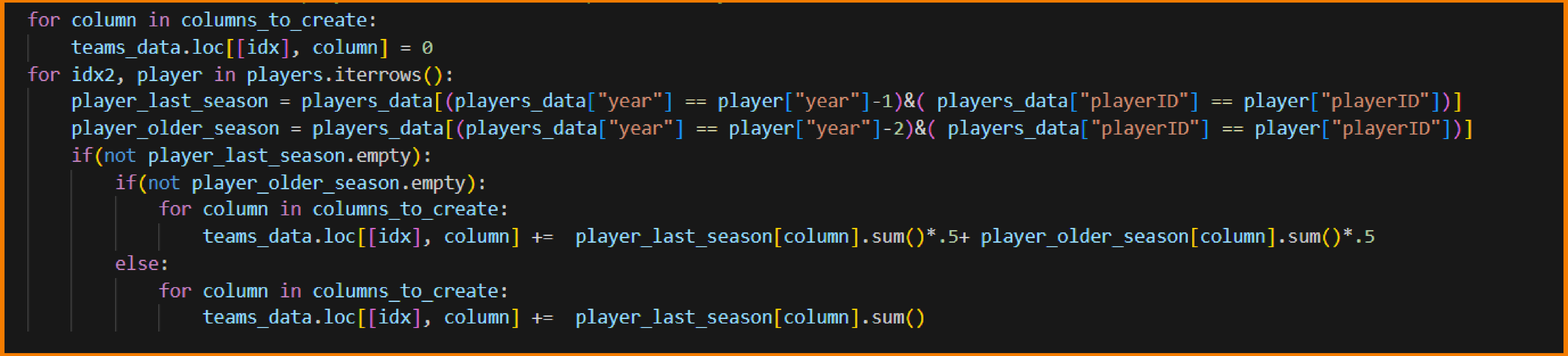
08 Creation of a variable based on the number of coach changes in the last two years

09 Spliting data between training data and test data.

Creation of variables based on team players and team performance in the last two years

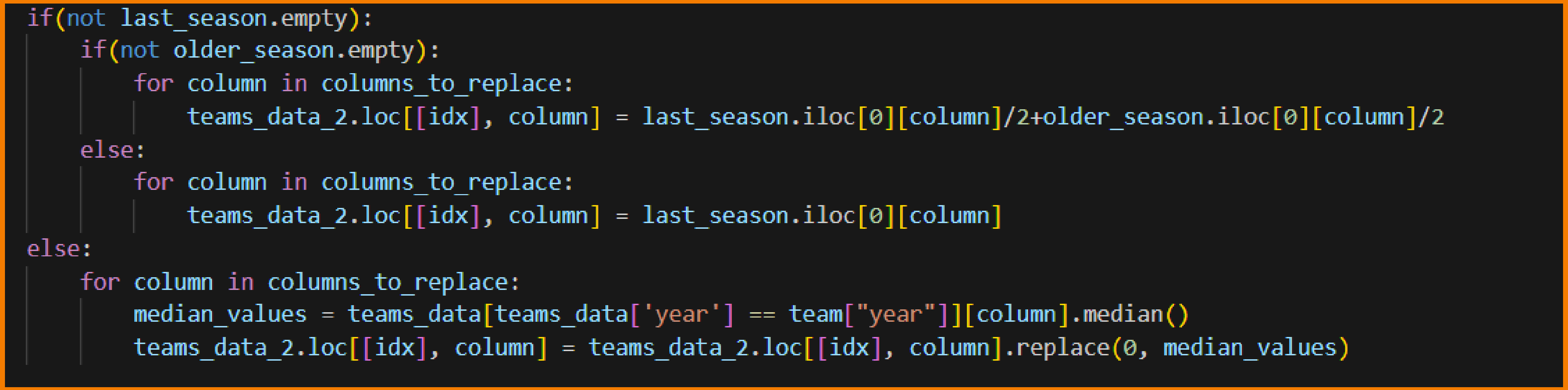
In order to use the players statistics to improve the results, their incorporation in the main table was made the following way:

- Creation of a column for each player table column, valued with the medium of the values of the team's player two last years or just last if two not available.



```
for column in columns_to_create:
    teams_data.loc[[idx], column] = 0
for idx2, player in players.iterrows():
    player_last_season = players_data[(players_data["year"] == player["year"]-1)&( players_data["playerID"] == player["playerID"])]
    player_older_season = players_data[(players_data["year"] == player["year"]-2)&( players_data["playerID"] == player["playerID"])]
    if(not player_last_season.empty):
        if(not player_older_season.empty):
            for column in columns_to_create:
                teams_data.loc[[idx], column] +=  player_last_season[column].sum()*.5+ player_older_season[column].sum()*.5
        else:
            for column in columns_to_create:
                teams_data.loc[[idx], column] +=  player_last_season[column].sum()
```

In terms of team data, the decision has the same, using a medium of the last two years' statistics and replacing the non existing values, for example, for new teams in the league, for the medium of that variable in that year.



```
if(not last_season.empty):
    if(not older_season.empty):
        for column in columns_to_replace:
            teams_data_2.loc[[idx], column] = last_season.iloc[0][column]/2+older_season.iloc[0][column]/2
    else:
        for column in columns_to_replace:
            teams_data_2.loc[[idx], column] = last_season.iloc[0][column]
else:
    for column in columns_to_replace:
        median_values = teams_data[teams_data['year'] == team["year"]][column].median()
        teams_data_2.loc[[idx], column] = teams_data_2.loc[[idx], column].replace(0, median_values)
```

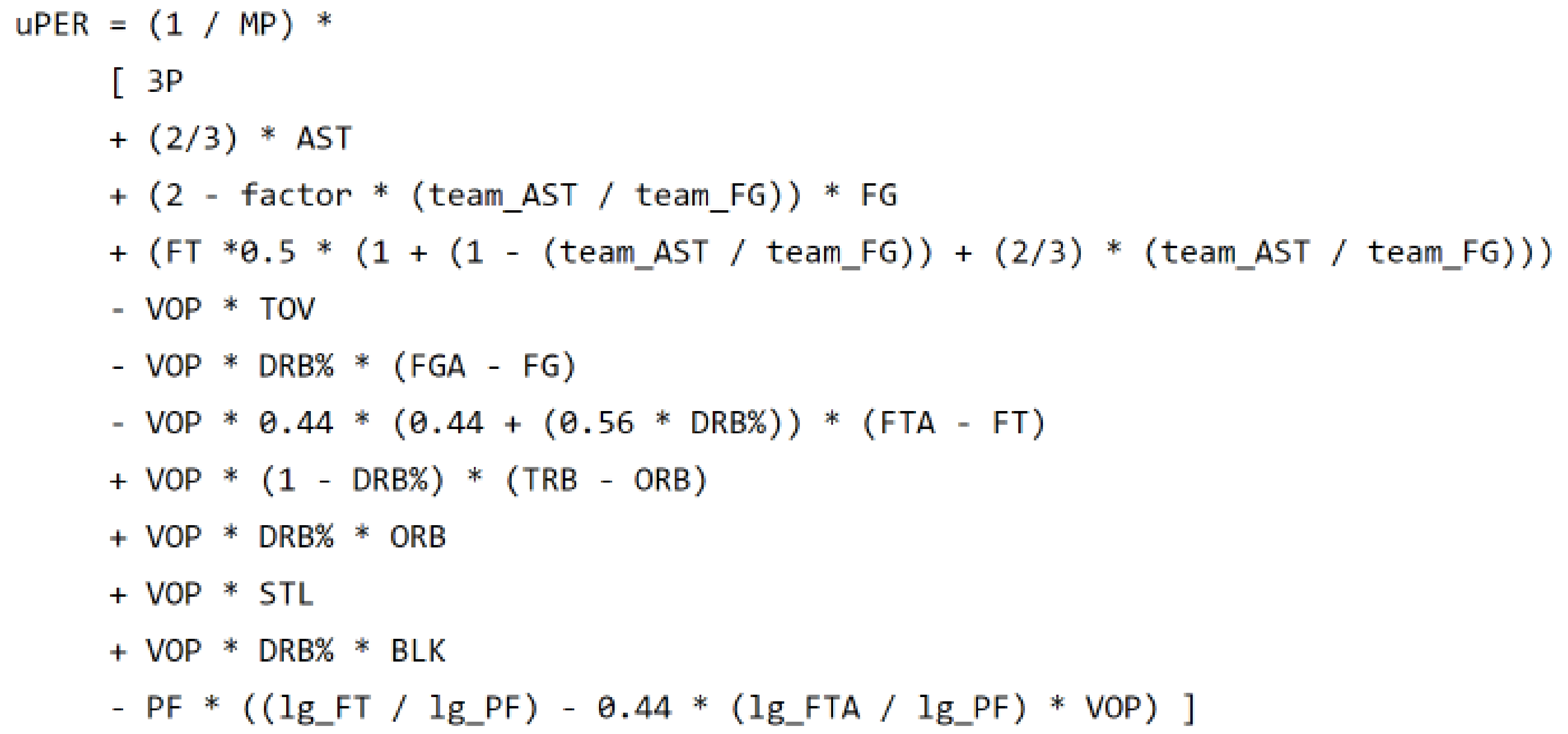
Creation of variables based on team players and team performance in the last two years - PER

One tested alternative to the use of all the players’ variables, avoiding the inclusion of a great number of new columns, was the summarization of them in only one new column - PER.

Player Efficiency Rating (PER) is a basketball statistic that aims to quantify a player's overall performance on the court. PER is calculated using a complicated formula, taking in account various player statistics, including points, assists, rebounds, steals, blocks, and turnovers.

- Points, assists, and rebounds contribute positively to PER.
- Turnovers and missed field goals contribute negatively.
- Higher PER values indicate more efficient and impactful players.

While PER provides a single-number summary of a player's overall performance and adjusts for pace and playing time, trying to allow fair comparisons between players, it does not capture intangible qualities like leadership and teamwork and doent’s contain all player's defensive contributions.

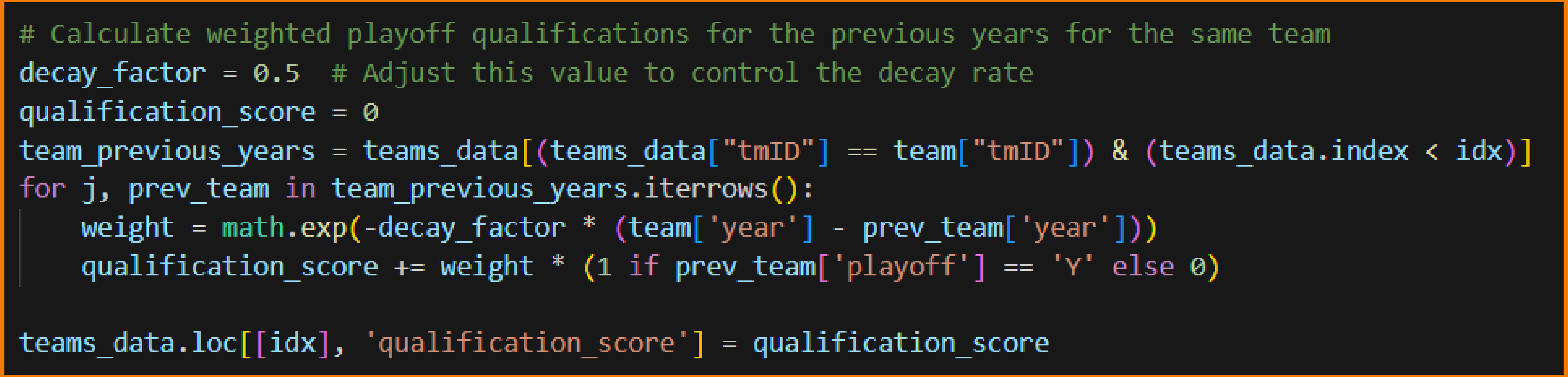


```
uPER = (1 / MP) *
     [ 3P
     + (2/3) * AST
     + (2 - factor * (team_AST / team_FG)) * FG
     + (FT *0.5 * (1 + (1 - (team_AST / team_FG)) + (2/3) * (team_AST / team_FG)))
     - VOP * TOV
     - VOP * DRB% * (FGA - FG)
     - VOP * 0.44 * (0.44 + (0.56 * DRB%)) * (FTA - FT)
     + VOP * (1 - DRB%) * (TRB - ORB)
     + VOP * DRB% * ORB
     + VOP * STL
     + VOP * DRB% * BLK
     - PF * ((lg_FT / lg_PF) - 0.44 * (lg_FTA / lg_PF) * VOP) ]
```

$$PER = \left(uPER \times \frac{lgPace}{tmPace} \right) \times \frac{15}{lguPER}$$

Creation of variable that assigns weights to the team's playoff qualifications based on the previous years

Other approach was to assign higher weights to teams who qualifed in more recent years and gradually decrease the weight for older years.



```
# Calculate weighted playoff qualifications for the previous years for the same team
decay_factor = 0.5  # Adjust this value to control the decay rate
qualification_score = 0
team_previous_years = teams_data[(teams_data["tmID"] == team["tmID"]) & (teams_data.index < idx)]
for j, prev_team in team_previous_years.iterrows():
    weight = math.exp(-decay_factor * (team['year'] - prev_team['year']))
    qualification_score += weight * (1 if prev_team['playoff'] == 'Y' else 0)

teams_data.loc[[idx], 'qualification_score'] = qualification_score
```

This code calculates a qualification score for each team based on the weighted sum of their past qualifications. We used this 'qualification_score' column as a variable that represents the team's playoff qualifications, weighted based on historical performance. Adjusting the decay factor allowed to control how much importance recent years have compared to older ones.

Experimental setup

It was decided that to solve the problem presented, it would be necessary to develop several models.

- For this purpose, the following algorithms were chosen: SVM, Naive Bayes, Decision Tree, Neural Network and Logistic Regression.
- It was also decided to divide the data into training/test data, using years 3 to 8 for training and 9 and 10 for testing.
- In order to evaluate the performance of the generated models, the values of accuracy, precision, recall and f1 score will be calculated.
- Test will be performed using diffent combinations of the created features: with PER or full players data; full or reduced of last two years teams data and with or without confederation division and limitation of only 4 positive answers for each (closer to reality).

Results

Using players information from last two years, but without team information.

Model	Accuracy	F1	Precision	Recall
Decision Tree	0.84615	0.875	0.875	0.875
SVM	0.61538	0.76190	0.61538	1.0
Naive Bayes	0.61538	0.70588	0.66667	0.75
Neural Network	0.61538	0.70588	0.66667	0.75
Logistic Regression	0.61538	0.70588	0.66667	0.75

Results

Using PER (player efficiency rating), spliting data between conferences and choosing the top four with highest probabilities.

Model	Accuracy Medium	F1 Medium	Precision Medium	Recall Medium
Decision Tree	0.6905	0.75	0.75	0.75
SVM	0.5476	0.625	0.625	0.625
Naive Bayes	0.6905	0.75	0.75	0.75
Neural Network	0.6905	0.75	0.75	0.75
Logistic Regression	0.5476	0.625	0.625	0.625

Conclusion

- Analysing the results obtained with the data enriched with newly created variables, it is possible to see a substancial improving in the model created using the new features, comparing with the original ones.
- While most of our tests bring the decision tree as the best model, it is necessary to be careful since it is prone to create too complex trees, leading to bad generalization, in a process known as overfitting.
- The current results demonstrate that the predictions are evolving, but there can be added even more features to enhance the models. Also the obtained results are not yet proving what models and procedures are the most correct since the amount of tests are still small, but it can lead and guide for future procedures and improvement. Another thing to be tried would be the implementation of more complex models, since the ones used are all quite simple.

DP: assessment of dimensions of data quality

In order to assure the quality of the data received, some key properties were verified:

- Completeness: The are some values missing from some players information, such as position, weight and height.
- Accuracy: Focuses on how close the data values are to the true values. Since the data received was obtained from WNBA real values, the accuracy is garanteed.
- Consistency: Only one data origin was used, so the consistency is also assured.
- Precision: Precision focuses on the level of detail or granularity present in the data. Since the dataset provides information about each season of each player for 10 years, it is considered assured.
- Uniqueness: No duplicated values were found, so the uniqueness is assured.
- Timeliness: The year of the capture of the data is unknown, so it cannot be verified.

Exploratory Analysis - 3+D plot

Observing the graph generated in the figure, we can draw conclusions. There is a clear increase in the number of points and blocks in the game over the years. It can be inferred that there is a great evolution in the speed of the game, leading to an increase in statistics related to both defense and attack, since the ball moves faster.

Another, more obvious, relationship is between the playoff and points and blocks: the more points and blocks, the more likely it is to qualify for the playoff. Finally, the positive relationship between points and blocks allows us to affirm that a team that defends better has a greater chance of scoring more points and, consequently, going further in the competitions.

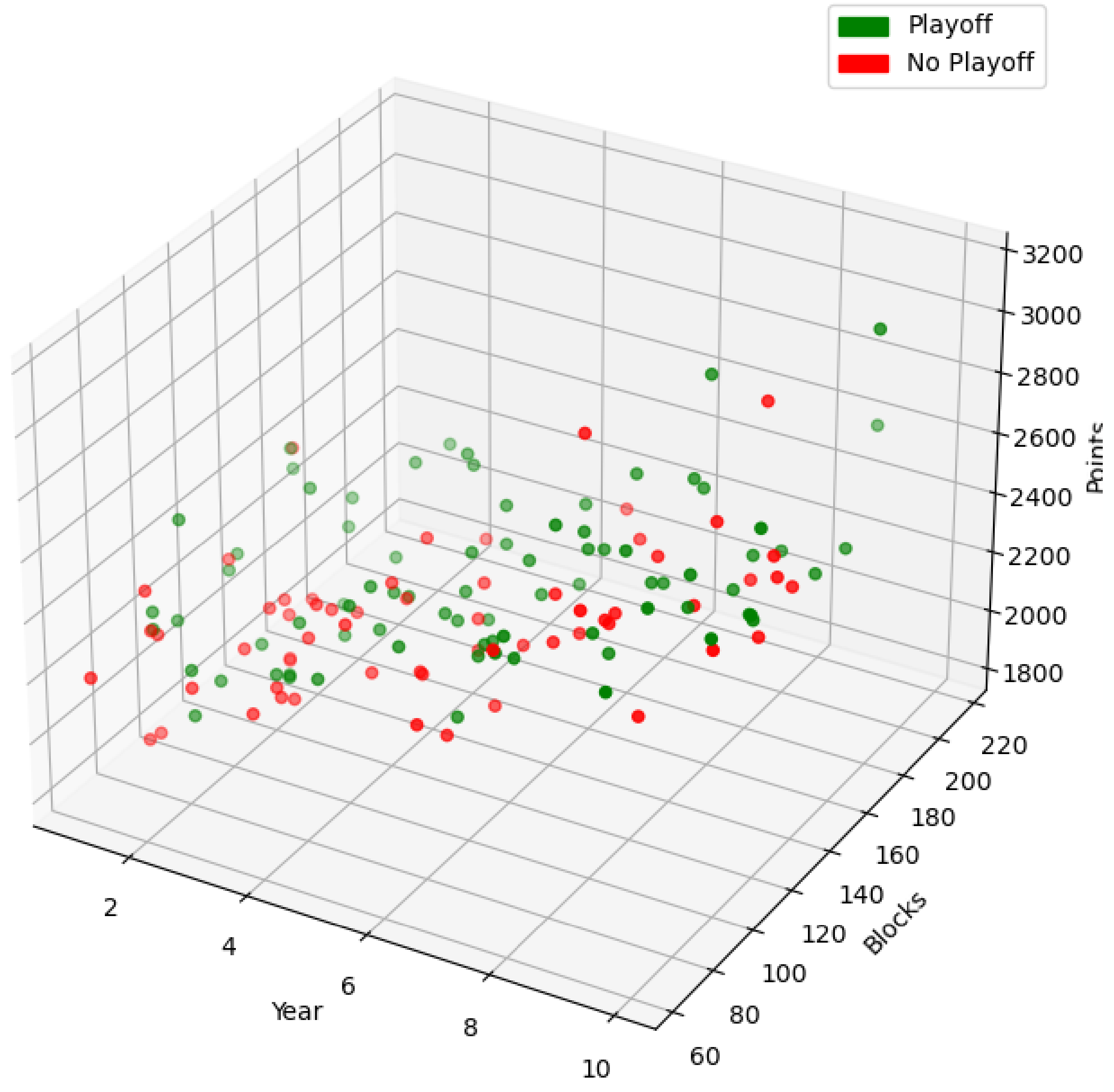


Figure - 4D Scatter Plot of Year, Blocks, Playoff and Points

Algorithms - Decision Tree

Decision Tree is a tree-like model where internal nodes represent features, branches represent decision rules, and leaves represent outcomes. It is intuitive, easy to interpret, and can handle both classification and regression tasks effectively.

During the testing, some of the parameters were tunned in order to achieve the best result possible according to the objective of the project. The selected parameters were the following:

- **Criterion: Gini** or **Entropy**
- **Max Depth:** Determines the maximum depth of the decision tree. A range between 3 and 8 was selected.
- **Min Samples Split:** A range from 10 to 110 with increments of 10.
- **Min Samples Leaf:** A range from 1 to 10 was selected.
- **Max Features:** Controls the number of features considered for splitting at each node.
    - **'sqrt':** Considers the square root of the total features.
    - **'log2':** Considers the logarithm base 2 of the total features.
    - **None:** Considers all features.
- **Splitter:**
    - **Best:** Chooses the best split based on impurity reduction.
    - **Random:** Chooses the best random split. Useful for reducing overfitting.

Algorithms - SVM

Support Vector Machine is a powerful supervised learning algorithm used for both classification and regression tasks. It works by finding the optimal hyperplane that best separates data points in a high-dimensional space. The parameters selected for tuning in order to achieve better results were:

- **C (Regularization Parameter)**: Controls the trade-off between smooth decision boundaries and classifying training points correctly. Options: [0.1, 1, 10].
- **Gamma (Kernel Coefficient**): Defines how far the influence of a single training example reaches. Options: [1, 0.1, 0.01].
- **Kernel Function**:
  - '**rbf**' (Radial basis function): Suitable for non-linear separation.
  - '**linear**': Suitable for linear separation.

Algorithms - Naive Bayes

Naive Bayes is a probabilistic machine learning algorithm based on Bayes' theorem. It is commonly used for classification tasks and assumes independence among features. The chosen implementation is Gaussian Naive Bayes. The parameter considered is:

- **Var Smoothing**: This parameter represents the portion of the largest variance of all features that is added to variances for calculation stability. A range of values was explored using a logarithmic scale from 1 to 1e-9.

Algorithms - Neural Networks

Neural Networks are versatile machine learning models inspired by the human brain's structure. The chosen implementation was the MLPClassifier.

The key parameters explored during the analysis are:

- **Hidden Layer Sizes**: represent the number of neurons in each hidden layer. (50, 50, 50)(50, 100, 50)(100,)
- **Activation Function**: determines the output of each neuron and impacts the network's ability to learn complex patterns.
    - 'tanh': Hyperbolic tangent function.
    - 'relu': Rectified Linear Unit.
- **Solver**: The solver defines the optimization algorithm used for weight updates during training.
    - 'sgd': Stochastic Gradient Descent.
    - 'adam': Adaptive Moment Estimation.
- **Alpha**: represents the regularization term that helps prevent overfitting by penalizing large weights.
    - 0.0001
    - 0.05
- **Learning Rate**:
    - 'constant': Maintains a constant learning rate.
    - 'adaptive': Adjusts the learning rate based on the network's performance.

Algorithms - Logistic Regression

Logistic Regression is a fundamental and interpretable algorithm commonly used for binary classification tasks.

The parameters explored during the analysis were:

- **C** (Regularization Parameter): controls the inverse of the regularization strength. Smaller values of C indicate stronger regularization, helping prevent overfitting.
  - np.logspace(-4, 4, 20)
- **Solver**: The solver defines the optimization algorithm used for weight updates during training.
  - 'liblinear'

Attachments B - Year 9 - Full Players Data - Full Teams Data

Model	Accuracy	F1	Precision	Recall
Decision Tree	0.84615	0.88888	0.8	1.0
SVM	0.61538	0.76190	0.61538	1.0
Naive Bayes	0.61538	0.66666	0.71428	0.625
Neural Network	0.76923	0.84210	0.77778	0.8
Logistic Regression	0.61538	0.73684	0.63636	0.875

Table 1 - Different Models Metrics

Attachments B - Year 9 - Full Players Data - Full Teams Data

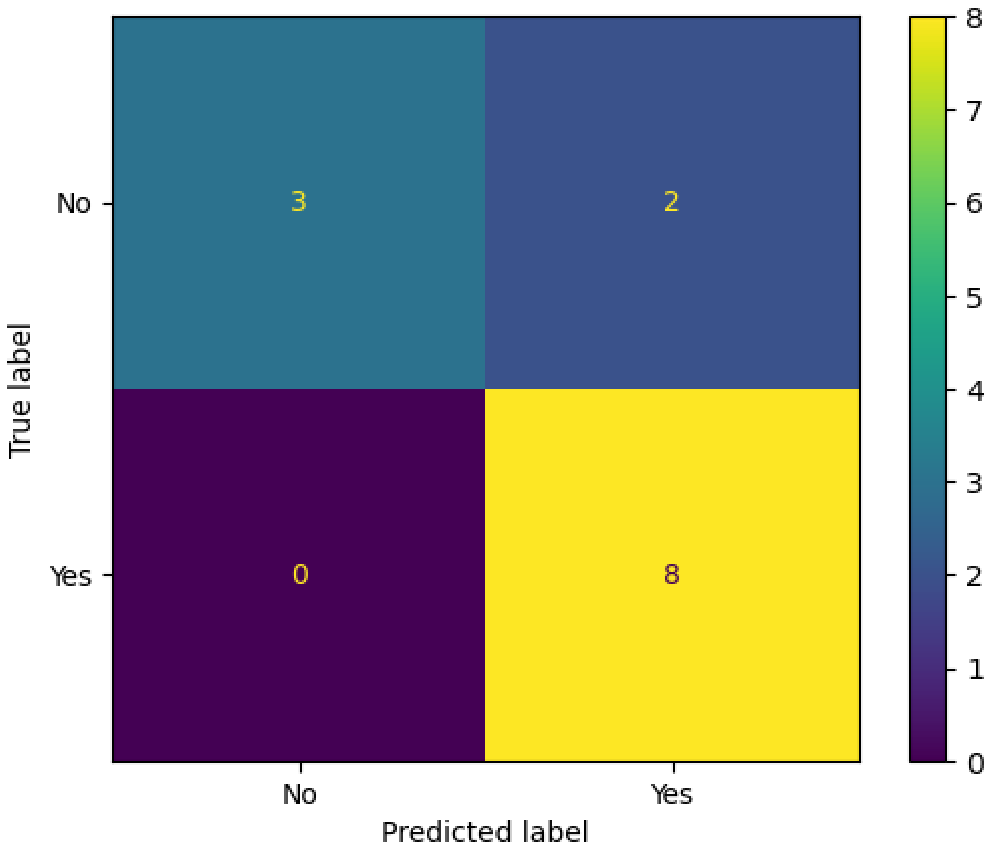


Figure 1 - Decision Tree Confusion Matrix

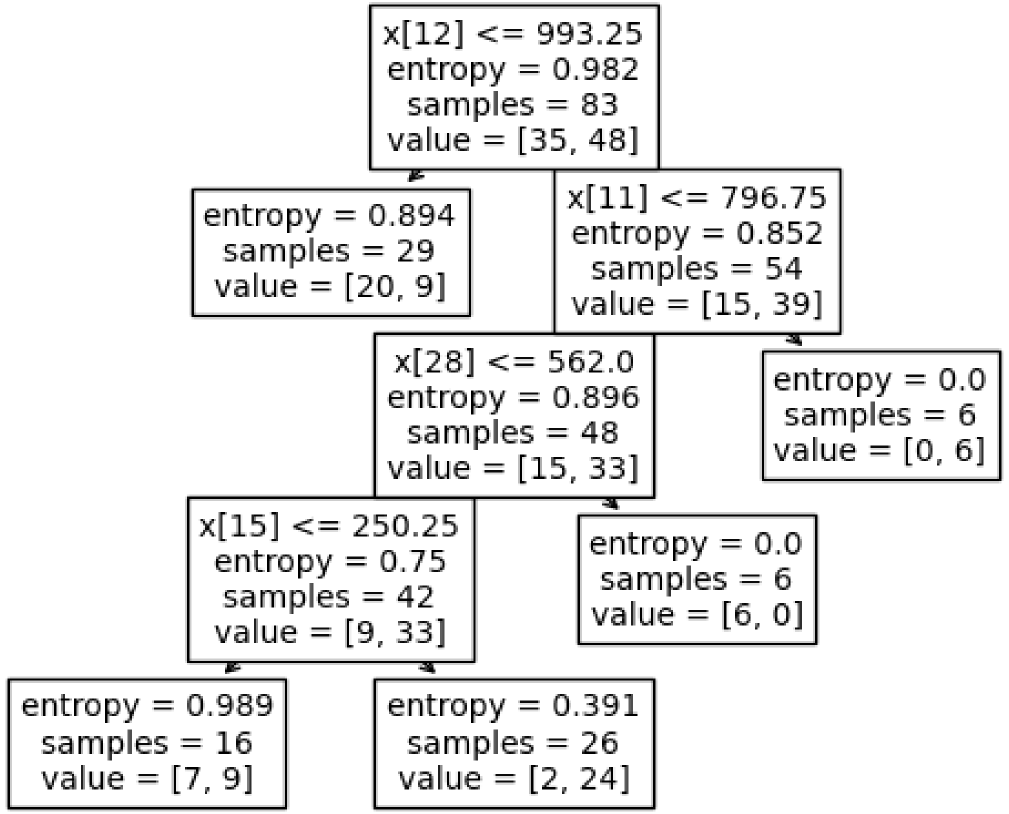


Figure 6 -Decision Tree

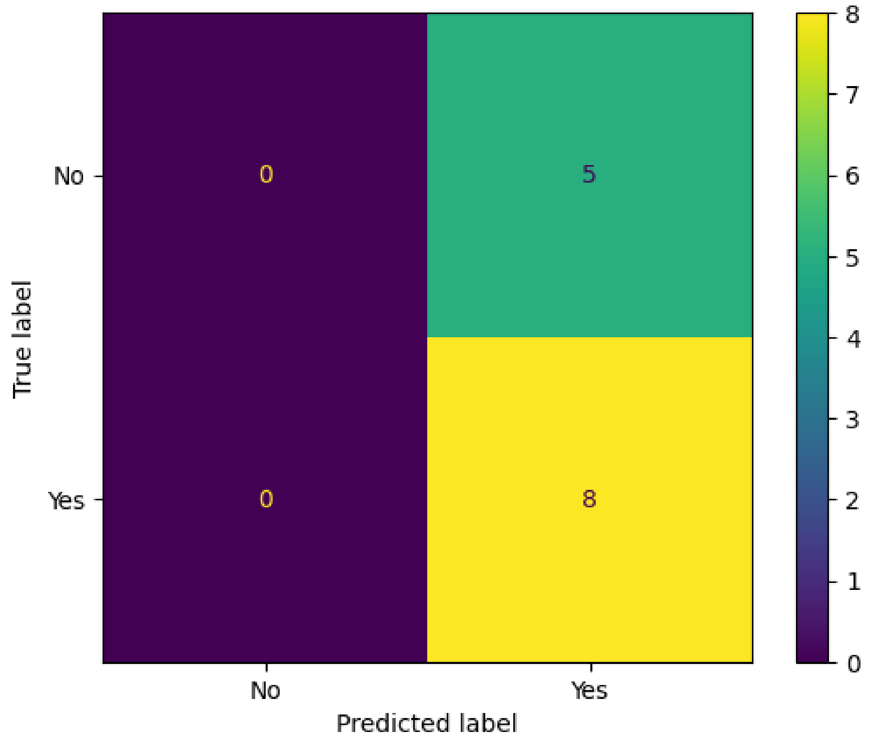


Figure 2 - SVM Confusion Matrix

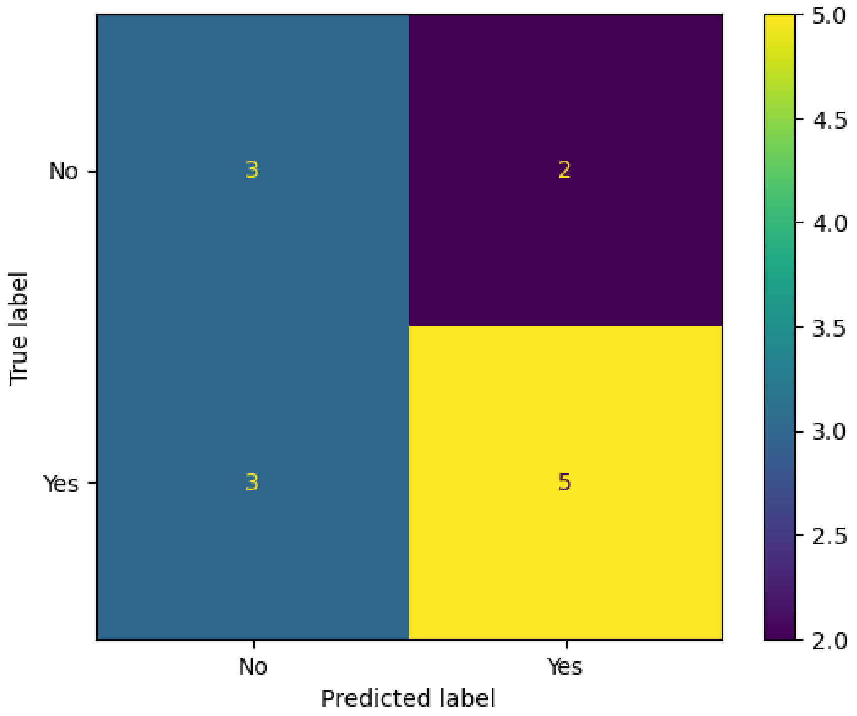


Figure 3 - Naive Bayes Confusion Matrix

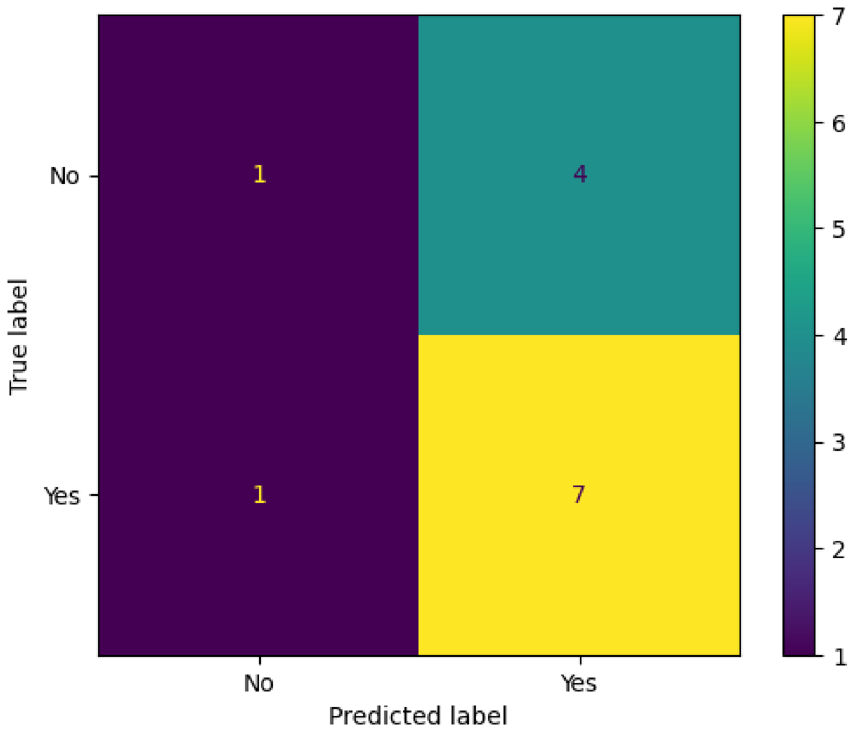


Figure 5 - Logistic Regression Confusion Matrix

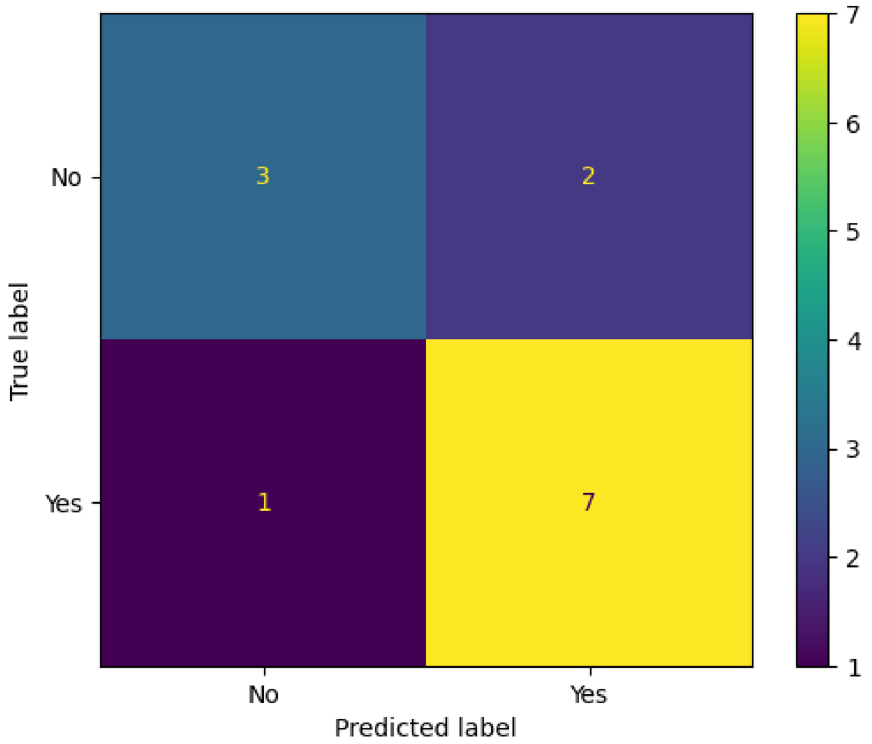
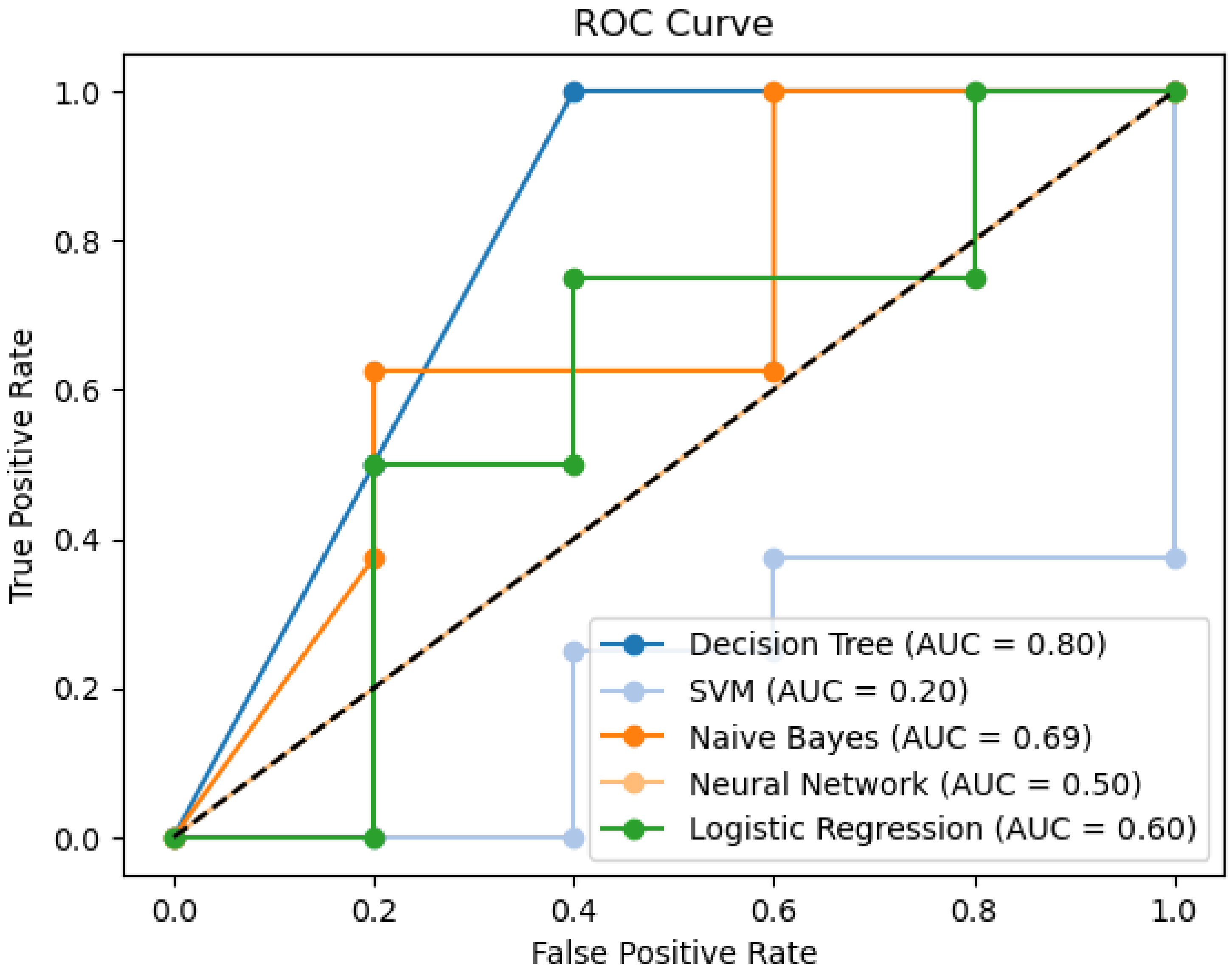


Figure 4 - Neural Networks Confusion Matrix

Attachments B - Year 9 - Full Players Data - Full Teams Data



In the graph, the models created used the following parameters:

Decision tree:

- criterion': 'entropy', 'max_depth': 5, 'max_features': log2, 'max_leaf_nodes': None, 'min_impurity_decrease': 0.0, 'min_samples_leaf': 2, 'min_samples_split': 40,'splitter': 'best'

SVM:

- 'C': 1.0,'gamma': 'scale', 'kernel': 'rbf'

Naive Bayes:

- 'var_smoothing': 1e-09

Neural Network:

- {'activation': 'relu', 'alpha': 0.05,'hidden_layer_sizes': (100,), 'learning_rate': 'adaptive,'solver': 'adam'

Logistic Regressions:

- 'C': 0.033598,'solver': 'liblinear’

Attachments C - Year 10 - Full Players Data - Full Teams Data

Model	Accuracy	F1	Precision	Recall
Decision Tree	0.53846	0.5	0.75	0.375
SVM	0.61538	0.76190	0.61538	1.0
Naive Bayes	0.53846	0.625	0.625	0.625
Neural Network	0.69230	0.8	0.66667	1.0
Logistic Regression	0.53846	0.57143	0.66667	0.5

Table 1 - Different Models Metrics

Attachments C - Year 10 - Full Players Data - Full Teams Data

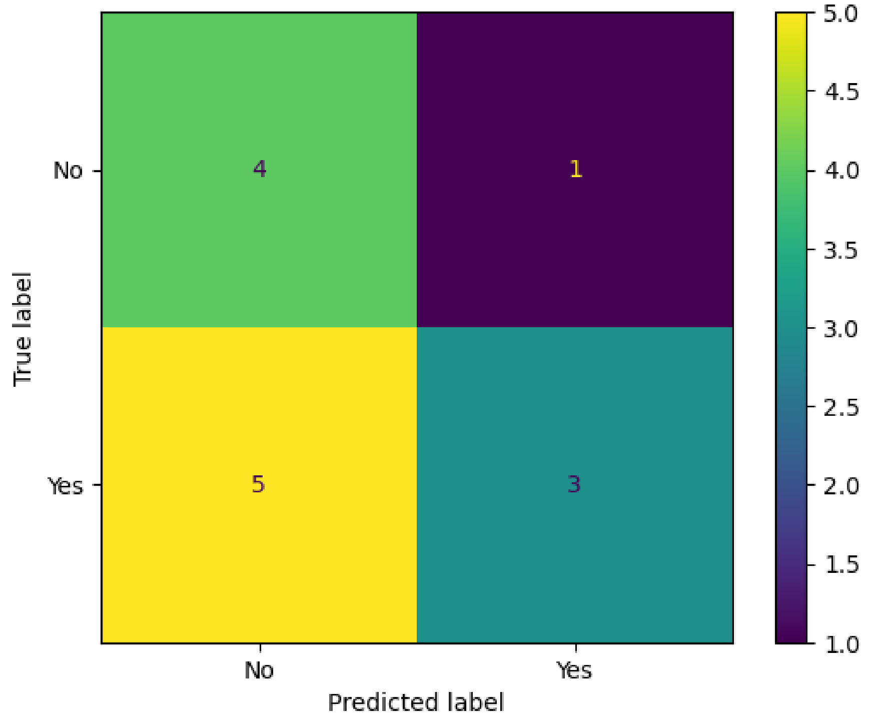


Figure 1 - Decision Tree Confusion Matrix

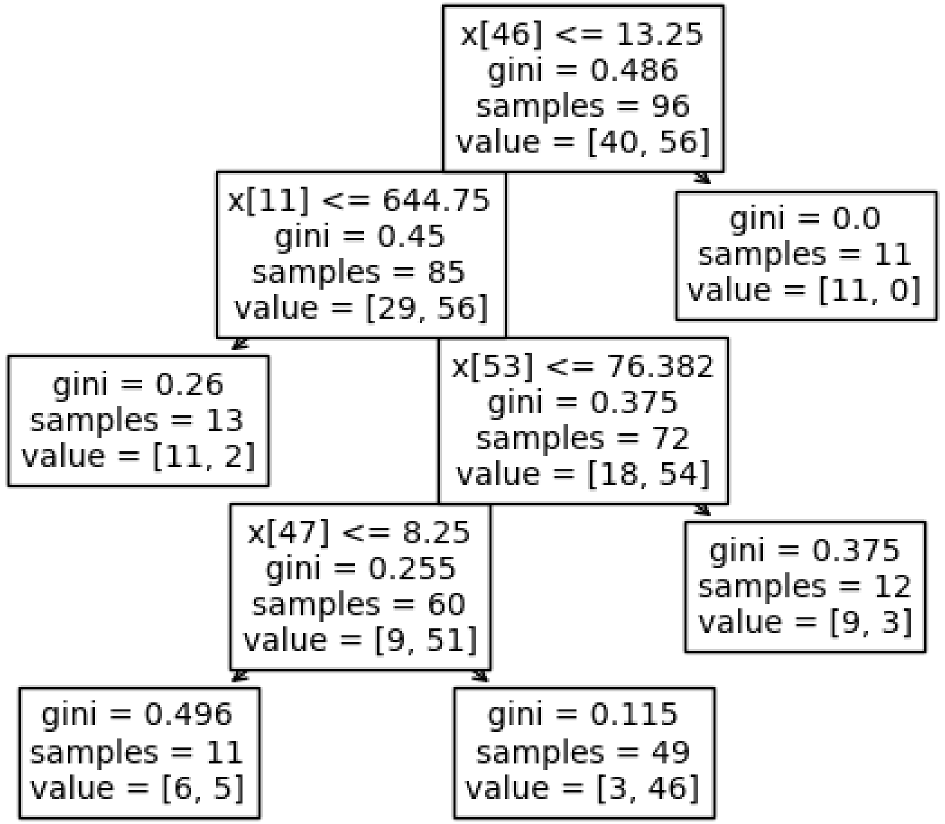


Figure 5 -Decision Tree

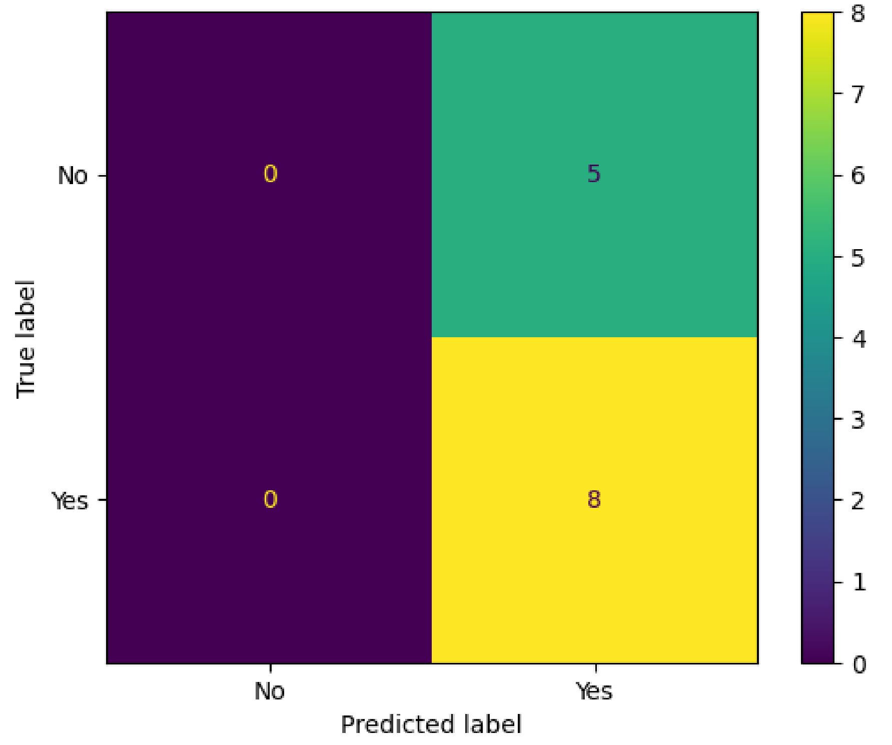


Figure 2 - SVM Confusion Matrix

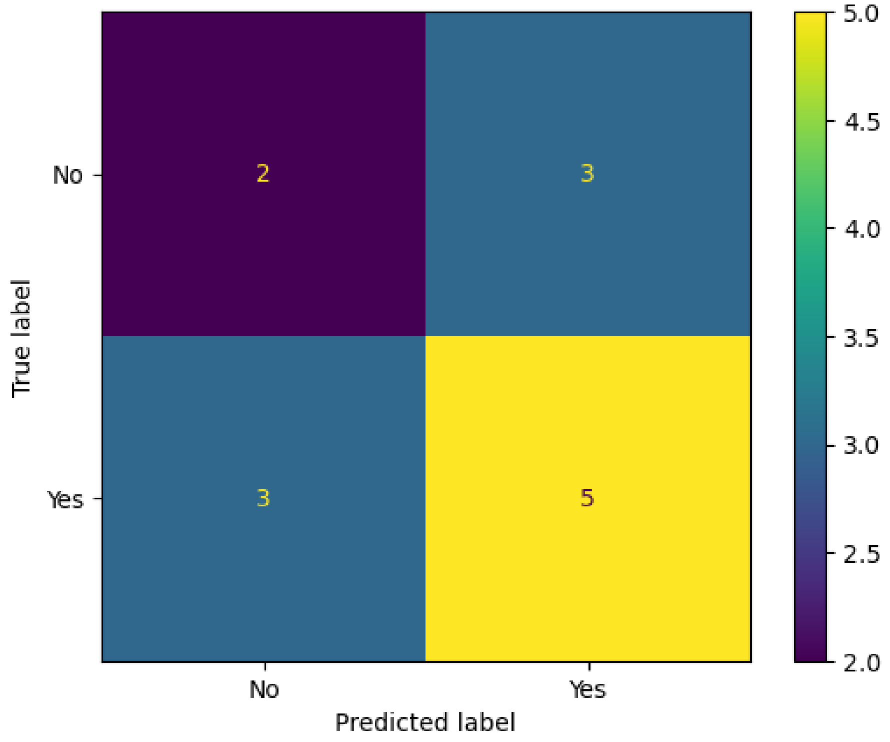


Figure 3 - Naive Bayes Confusion Matrix

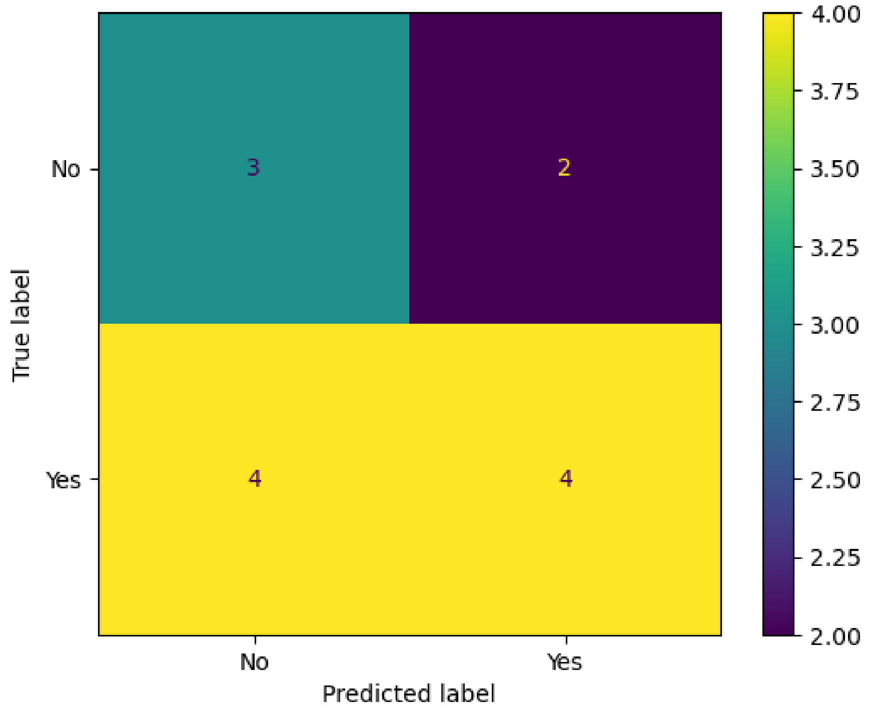


Figure 5 - Logistic Regression Confusion Matrix

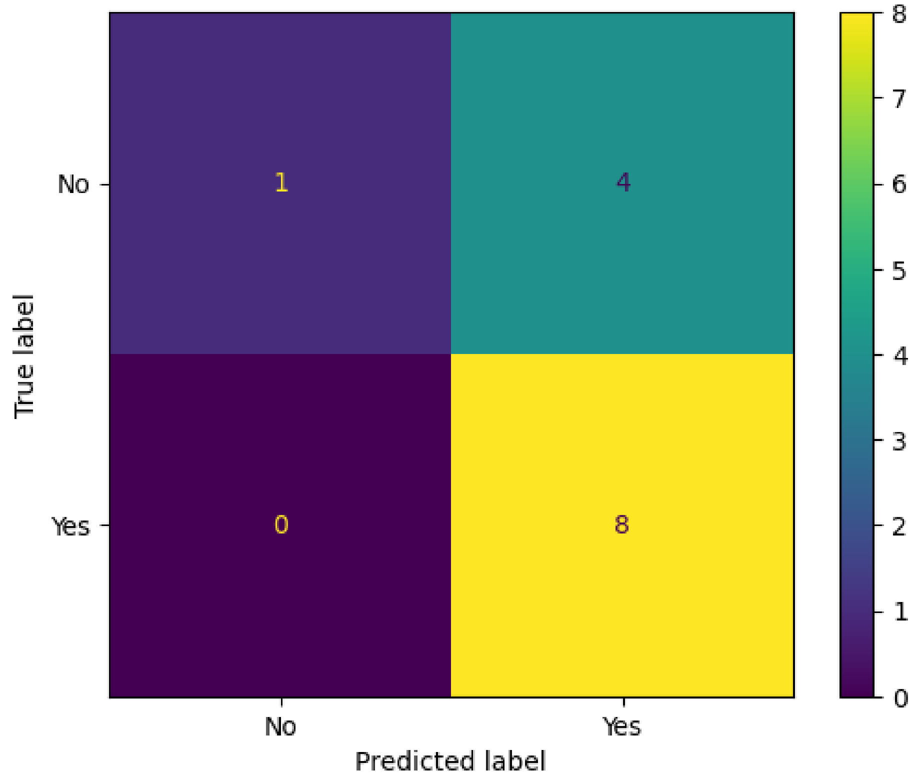
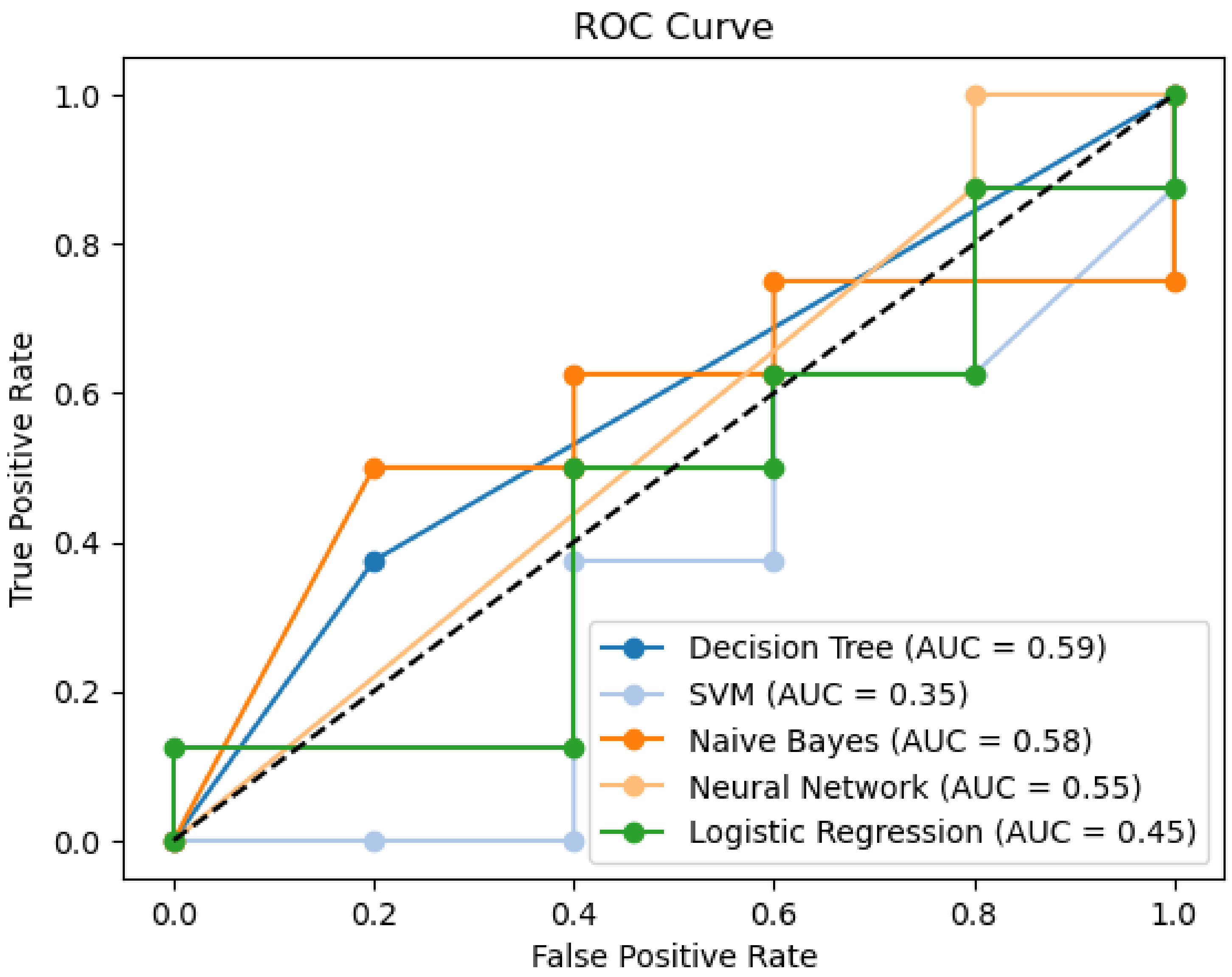


Figure 4 - Neural Networks Confusion Matrix

Attachments C - Year 10 - Full Players Data - Full Teams Data



Analyzing the last two graphs and results, it is visible that the models developed have better performance in year 9 compared to year 10, with the exception of SVM.

In year 9, the model created based on the Decision Tree algorithm achieved a good performance and the one based on Neural Network achieved also a satisfying result. There are no clear good models to predict year 10, all of them with ROC very near the random choice.

Attachments D - Year 10 - Full Players Data - Reduced Teams Data

Model	Accuracy	F1	Precision	Recall
Decision Tree	0.84615	0.875	0.875	0.875
SVM	0.61538	0.76190	0.61538	1.0
Naive Bayes	0.61538	0.70588	0.66667	0.75
Neural Network	0.61538	0.70588	0.66667	0.75
Logistic Regression	0.61538	0.70588	0.66667	0.75

Table 1 - Different Models Metrics

Attachments D - Year 10 - Full Players Data - Reduced Teams Data

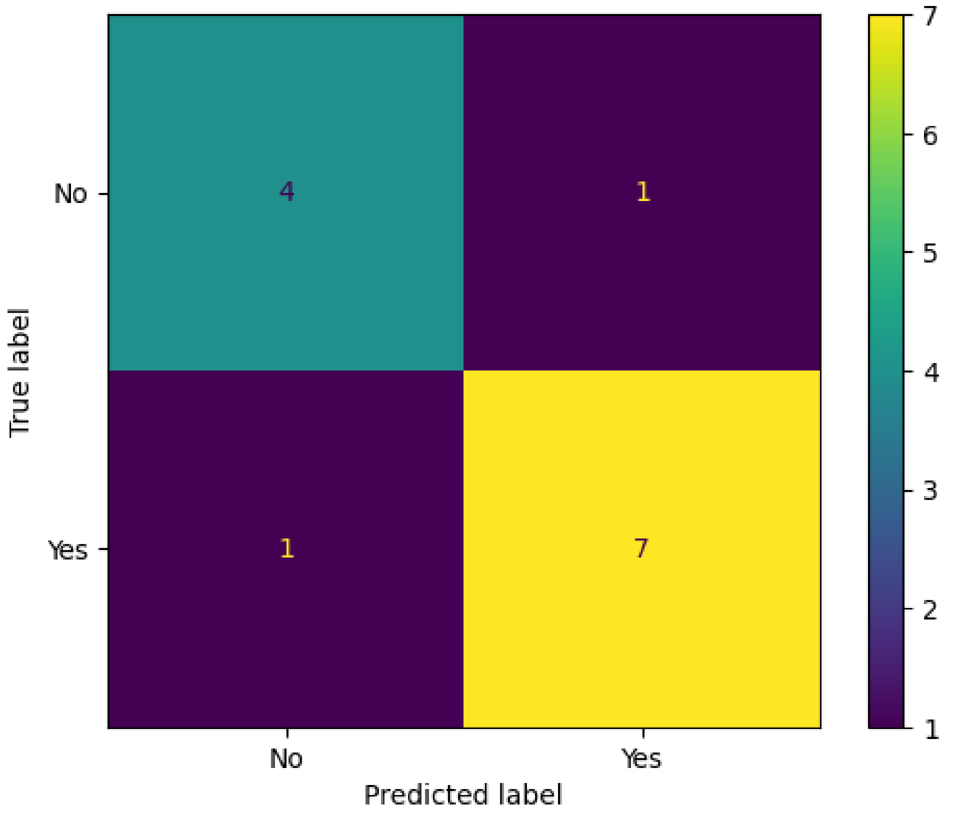


Figure 1 - Decision Tree Confusion Matrix

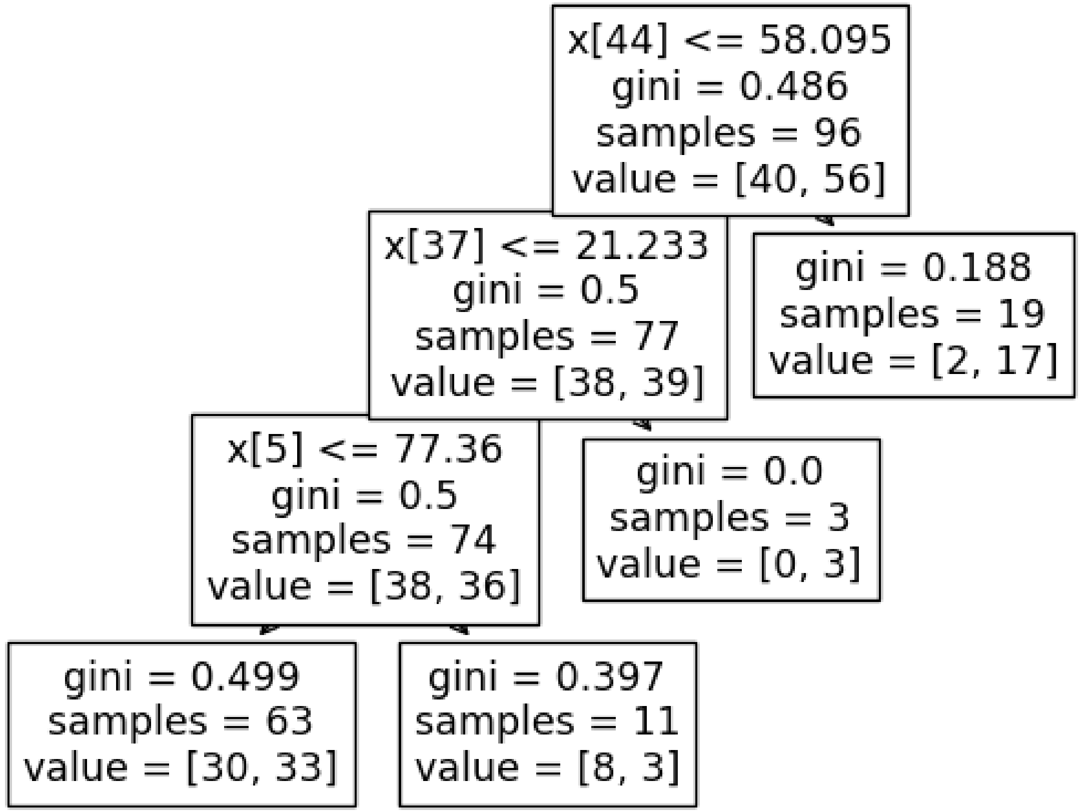


Figure 5 -Decision Tree

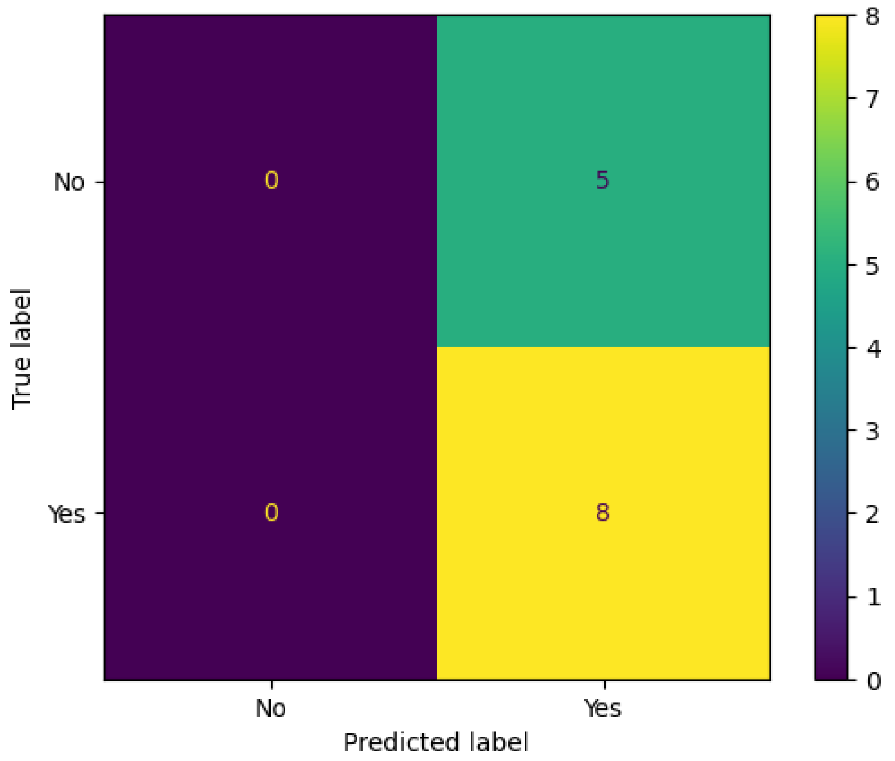


Figure 2 - SVM Confusion Matrix

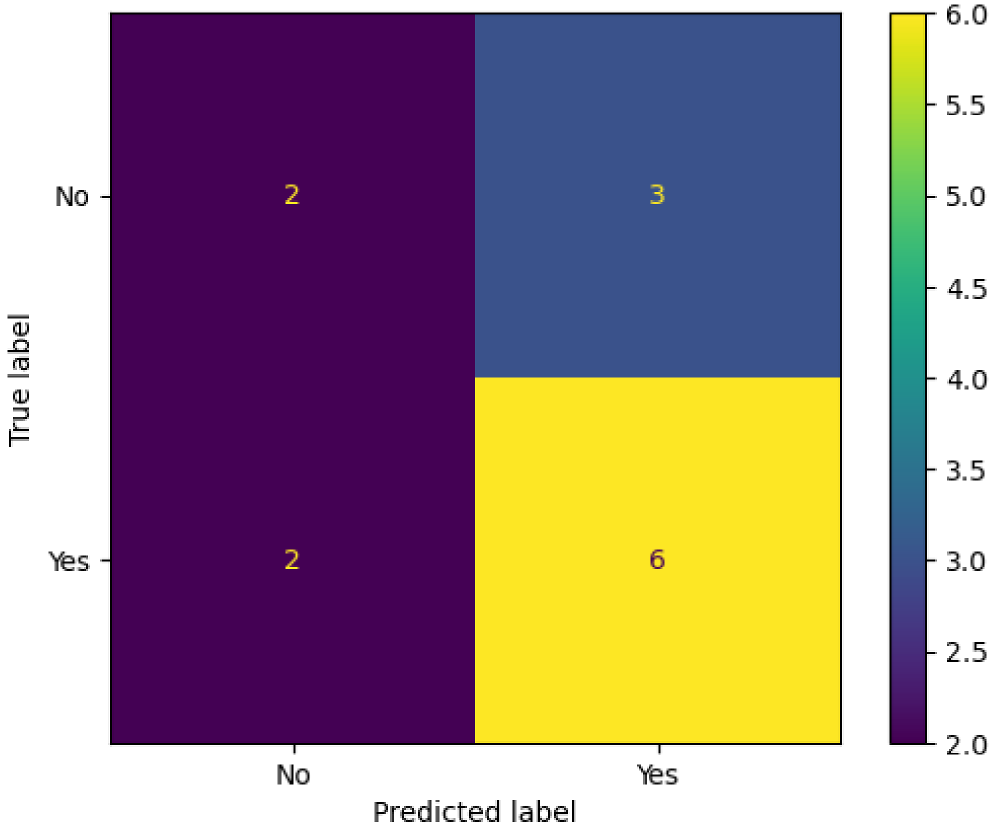


Figure 3 - Naive Bayes Confusion Matrix

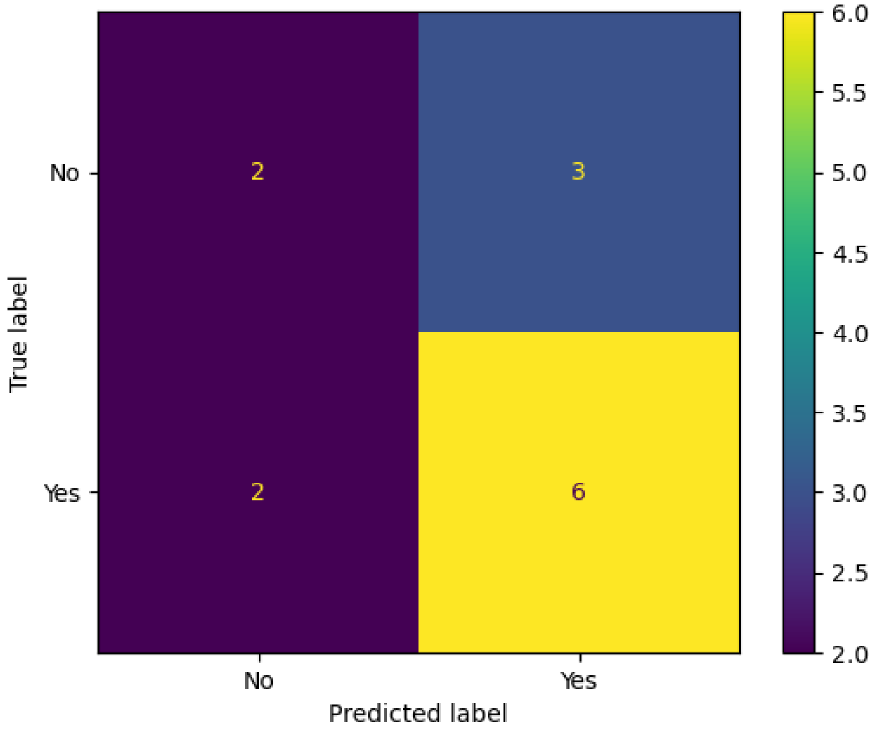


Figure 5 - Logistic Regression Confusion Matrix

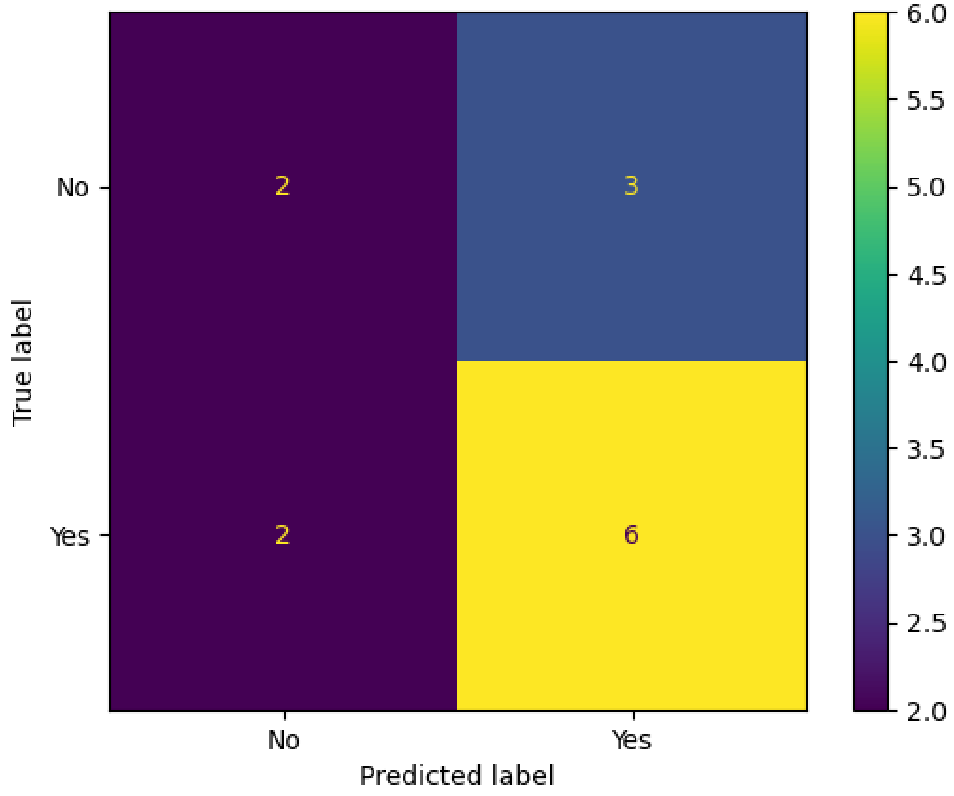
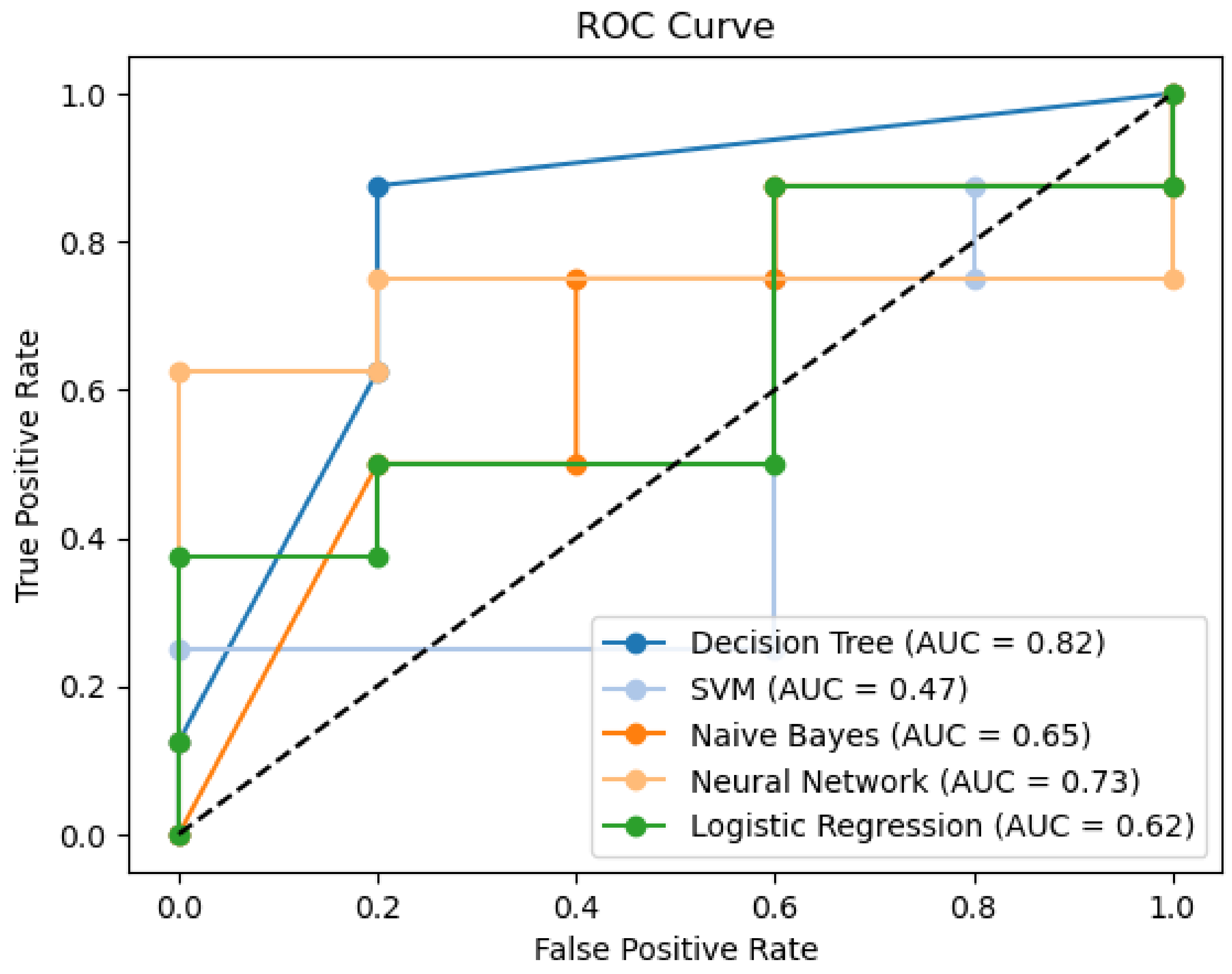


Figure 4 - Neural Networks Confusion Matrix

Attachments D - Year 10 - Full Players Data - Reduced Teams Data



This models were generated using a smaller number of features, using only more generic teams' values, like wins, points/game and removing the more specific statistics as blocks and rebounds, using only its values from the players and not team's past years. There is a clear improval in the all models, comparing to the one using full team data.

The models created using the Decision Tree and SVM algorithms weren't affected, meaning that those models didn't made use of the supressed information.

Attachments E - Year 10 - PER - Reduced Teams Data

Model	Accuracy	F1	Precision	Recall
Decision Tree	0.61538	0.73684	0.63636	0.875
SVM	0.61538	0.76190	0.61538	1.0
Naive Bayes	0.53846	0.625	0.625	0.625
Neural Network	0.69230	0.75	0.75	0.75
Logistic Regression	0.69231	0.77778	0.7	0.875

Attachments E - Year 10 - PER - Reduced Teams Data

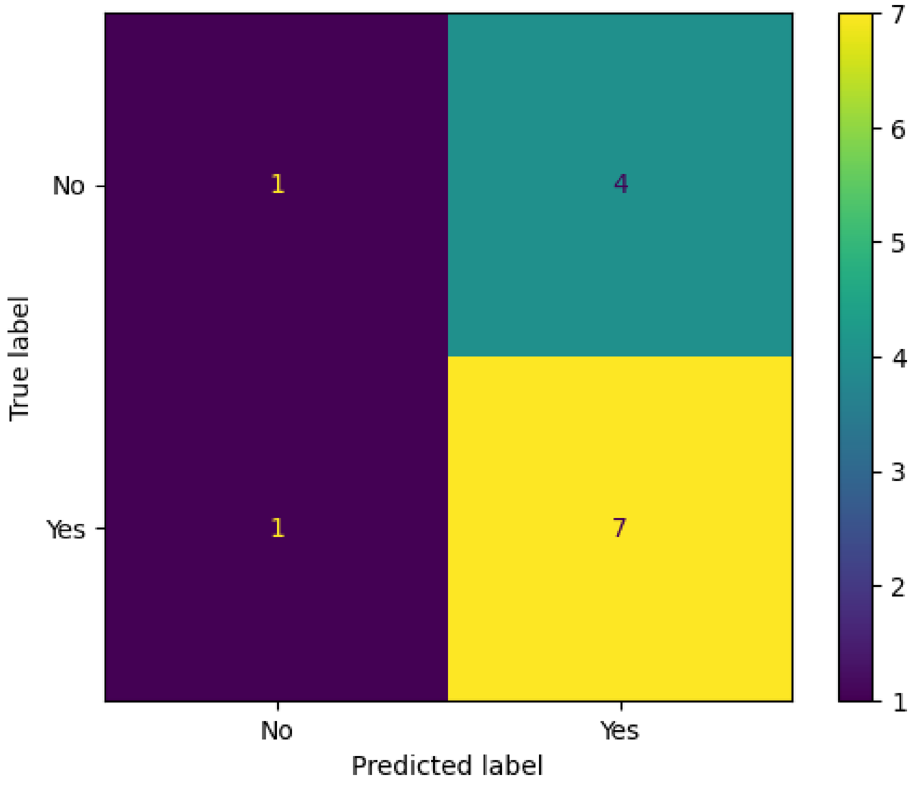


Figure 1 - Decision Tree Confusion Matrix

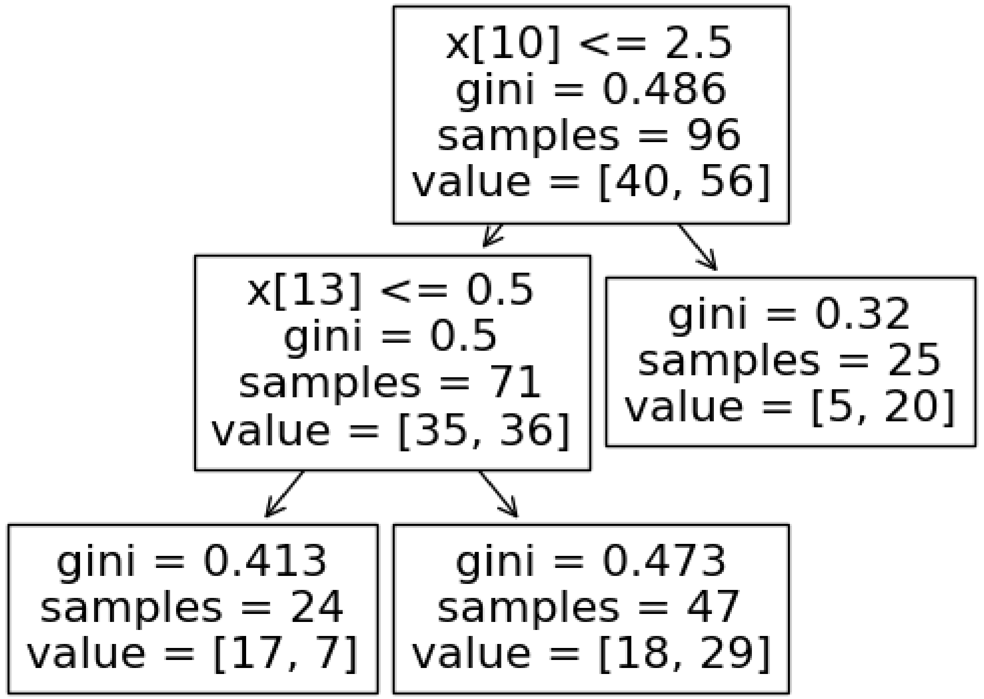


Figure 5 -Decision Tree

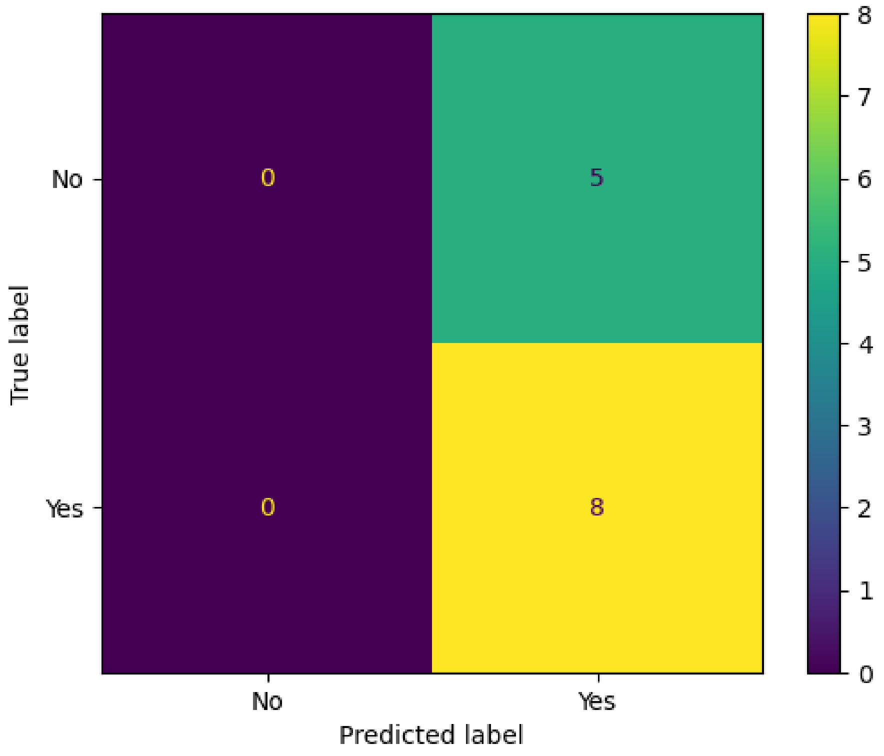


Figure 2 - SVM Confusion Matrix

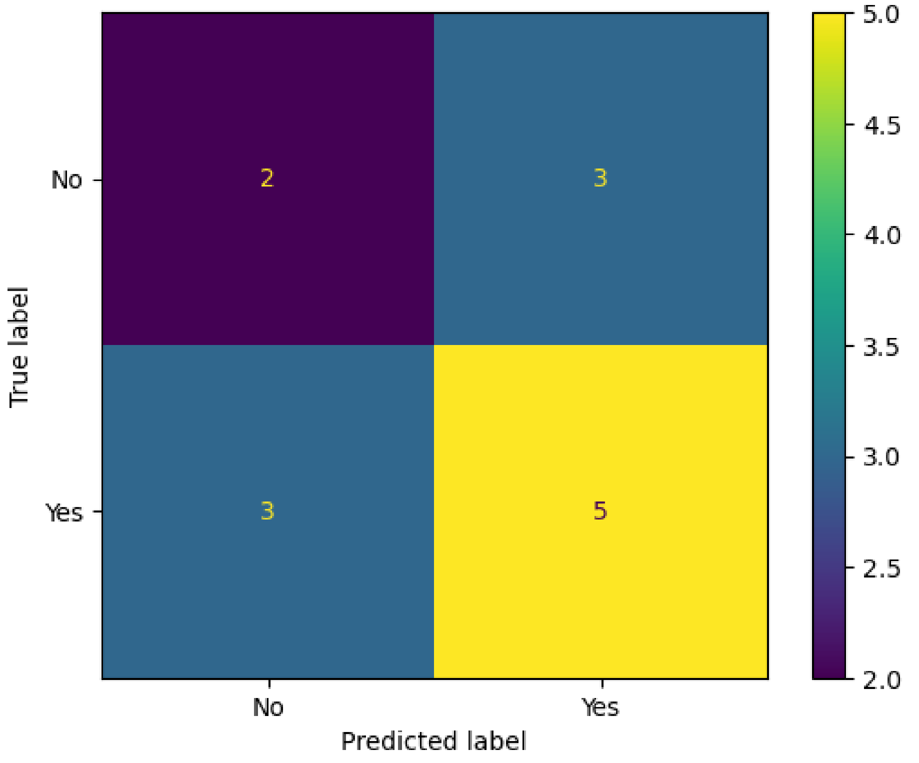


Figure 3 - Naive Bayes Confusion Matrix

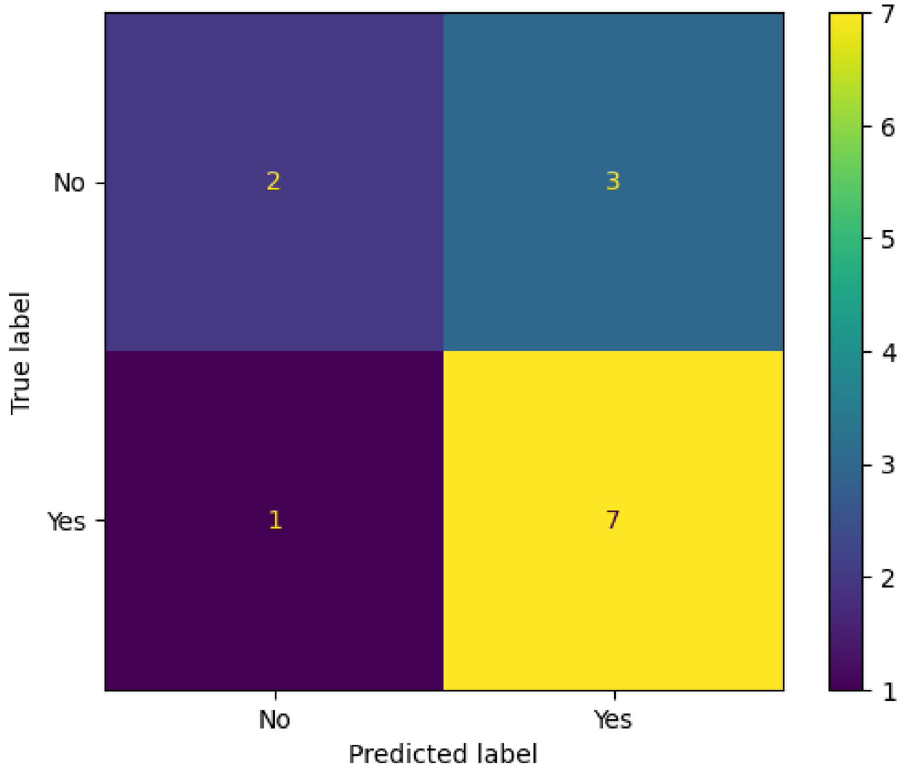


Figure 5 - Logistic Regression Confusion Matrix

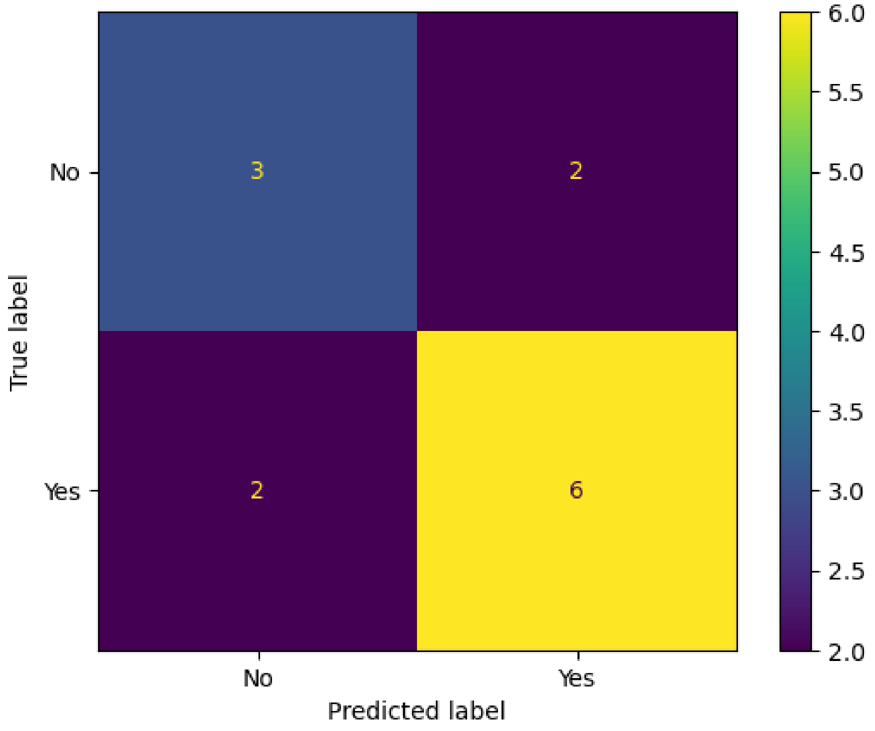
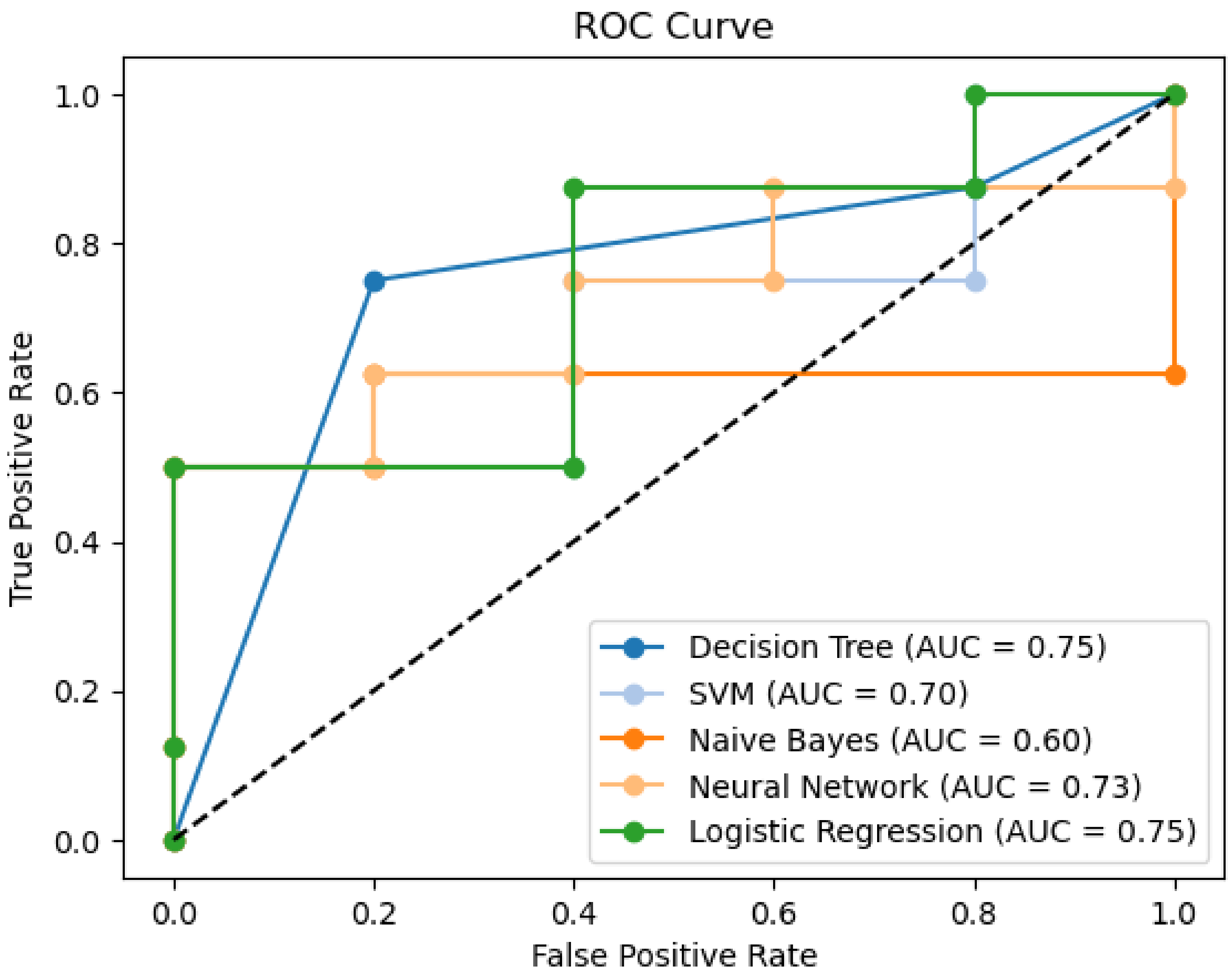


Figure 4 - Neural Networks Confusion Matrix

Attachments E - Year 10 - PER - Reduced Teams Data



In this models, the great difference was the substitution in the data of the players data columns (medium of the team players statics from last two years) for the average of the team players PER in the last two years. By replacing more than 10 columns with only one sumarizing all of them, better performances were obtained specialy in SVM based model. There is a clear improve visible in the ROC graph and in the AUC value. There is also a slighty better performance from the model created using Logistic Regression. On the negative side, there is a clear loss of performance from the model based on Decision Tree algorithm, which was the best one in previous tests.

Attachments F - Year 10 - PER - Reduced Teams Data - Confederation Separation and Limitation

Model	Accuracy Medium	F1 Medium	Precision Medium	Recall Medium
Decision Tree	0.6905	0.75	0.75	0.75
SVM	0.5476	0.625	0.625	0.625
Naive Bayes	0.6905	0.75	0.75	0.75
Neural Network	0.6905	0.75	0.75	0.75
Logistic Regression	0.5476	0.625	0.625	0.625

Attachments F - Year 10 - PER - Reduced Teams Data - Confederation Separation and Limitation

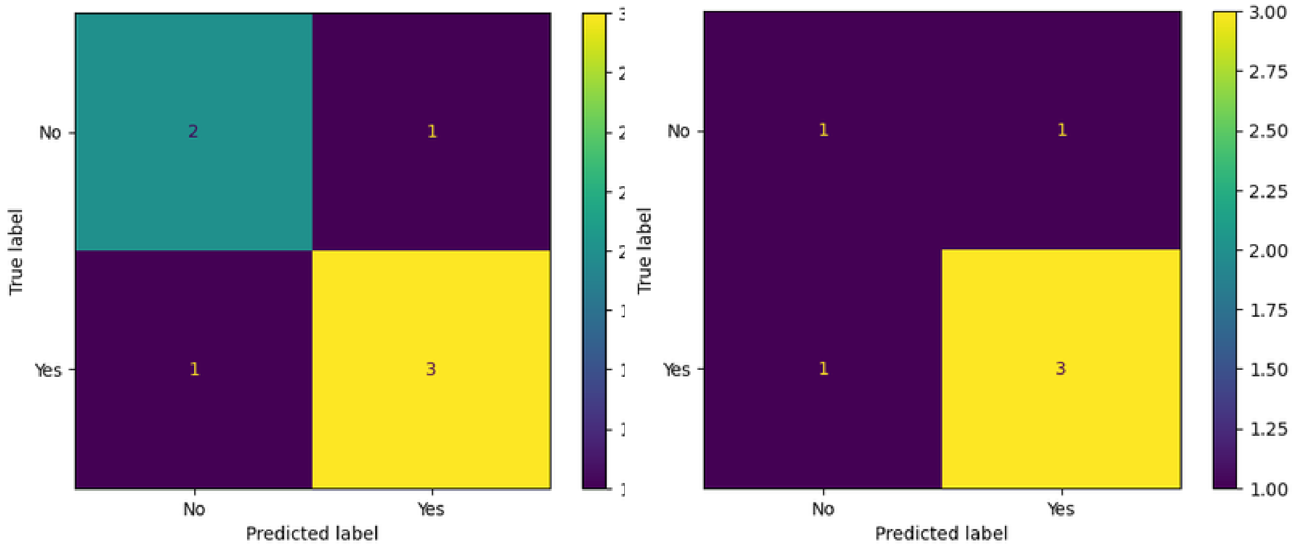


Figure 1 - Decision Tree Confusion Matrix
Left - EA Right - WE

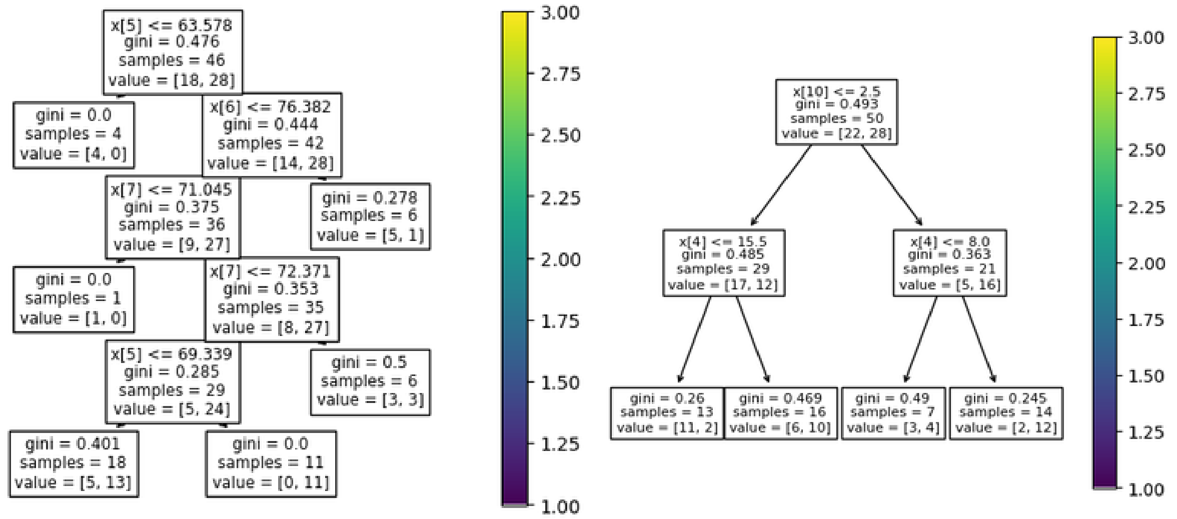


Figure 5 -Decision Tree
Left - EA Right - WE

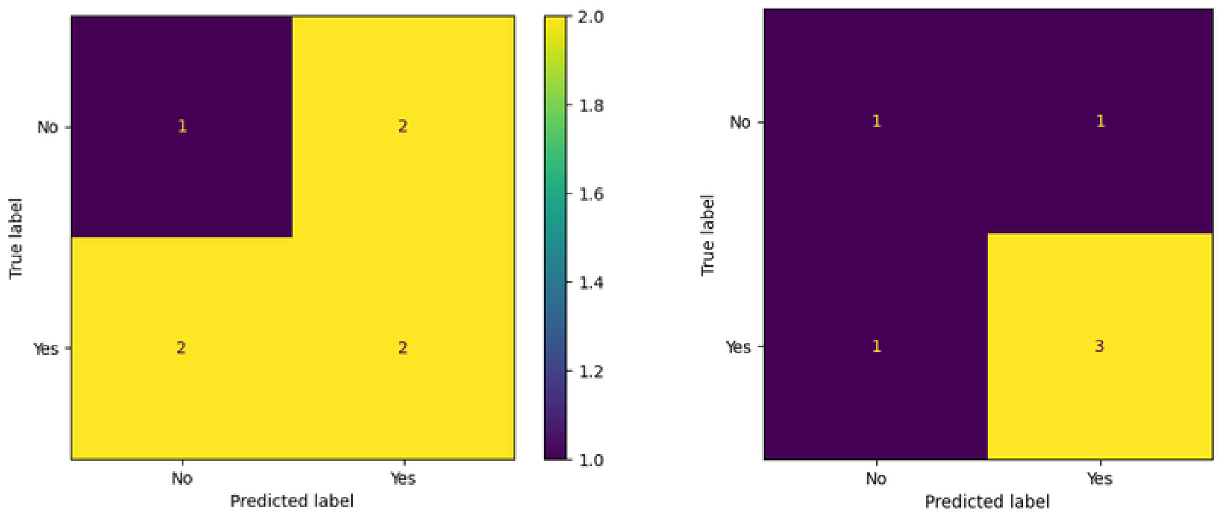


Figure 2 - SVM Confusion Matrix
Left - EA Right - WE

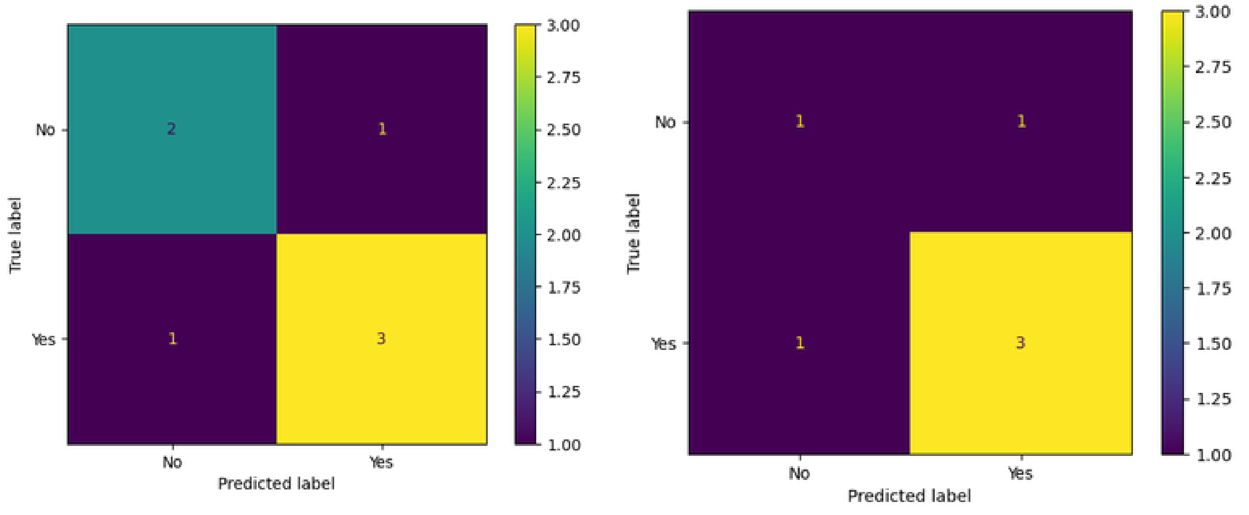


Figure 3 - Naive Bayes Confusion Matrix
Left - EA Right - WE

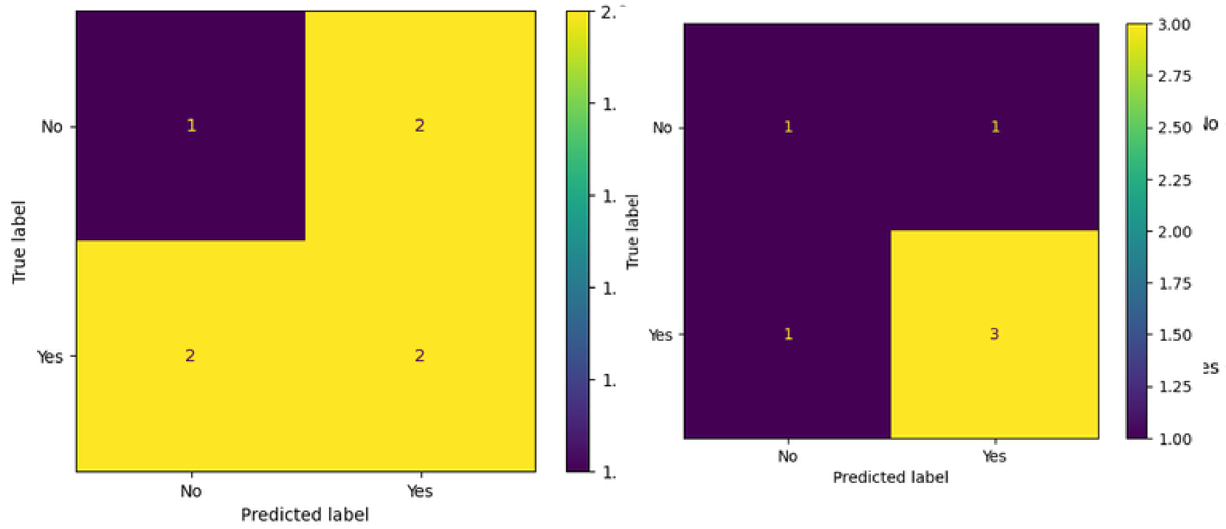


Figure 5 - Logistic Regression Confusion Matrix
Left - EA Right - WE

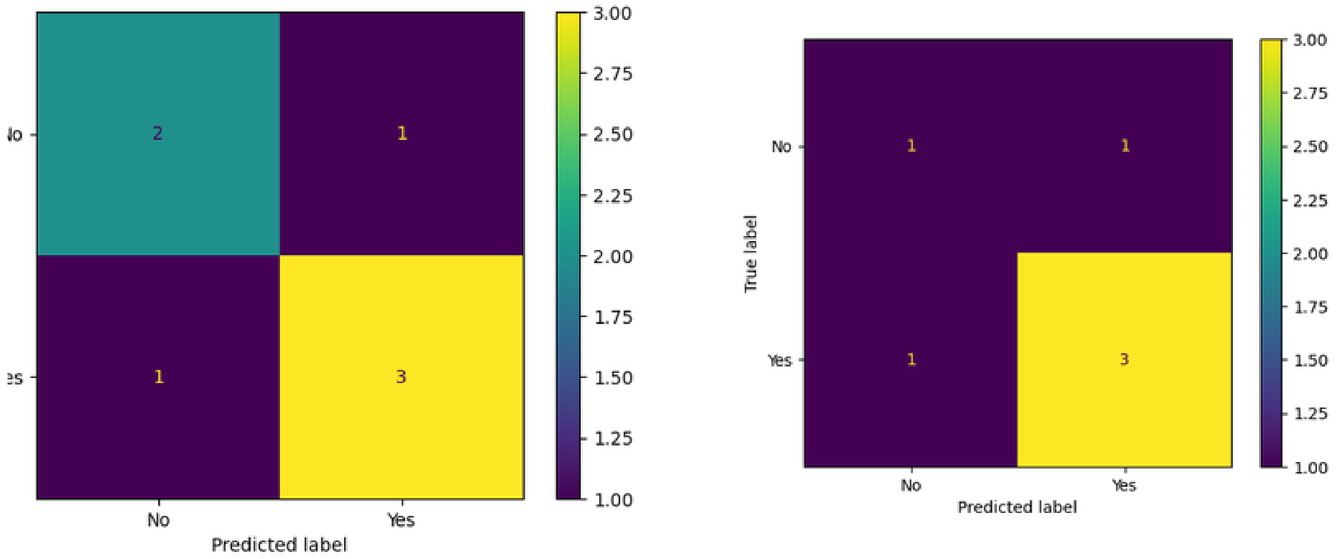
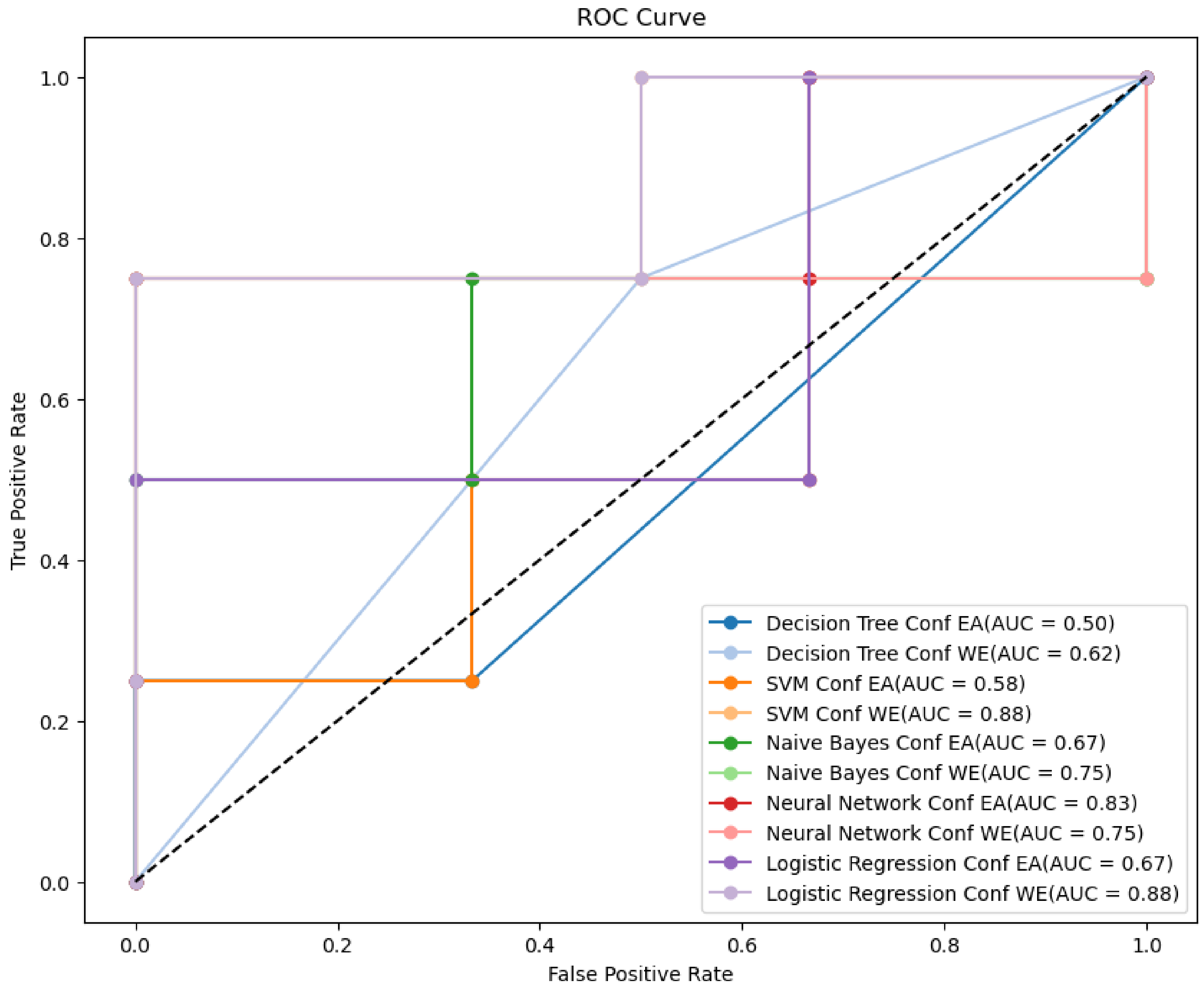


Figure 4 - Neural Networks Confusion Matrix
Left - EA Right - WE

Attachments F - Year 10 - PER - Reduced Teams Data - Confederation Separation and Limitation



For the final test, it were implemented limitations so that the test could be the most real possible. This way the test/train set were divided in Confederation and the Classification responses were made using the probabilies with a limitation of no more and no less than 4 positive classifications for playoff.

In most of the models there is a clear better performance in the WE conference. Since this test imposes a limit to Yes's, the general performances were worst. This is due to the fact that, without it, a model could have the tendency to mostly give a positive answer. Since 8 of the 13 values must positive, a model that always responded positive would have a 61% accuracy. Using this limitation and separation, that tendency wont be present.