

Basketball Playoffs Qualification WBNA

Data Mining Project



G66

Gonçalo de Abreu Matias (up202108703) (IF = 1)
Tomás Martim Santos Monteiro (up202109646) (IF = 1)

TABLE OF CONTENTS



01

DOMAIN DESCRIPTION

PREDICTIVE DATA
MINING PROBLEM

03



02

EXPLORATORY DATA
ANALYSIS

KAGGLE RESULTS



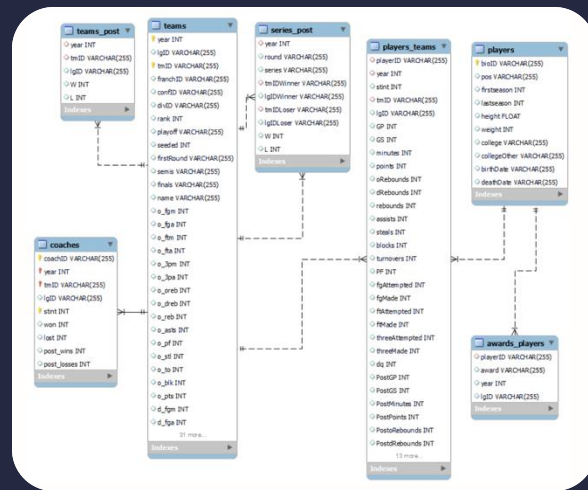
04

05 CONCLUSIONS, LIMITATIONS, FUTURE WORK

06 ANNEXES

01

DOMAIN DESCRIPTION



Domain Description

Project Overview

This Project consists of developing a data mining case study. We have to predict which teams from the WNBA will qualify for the next playoffs, taking into consideration data from the past 10 years. Each year, 4 teams from each conference (Eastern and Western) qualify for the playoffs based on their performance during the regular season. This is a binary problem, as we use 1 and 0 to indicate if a team will qualify for the playoffs (1 - qualifies, 0 - doesn't qualify).

Provided Data

The analysis uses datasets on players, teams, coaches, and playoffs performance, including awards history and relationships between players and teams

- **awards_players** - awards and prizes received by players across 10 seasons
- **coaches** - coaches who've managed the teams during the time period
- **players** - details of all players
- **players_teams** - performance of each player for each team they played
- **series_post** - series results
- **teams** - performance of the teams for each season
- **teams_post** - results of each team at the post-season



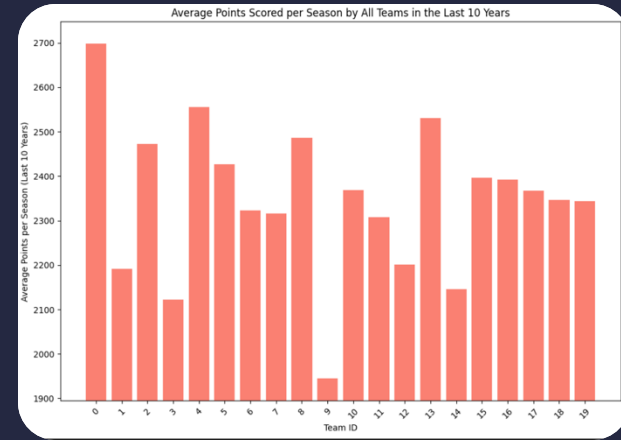
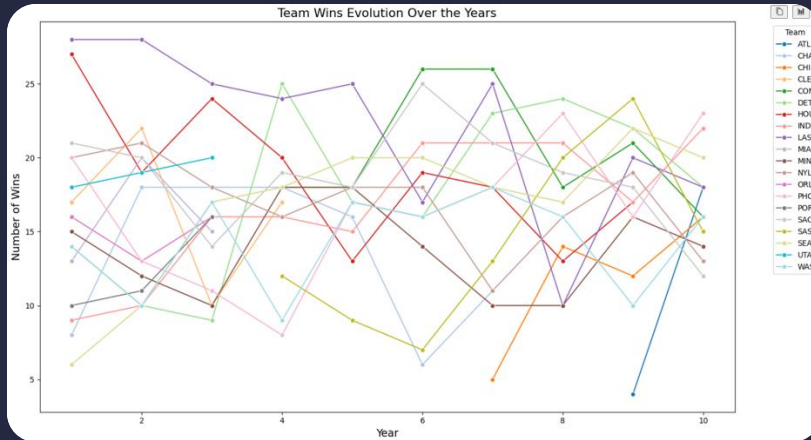
02

EXPLORATORY DATA ANALYSIS

Exploratory Data Analysis (EDA) is the process of analyzing data to summarize its main characteristics, often through visualizations, in order to identify patterns, or anomalies.

Exploratory Data Analysis - 1

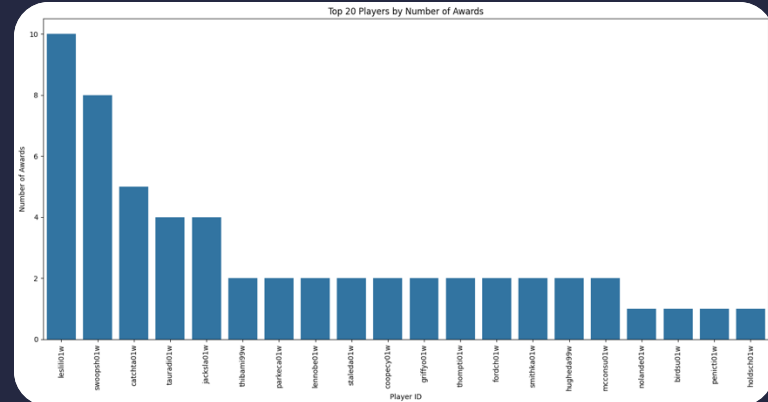
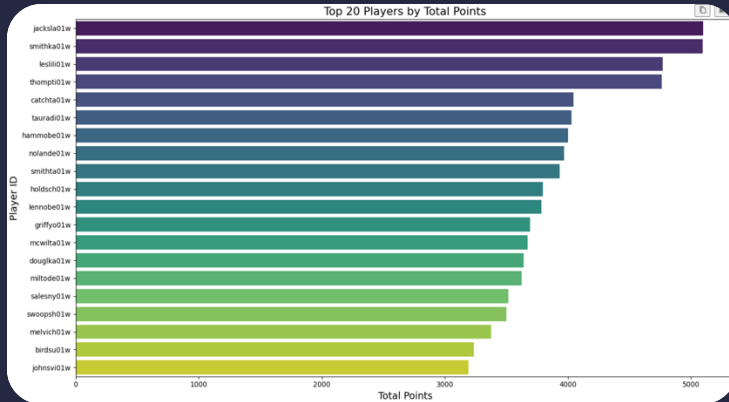
The first thing we decided to do was to create graphs to better understand the data we were given, to find patterns and useful information to be carried out to our analysis later. We started by looking at the team wins evolution and average points scored per season by all teams in the last 10 years.



Some teams stand out
Relation between wins and points scored

Exploratory Data Analysis - 2

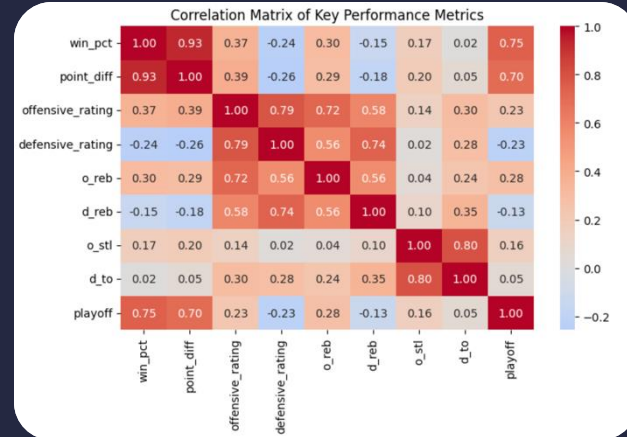
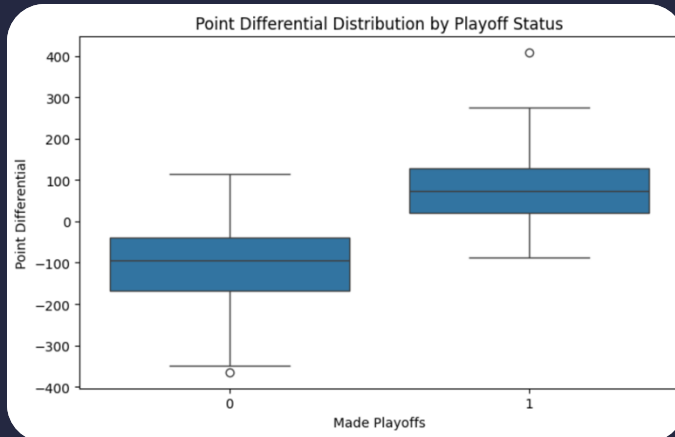
The second step involved analyzing player data, focusing on the players with the highest points scored and the most awards received, to understand their impact on team performance.



Some players stand out
Correlation between points scored and number of awards

Exploratory Data Analysis - 3

The third step involved analyzing correlations in the data, that could potentially show patterns and how metrics are related between them. For that, we used a Box Plot and a Correlation Matrix.



Strong correlation (0.93) between win percentage and point differential
Offensive and defensive rebounds (o_reb, d_reb) have relatively weak correlations with playoff



03

PREDICTIVE DATA MINING PROBLEM

A predictive data mining problem is a problem that uses historical data to create models that predict future outcomes or trends.

PREDICTIVE DATA MINING PROBLEM – Problem Definition

Every year, each team tries to achieve the best results possible in the normal season. The best 4 teams from each conference advance to the playoffs.

Objective

Build a predictive model to forecast playoff qualification, whether a team qualifies for the playoffs or not (binary outcome).



Input Data

Player statistics, team performance, coach information, and historical playoff data.

Evaluation Metric

The model will be evaluated using **accuracy** to assess the overall correctness of the predictions.

DATA PREPARATION - Summary

Mapping Categorical Values



Removing Unnecessary Data



Handling Missing Values and Outliers



DATA PREPARATION – Mapping Categorical Values (1)

Mapping Definition: Preprocessing step that converts text data into numbers to improve data analysis efficiency

- We mapped categorical values (teams, arenas, awards) to numerical codes for model training
- Player positions into numerical values
- Playoff outcomes to numerical representations
- Eliminates issues with inconsistent string entries
- Makes data cleaning easier as we have a standardized reference
- Most ML algorithms require numerical inputs

```
players['pos'] = players['pos'].replace({'G-C': 'C-G', 'F-G': 'G-F'})

positions = players['pos'].unique()
map_positions = {pos: i for i, pos in enumerate(positions)}

players['pos'] = players['pos'].map(map_positions)
```

```
teams['firstRound'] = replace_values(teams, 'firstRound', ['L', 'W'], [0, 1])
teams['semis'] = replace_values(teams, 'semis', ['L', 'W'], [0, 1])
teams['finals'] = replace_values(teams, 'finals', ['L', 'W'], [0, 1])
teams['playoff'] = replace_values(teams, 'playoff', ['N', 'Y'], [0, 1])
teams['confID'] = replace_values(teams, 'confID', ['EA', 'WE'], [0, 1])
```

```
map_teams = {
    'ATL': 0, 'CHA': 1, 'CHI': 2, 'CLE': 3, 'CON': 4,
    'DET': 5, 'HOU': 6, 'IND': 7, 'LAS': 8, 'MIA': 9,
    'MIN': 10, 'NYL': 11, 'ORL': 12, 'PHO': 13, 'POR': 14,
    'SAC': 15, 'SAS': 16, 'SEA': 17, 'UTA': 18, 'WAS': 19
}
```

```
map_awards = {
    'All-Star Game Most Valuable Player': 0,
    'Coach of the Year': 1,
    'Defensive Player of the Year': 2,
    'Kim Perrot Sportsmanship Award': 3,
    'Kim Perrot Sportsmanship': 3,
    'Most Improved Player': 4,
    'Most Valuable Player': 5,
    'Rookie of the Year': 6,
    'Sixth Woman of the Year': 7,
    'WNBA Finals Most Valuable Player': 8,
    'WNBA All-Decade Team': 9,
    'WNBA All Decade Team Honorable Mention': 10
}
```

DATA PREPARATION – Removing Unnecessary Data (2)

Removing Data:

We cleaned the players dataset by removing rows and dropping irrelevant columns.

- Death Dates
- firstseason, lastseason (since are all 0)
- LgID(always the same) , lgIDWinner, lgIDLoser
- Players with no Birth Date
- CollegeOther

```
players_cleaned = players[players['deathDate'] == '0000-00-00'].drop(columns=['deathDate','firstseason','lastseason'])
```

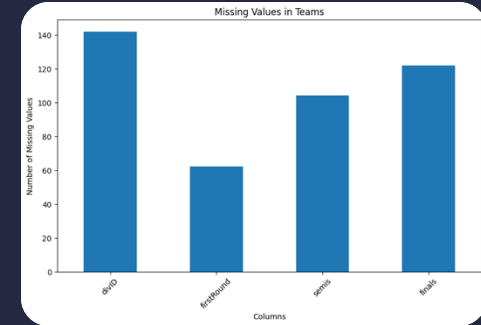
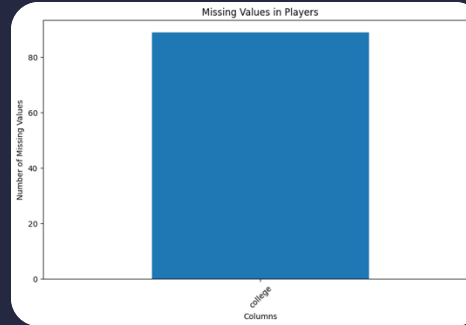
```
players_cleaned = players_cleaned.drop(columns=['collegeOther'])
```

```
players_no_birthdate = players_cleaned[players_cleaned['birthDate'].isnull()]
```

```
coaches_cleaned = coaches.drop(columns=['lgID'])
```

DATA PREPARATION – Handling Missing Values (2)

Handling missing values



- For some of the missing values, we simply deleted the column and treated them as unnecessary data. e.g: CollegeOther, BirthDate, DivId
- For the others, we only used the average of the ones that already existed and added the average to the missing values, which in some cases also act as outliers e.g: Missing Weights
- For others that we considered important, we simply decided to assign them a different value than 0, because we might use them in the future e.g: firstRound, semis, finals

```
ideal_bmi = 22
players_cleaned['height_m'] = players_cleaned['height'] * 0.0254
players_cleaned.loc[players_cleaned['weight'] == 0, 'weight'] = (ideal_bmi * (players_cleaned['height_m']
players_cleaned['weight'] = players_cleaned['weight'].astype(int)
players_cleaned.drop(columns=['height_m'], inplace=True)
```

```
teams_cleaned['firstRound'] = teams_cleaned['firstRound'].fillna(2)
teams_cleaned['semis'] = teams_cleaned['semis'].fillna(2)
teams_cleaned['finals'] = teams_cleaned['finals'].fillna(2)
```

DATA PREPARATION – Outliers (3)

Handling Outliers

For outliers, we used the Z-Score. To handle the weight, we replaced the values with the average value based on the ideal BMI, using a BMI of **22** as the reference.

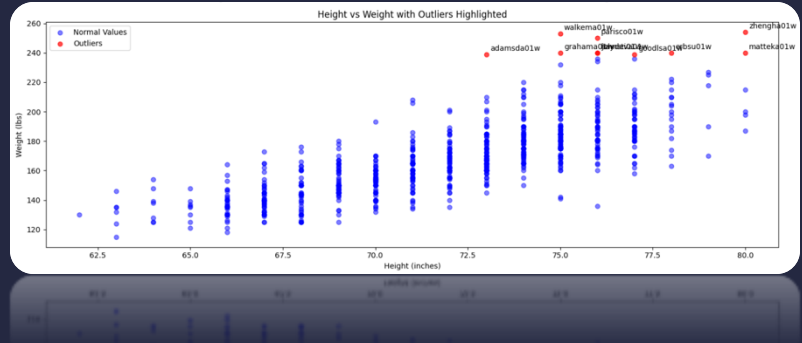
This type of chart can help provide a better perspective on outliers that cannot be seen in an initial analysis of the data.

Other than the weight, we didn't find any relevant outlier, as in the WNBA, some values tend to be slightly different from what we are normally used to.

```
ideal_bmi = 22

players_cleaned['height_m'] = players_cleaned['height'] * 0.0254
players_cleaned.loc[players_cleaned['weight'] == 0, 'weight'] = (ideal_bmi * (players_cleaned['height_m']
players_cleaned['weight'] = players_cleaned['weight'].astype(int)

players_cleaned.drop(columns=['height_m'], inplace=True)
```



EXPERIMENTAL SETUP

Feature Engineering:

- Players Overall Calculation
- Rookies Average Calculation
- Coaches Overall Calculation

5 models implemented:

- SVM
- Decision Tree
 - RFC
- Logistic Regression
 - KNN

1 year of data used for testing and 9 years of data used for training, using these features:

- Games Won and Lost
- Offensive and Defensive points
- Offensive and Defensive rebounds
- 3 year win rate, playoff appearance rate and points differential average
- Team player overall (stamina, offense, defense)
 - Coach Overall

Feature Engineering (Player Overall Calculation (1/2))

To calculate the Players Overall Calculation, we calculated three sub scores:

- **Stamina Overall Calculation** ((games played *20)+(minutes played *80))
- **Defense Overall Calculation** (defensive rebounds (45%), steals (20%), blocks (20%), turnovers (-5%), personal fouls (-5%), disqualifications (-5%))
- **Offense Overall Calculation** (points scored (40%), assists (25%), field goals made (10%), free throws made (10%), three-pointers made (10%), offensive rebounds (5%))

```
def calculate_overall_offense(df):
    total_games = df['GP'].sum() + df['PostGP'].sum()

    mean_points = (df['points'].sum() + df['PostPoints'].sum()) / total_games if total_games > 0 else 1
    mean_assists = (df['assists'].sum() + df['PostAssists'].sum()) / total_games if total_games > 0 else 1
    mean_fgMade = (df['fgMade'].sum() + df['PostfgMade'].sum()) / total_games if total_games > 0 else 1
    mean_ftMade = (df['ftMade'].sum() + df['PostftMade'].sum()) / total_games if total_games > 0 else 1
    mean_threeMade = (df['threeMade'].sum() + df['PostthreeMade'].sum()) / total_games if total_games > 0 else 1
    mean_orebounds = (df['orebounds'].sum() + df['Postorebounds'].sum()) / total_games if total_games > 0 else 1

    player_points = (df['points'] + df['PostPoints']) / (df['GP'] + df['PostGP'])
    player_assists = (df['assists'] + df['PostAssists']) / (df['GP'] + df['PostGP'])
    player_fgMade = (df['fgMade'] + df['PostfgMade']) / (df['GP'] + df['PostGP'])
    player_ftMade = (df['ftMade'] + df['PostftMade']) / (df['GP'] + df['PostGP'])
    player_threeMade = (df['threeMade'] + df['PostthreeMade']) / (df['GP'] + df['PostGP'])
    player_orebounds = (df['orebounds'] + df['Postorebounds']) / (df['GP'] + df['PostGP'])

    overall_offense = {
        (0.4 * (player_points / mean_points)) +
        (0.25 * (player_assists / mean_assists)) +
        (0.1 * (player_fgMade / mean_fgMade)) +
        (0.1 * (player_ftMade / mean_ftMade)) +
        (0.1 * (player_threeMade / mean_threeMade)) +
        (0.05 * (player_orebounds / mean_orebounds))
    }

    overall_offense = np.clip(overall_offense * 10, 1, 20)
    return overall_offense.round(1)

players_teams_cleaned['overallOFFENSE'] = calculate_overall_offense(players_teams_cleaned)
```

```
def calculate_stamina(df):
    total_games = df['GP'].sum() + df['PostGP'].sum()
    total_minutes = df['minutes'].sum() + df['PostMinutes'].sum()

    mean_games = total_games / len(df)
    mean_minutes = total_minutes / len(df)

    overall_stamina = (
        (0.2 * (df['GP'] + df['PostGP']) / mean_games) +
        (0.8 * (df['minutes'] + df['PostMinutes']) / mean_minutes)
    )

    overall_stamina = np.clip(overall_stamina * 10, 1, 20)
    return overall_stamina.round(1)

players_teams_cleaned['overallSTAMINA'] = calculate_stamina(players_teams_cleaned)
```

```
def calculate_overall_defense(df):
    total_games = df['GP'].sum() + df['PostGP'].sum()

    mean_drebounds = (df['drebounds'].sum() + df['Postdrebounds'].sum()) / total_games if total_games > 0 else 1
    mean_steals = (df['steals'].sum() + df['Poststeals'].sum()) / total_games if total_games > 0 else 1
    mean_blocks = (df['blocks'].sum() + df['Postblocks'].sum()) / total_games if total_games > 0 else 1
    mean_turnovers = (df['turnovers'].sum() + df['PostTurnovers'].sum()) / total_games if total_games > 0 else 1
    mean_pf = (df['pf'].sum() + df['Postpf'].sum()) / total_games if total_games > 0 else 1
    mean_dq = (df['dq'].sum() + df['PostDQ'].sum()) / total_games if total_games > 0 else 0

    player_drebounds = (df['drebounds'] + df['Postdrebounds']) / (df['GP'] + df['PostGP'])
    player_steals = (df['steals'] + df['Poststeals']) / (df['GP'] + df['PostGP'])
    player_blocks = (df['blocks'] + df['Postblocks']) / (df['GP'] + df['PostGP'])
    player_turnovers = (df['turnovers'] + df['PostTurnovers']) / (df['GP'] + df['PostGP'])
    player_pf = (df['pf'] + df['Postpf']) / (df['GP'] + df['PostGP'])
    player_dq = (df['dq'] + df['PostDQ']) / (df['GP'] + df['PostGP'])

    overall_defense = {
        (0.45 * (player_drebounds / mean_drebounds)) +
        (0.2 * (player_steals / mean_steals)) +
        (0.2 * (player_blocks / mean_blocks)) +
        (0.05 * (player_turnovers / mean_turnovers)) -
        (0.05 * (player_pf / mean_pf)) -
        (0.05 * (player_dq / mean_dq))
    }

    overall_defense = np.clip(overall_defense * 10, 1, 20)
    return overall_defense.round(1)

players_teams_cleaned['overallDEFENSE'] = calculate_overall_defense(players_teams_cleaned)
```

Feature Engineering (Player Overall Calculation (2/2))

To calculate the final Player Combined Overall Calculation, we merged the previous calculated scores as follow:

- **Base Score (92% of total):**

Stamina Score (18%)

Defense score (37%)

Offense Score (37%)

- **Additional factors (8% of total):**

Player Height (4%)

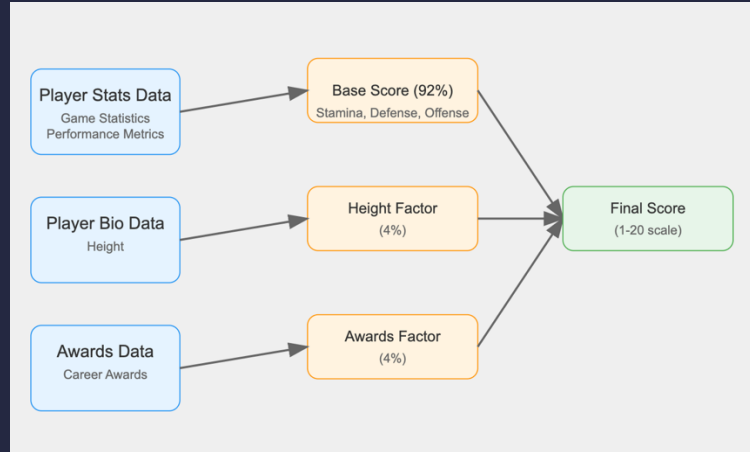
Career Awards (4%)

- **Final Adjustments:**

Scores are capped between 1 and 20

Top 5% of scores are dampened to prevent outliers

Final score is rounded to one decimal place



Thanks to this overall, we managed to understand how the players performed in the last seasons. This overall will also play a vital role in the model prediction, as better players tend to frequently help their teams reach the playoffs.

Feature Engineering (Coaches Overall Calculation)

To calculate the Coaches Overall Calculation, we evaluate:

- Total wins (Regular Season + Playoffs) vs Total Losses
- Compare individual Coach performance against league average
- Score ranges between 1-20, based on relative win percentage
- Formula: $(\text{coach Win \%} / \text{League Average Win\%}) * 10$
- Results are rounded to one decimal place

To calculate all-time Coach Ratings:

- Calculates career-long performance metrics for each coach
- Groups all seasons by unique coach ID
- Takes the average of all their seasonal overall ratings

```
def calculate_overall_coach(df):  
    total_wins = df['won'].sum() + df['post_wins'].sum()  
    total_losses = df['lost'].sum() + df['post_losses'].sum()  
  
    total_games = total_wins + total_losses  
  
    win_percentage = total_wins / total_games if total_games > 0 else 0  
  
    total_wins_per_coach = df['won'] + df['post_wins']  
    total_losses_per_coach = df['lost'] + df['post_losses']  
  
    coach_win_percentage = total_wins_per_coach / (total_wins_per_coach + total_losses_per_coach)  
  
    relative_performance = coach_win_percentage / win_percentage  
  
    overall = np.clip(relative_performance * 10, 1, 20)  
    return overall.round(1)  
  
coaches_cleaned['OVERALL'] = calculate_overall_coach(coaches_cleaned)  
coaches_cleaned.to_csv('../data/basketballPlayoffs_cleaned/coaches_cleaned.csv', index=False)
```

Thanks to this overall, we managed to understand how the coaches performed in the last seasons. This overall is not the most important factor in a team success, but we still think it makes sense to include it, because some teams really rely on good coaches to achieve good results and reach the playoffs.

Feature Engineering (Rookies Average Calculation)

To calculate the Rookies Average Calculation, we evaluate:

- Earliest year for each player ID
- Mark players as rookies (1) in their first season
- All others are marked as non rookies (0)
- Mean of overall scores from these seasons

Thanks to this overall, we can predict a supposed average performance, for a player entering the league for the first time, and we won't have 'ghost players' (players without any kind of information).

```
rookie_year = players_teams_cleaned.groupby('playerID')['year'].min()

players_teams_cleaned['is_rookie'] = players_teams_cleaned.apply(
    lambda row: 1 if row['year'] == rookie_year[row['playerID']] else 0,
    axis=1
)

players_teams_cleaned.to_csv('../data/basketballPlayoffs_cleaned/players_teams_cleaned.csv', index=False)

rookie_players = players_teams_cleaned[players_teams_cleaned['is_rookie'] == 1]

rookie_overall_avg = rookie_players['OVERALL'].mean().round(1)

rookie_overall_avg_df = pd.DataFrame({'rookie_overall_avg': [rookie_overall_avg]})

rookie_overall_avg_df.to_csv('../data/basketballPlayoffs_cleaned/rookie_overall_avg.csv', index=False)
```

'ATL': 0, 'CHA': 1, 'CHI': 2, 'CLE': 3, 'CON': 4,

'DET': 5, 'HOU': 6, 'IND': 7, 'LAS': 8, 'MIA': 9,

RESULTS SVM MODEL

Training Metrics (Years 1-9):

- Training Accuracy: 73.39% (reasonable learning without overfitting)
- Training F1 Score: 0.79 (balanced performance)
- Training Precision: 0.74
- Training Recall: 0.84 (model favors positive predictions)
- Sum of probabilities: 8.00
- Test Accuracy (Year 10): 62.50%
- Prediction Error: 5.86

Model shows decent training performance but limited generalization to new data, suggesting potential improvements needed in feature selection or parameter tuning.

5/8

Number of teams making the playoffs predicted correctly

RESULTS

Team Predictions (1 = Playoff, 0 = No Playoff):

Team 0: Probability = 0.56, Predicted = 0, Actual = 1
Team 2: Probability = 0.55, Predicted = 0, Actual = 0
Team 4: Probability = 0.63, Predicted = 1, Actual = 0
Team 5: Probability = 0.72, Predicted = 1, Actual = 1
Team 7: Probability = 0.76, Predicted = 1, Actual = 1
Team 8: Probability = 0.62, Predicted = 1, Actual = 1
Team 10: Probability = 0.54, Predicted = 0, Actual = 0
Team 11: Probability = 0.63, Predicted = 1, Actual = 0
Team 13: Probability = 0.55, Predicted = 0, Actual = 1
Team 15: Probability = 0.59, Predicted = 1, Actual = 0
Team 16: Probability = 0.60, Predicted = 1, Actual = 1
Team 17: Probability = 0.67, Predicted = 1, Actual = 1
Team 19: Probability = 0.60, Predicted = 0, Actual = 1

'MIN': 10, 'NYL': 11, 'ORL': 12, 'PHO': 13, 'POR': 14,

'SAC': 15, 'SAS': 16, 'SEA': 17, 'UTA': 18, 'WAS': 19

'ATL': 0, 'CHA': 1, 'CHI': 2, 'CLE': 3, 'CON': 4,

'DET': 5, 'HOU': 6, 'IND': 7, 'LAS': 8, 'MIA': 9,

RESULTS Decision Tree MODEL

Training Metrics (Years 1-9):

- Training Accuracy: 79.82% (strong learning capability)
- Training F1 Score: 0.84 (indicates balanced predictions)
- Training Precision: 0.79
- Training Recall: 0.89 (shows strong ability to identify playoff teams)
- Sum of probabilities: 8.00
- Test Accuracy (Year 10): 87.50%
- Prediction Error: 5.32

Best performing model among all tested, showing excellent generalization to new data and reliable playoff predictions. Strong performance in both training and testing suggests optimal model complexity.

7/8

Number of teams making the playoffs predicted correctly

RESULTS

Team Predictions (1 = Playoff, 0 = No Playoff):

Team 0: Probability = 0.53, Predicted = 1, Actual = 1
Team 2: Probability = 0.53, Predicted = 0, Actual = 0
Team 4: Probability = 0.54, Predicted = 1, Actual = 0
Team 5: Probability = 0.89, Predicted = 1, Actual = 1
Team 7: Probability = 0.89, Predicted = 1, Actual = 1
Team 8: Probability = 0.53, Predicted = 1, Actual = 1
Team 10: Probability = 0.53, Predicted = 0, Actual = 0
Team 11: Probability = 0.53, Predicted = 0, Actual = 0
Team 13: Probability = 0.54, Predicted = 1, Actual = 1
Team 15: Probability = 0.53, Predicted = 0, Actual = 0
Team 16: Probability = 0.54, Predicted = 1, Actual = 1
Team 17: Probability = 0.89, Predicted = 1, Actual = 1
Team 19: Probability = 0.53, Predicted = 0, Actual = 1

'MIN': 10, 'NYL': 11, 'ORL': 12, 'PHO': 13, 'POR': 14,

'SAC': 15, 'SAS': 16, 'SEA': 17, 'UTA': 18, 'WAS': 19

'ATL': 0, 'CHA': 1, 'CHI': 2, 'CLE': 3, 'CON': 4,

'DET': 5, 'HOU': 6, 'IND': 7, 'LAS': 8, 'MIA': 9,

RESULTS RFC MODEL

Training Metrics (Years 1-9):

- Training Accuracy: 94.50% (strong pattern recognition)
- Training F1 Score: 0.95 (very balanced predictions)
- Training Precision: 0.95 (Perfect balance)
- Training Recall: 0.95
- Sum of probabilities: 8.00
- Test Accuracy (Year 10): 75.00%
- Prediction Error: 5.31

Strong training performance but lower test accuracy suggests some overfitting. Not optimal generalization to new data compared to simpler models.

6/8

Number of teams making the playoffs predicted correctly

RESULTS

Team Predictions (1 = Playoff, 0 = No Playoff):

Team 0: Probability = 0.45, Predicted = 0, Actual = 1
Team 2: Probability = 0.41, Predicted = 0, Actual = 0
Team 4: Probability = 0.76, Predicted = 1, Actual = 0
Team 5: Probability = 0.83, Predicted = 1, Actual = 1
Team 7: Probability = 0.87, Predicted = 1, Actual = 1
Team 8: Probability = 0.71, Predicted = 1, Actual = 1
Team 10: Probability = 0.39, Predicted = 0, Actual = 0
Team 11: Probability = 0.60, Predicted = 1, Actual = 0
Team 13: Probability = 0.60, Predicted = 1, Actual = 1
Team 15: Probability = 0.49, Predicted = 0, Actual = 0
Team 16: Probability = 0.72, Predicted = 1, Actual = 1
Team 17: Probability = 0.78, Predicted = 1, Actual = 1
Team 19: Probability = 0.38, Predicted = 0, Actual = 1

'MIN': 10, 'NYL': 11, 'ORL': 12, 'PHO': 13, 'POR': 14,

'SAC': 15, 'SAS': 16, 'SEA': 17, 'UTA': 18, 'WAS': 19

'ATL': 0, 'CHA': 1, 'CHI': 2, 'CLE': 3, 'CON': 4,

'DET': 5, 'HOU': 6, 'IND': 7, 'LAS': 8, 'MIA': 9,

RESULTS Logistic Regression MODEL

Training Metrics (Years 1-9):

- Training Accuracy: 71.56% (shows balanced learning)
- Training F1 Score: 0.76
- Training Precision: 0.75
- Training Recall: 0.76
- Sum of probabilities: 8.00
- Test Accuracy (Year 10): 75.00%
- Prediction Error: 4.96 (Best prediction)

LC provides a solid performance with the lowest prediction error of all models we implemented. Also has good balance between training and test performance, which indicates proper generalization.

6/8

Number of teams making the playoffs predicted correctly

RESULTS

Team Predictions (1 = Playoff, 0 = No Playoff):

Team 0: Probability = 0.59, Predicted = 0, Actual = 1
Team 2: Probability = 0.39, Predicted = 0, Actual = 0
Team 4: Probability = 0.72, Predicted = 1, Actual = 0
Team 5: Probability = 0.89, Predicted = 1, Actual = 1
Team 7: Probability = 0.76, Predicted = 1, Actual = 1
Team 8: Probability = 0.81, Predicted = 1, Actual = 1
Team 10: Probability = 0.20, Predicted = 0, Actual = 0
Team 11: Probability = 0.59, Predicted = 0, Actual = 0
Team 13: Probability = 0.77, Predicted = 1, Actual = 1
Team 15: Probability = 0.58, Predicted = 1, Actual = 0
Team 16: Probability = 0.51, Predicted = 1, Actual = 1
Team 17: Probability = 0.43, Predicted = 0, Actual = 1
Team 19: Probability = 0.76, Predicted = 1, Actual = 1

'MIN': 10, 'NYL': 11, 'ORL': 12, 'PHO': 13, 'POR': 14,

'SAC': 15, 'SAS': 16, 'SEA': 17, 'UTA': 18, 'WAS': 19

'ATL': 0, 'CHA': 1, 'CHI': 2, 'CLE': 3, 'CON': 4,

'DET': 5, 'HOU': 6, 'IND': 7, 'LAS': 8, 'MIA': 9,

RESULTS KNN MODEL

Training Metrics (Years 1-9):

- Training Accuracy: 74.31% (consistent learning)
- Training F1 Score: 0.78
- Training Precision: 0.78
- Training Recall: 0.78
- Sum of probabilities: 8.00
- Test Accuracy (Year 10): 87.50%
- Prediction Error: 4.98

One of the top performing models with excellent generalization and consistent performance.

7/8

Number of teams making the playoffs predicted correctly

RESULTS

Team Predictions (1 = Playoff, 0 = No Playoff):

Team 0: Probability = 0.44, Predicted = 1, Actual = 1
Team 2: Probability = 0.44, Predicted = 0, Actual = 0
Team 4: Probability = 0.44, Predicted = 0, Actual = 0
Team 5: Probability = 0.87, Predicted = 1, Actual = 1
Team 7: Probability = 1.02, Predicted = 1, Actual = 1
Team 8: Probability = 0.73, Predicted = 1, Actual = 1
Team 10: Probability = 0.44, Predicted = 0, Actual = 0
Team 11: Probability = 0.73, Predicted = 1, Actual = 0
Team 13: Probability = 0.73, Predicted = 1, Actual = 1
Team 15: Probability = 0.44, Predicted = 0, Actual = 0
Team 16: Probability = 0.87, Predicted = 1, Actual = 1
Team 17: Probability = 0.58, Predicted = 1, Actual = 1
Team 19: Probability = 0.29, Predicted = 0, Actual = 1

'MIN': 10, 'NYL': 11, 'ORL': 12, 'PHO': 13, 'POR': 14,

'SAC': 15, 'SAS': 16, 'SEA': 17, 'UTA': 18, 'WAS': 19

'ATL': 0, 'CHA': 1, 'CHI': 2, 'CLE': 3, 'CON': 4,

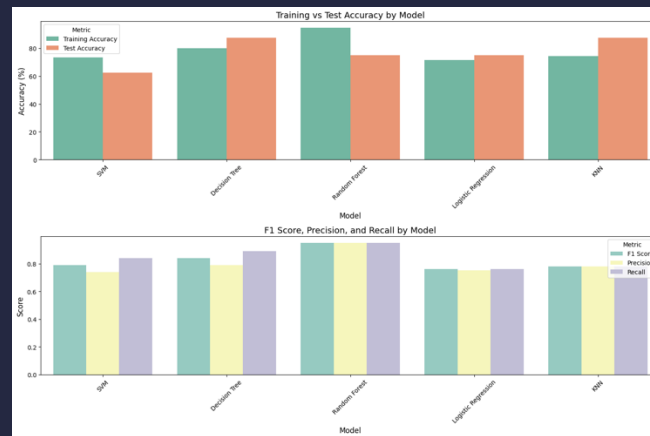
'DET': 5, 'HOU': 6, 'IND': 7, 'LAS': 8, 'MIA': 9,

MODEL RESULTS COMPARISON

Best Model: Decision Tree

- Most balanced performance across metrics
- Highest test accuracy (87.50%, tied with KNN)
- Best playoff prediction accuracy (7/8 correct)
- Strong F1 score (0.84) indicating good precision and recall balance
- Consistent performance between training and testing
- Excellent generalization to new data (Year 10)
- Best balance of precision and recall
- Most reliable playoff predictions
- Avoids overfitting (unlike RFC's 94.50%)

| Model | Test Accuracy | Prediction Error | Correct Predictions |
|---------------------|---------------|------------------|---------------------|
| KNN | 87.50% | 4.98 | 7/8 |
| Decision Tree | 87.50% | 5.32 | 7/8 |
| Random Forest | 75.00% | 5.31 | 6/8 |
| SVM | 62.50% | 5.86 | 5/8 |
| Logistic Regression | 75.00% | 4.96 | 6/8 |

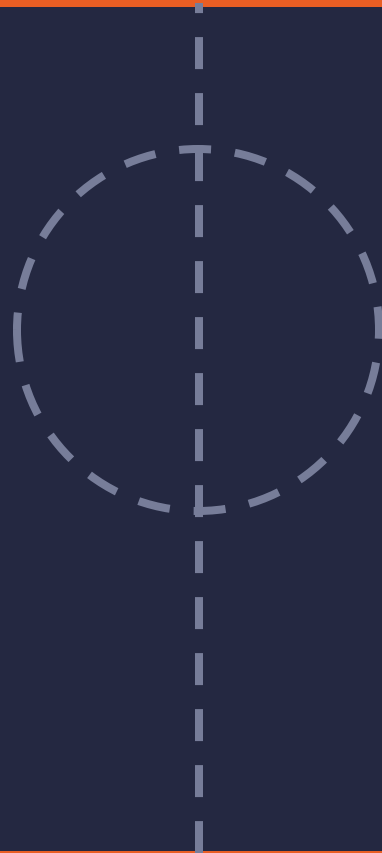


'MIN': 10, 'NYL': 11, 'ORL': 12, 'PHO': 13, 'POR': 14,

'SAC': 15, 'SAS': 16, 'SEA': 17, 'UTA': 18, 'WAS': 19

04

KAGGLE RESULTS



COMPETITION SETUP

Feature Engineering:

- Players Overall Calculation
- Rookies Average Calculation
- Coaches Overall Calculation

5 models implemented:

- SVM
- Decision Tree
 - RFC
- Logistic Regression
 - KNN

1 year of data used for testing and 10 years of data used for training, using these features:

- Games Won and Lost
- Offensive and Defensive points
- Offensive and Defensive rebounds
- 3 year win rate, playoff appearance rate and points differential average
- Team player overall (stamina, offense, defense)
 - Coach Overall

COMPETITION SUBMISSION 1

For the first day of the submission, we submitted a file showing our original probabilities outputs, without converting them to binary (0 and 1). These results directly represent the outputs of our **Random Forest** model, which we selected as it was our best-performing model at the time.

Values range from 0 to 1 and higher values indicate greater confidence in playoff qualification. Probabilities sum to 8.

We got an error of 5.790.

| tmID | Playoff |
|------|---------|
| ATL | 0.77 |
| CHI | 0.76 |
| CON | 0.57 |
| IND | 0.57 |
| LAS | 0.71 |
| MIN | 0.82 |
| NYL | 0.64 |
| PHO | 0.56 |
| SAS | 0.60 |
| SEA | 0.58 |
| TUL | 0.72 |
| WAS | 0.71 |

COMPETITION SUBMISSION 2

For the second day of the submission, we decided to convert the probabilities to discrete values (0, 0.5 and 1).

- 1.0 (strong playoff candidates)
- 0.0 (unlikely playoff candidates)
- 0.5 (teams with similar probabilities)

These results represent the **SVM model**.

We got an error of 4.000, which is better than the previous day.

We used the values generated by the SVM model (0/1 output) with some manual adjustments that we believed made sense and could enhance the predictions. For example, in cases where teams had almost identical probabilities, rather than choosing between 0 or 1, we assigned a value of 0.5 to reduce errors.

| tmID | Playoff |
|------|---------|
| ATL | 0.00 |
| CHI | 0.00 |
| CON | 1.00 |
| IND | 1.00 |
| LAS | 1.00 |
| MIN | 1.00 |
| NYL | 1.00 |
| PHO | 1.00 |
| SAS | 0.50 |
| SEA | 0.50 |
| TUL | 0.00 |
| WAS | 1.00 |

COMPETITION SUBMISSION 3

For the third day of the submission, we decided to stop using 0.5 discrete values, and only use binary values (0 and 1). We changed SAS from 0.5 to 1 and SEA from 0.5 to 0.

- Clearer decision boundary
- Eliminates uncertainty represented by middle values


These results represent the **SVM Model**.

We got an error of 4.000, which is equal to the previous day.

We continued using the results from the SVM model generated on the previous day as the foundation for this submission.

| tmID | Playoff |
|------|---------|
| ATL | 0.00 |
| CHI | 0.00 |
| CON | 1.00 |
| IND | 1.00 |
| LAS | 1.00 |
| MIN | 1.00 |
| NYL | 1.00 |
| PHO | 1.00 |
| SAS | 1.00 |
| SEA | 0.00 |
| TUL | 0.00 |
| WAS | 1.00 |

COMPETITION SUBMISSION 4



For the fourth day of the submission, we changed some values, to try to improve results. We essentially swapped the predictions between SAS and SEA. SAS is now 0 and SEA is now 1.

These results represent the SVM Model.

We got an error of 4.000, which is equal to the previous two days.

We continued using the results from the SVM model, along with a few manual changes that we believed made sense to reduce the error for the competition.

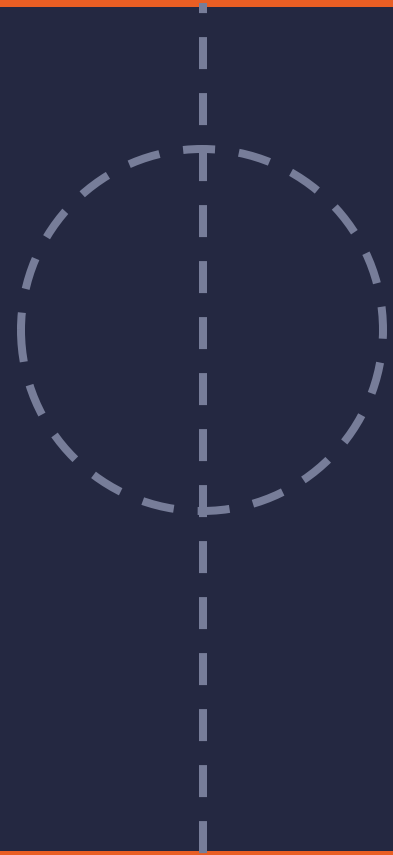


| tmID | Playoff |
|------|---------|
| ATL | 0.00 |
| CHI | 0.00 |
| CON | 1.00 |
| IND | 1.00 |
| LAS | 1.00 |
| MIN | 1.00 |
| NYL | 1.00 |
| PHO | 1.00 |
| SAS | 0.00 |
| SEA | 1.00 |
| TUL | 0.00 |
| WAS | 1.00 |



05

CONCLUSIONS, LIMITATIONS AND FUTURE WORK



CONCLUSIONS, LIMITATIONS AND FUTURE WORK

Conclusions

- Decision Tree and KNN emerged as top performers (87.5% accuracy)
- Most models achieved 6-7 correct playoff predictions out of 8 (only one didn't)
- Most models showed good generalization
- Random Forest showed signs of overfitting (94.5% training vs 75% test)
- Decision Tree achieved best balance between training and test performance
- Exploratory data analysis revealed important patterns in team's performance and qualification trends
- Data Preparation steps were crucial for model success
- Historical performance metrics proved crucial

Limitations

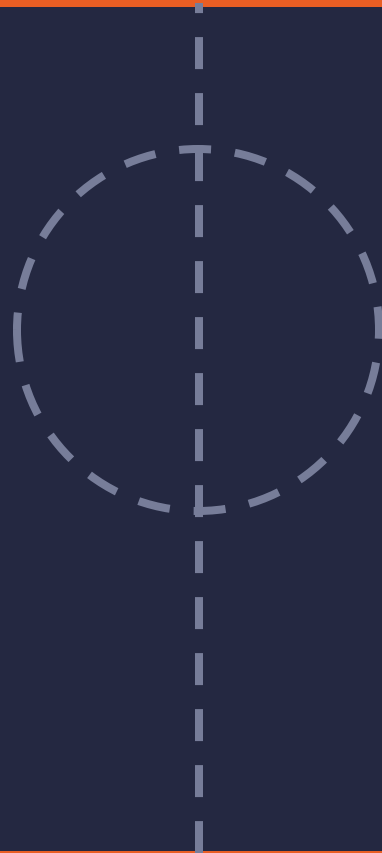
- Limited dataset size (only 9 years for training, 1 for testing (competition phase only has one more year for training))
- Models don't account for trades or injuries
- Historical data maybe doesn't reflect current league dynamics

Future Work

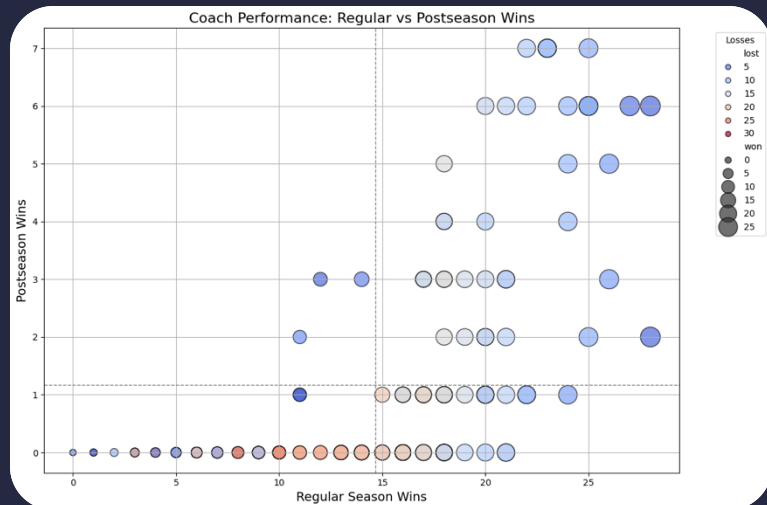
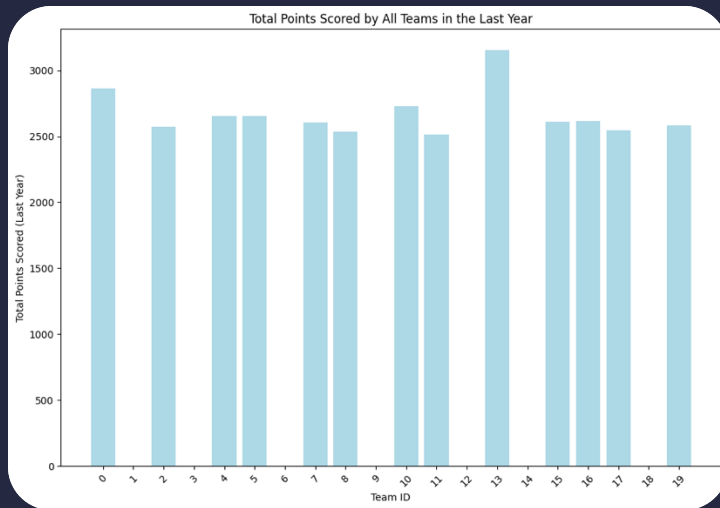
- Improve current models to achieve better results
- Try to get more data to work with

06

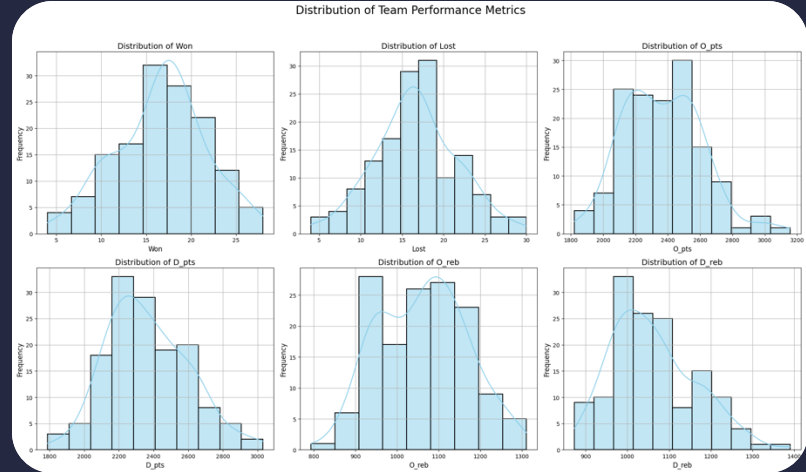
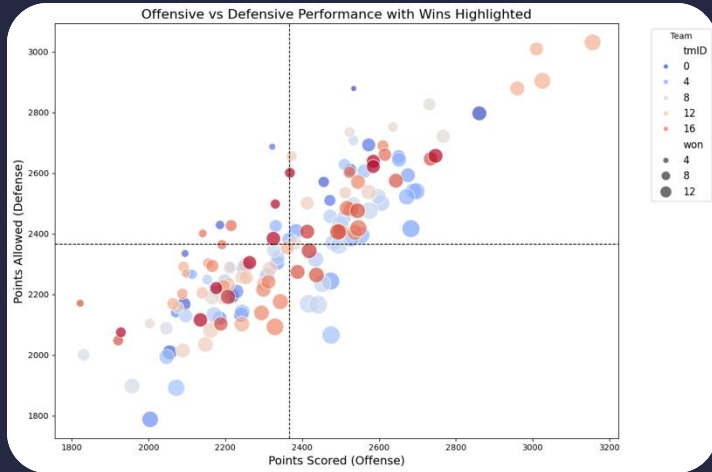
Annexes



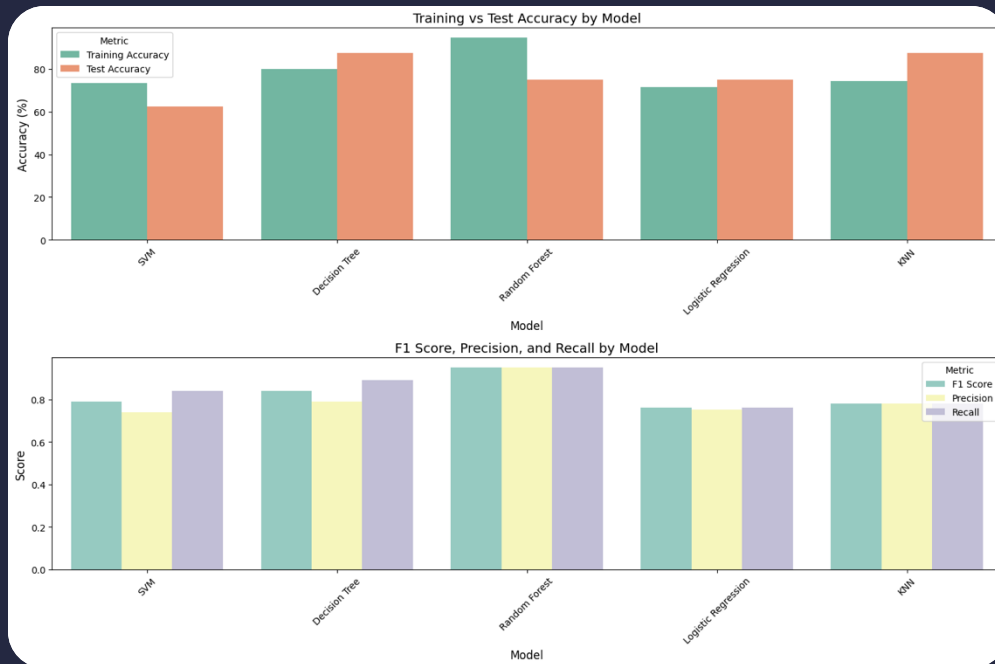
Annexes (1)



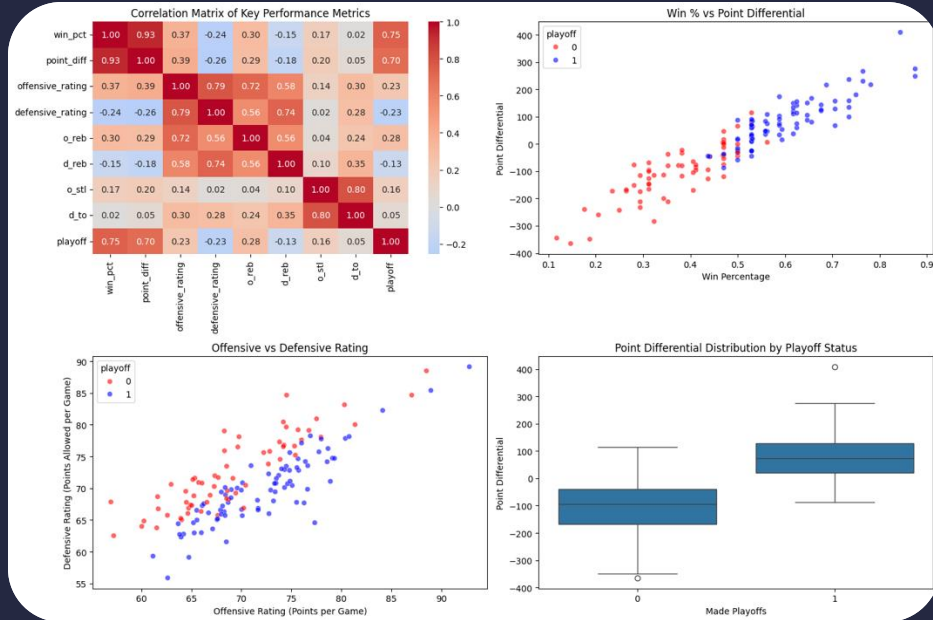
Annexes (2)



Annexes (3)



Annexes (4)



Annexes (5)

