UNIVERSITY OF KWAZULU-NATAL ™

INYUVESI YAKWAZULU-NATALI

COMP721 MACHINE LEARNING

The National Basketball Association (NBA) Prediction Report

Simakahle Goge – 219017247

Mvelo Mlangeni - 219054075

Github link;

https://github.com/gogesimma/Machine-Learning-Project

1. Introduction

The National Basketball Association (NBA) stands as one of the most prominent professional sports leagues globally, captivating millions of fans with high-intensity games and exceptional athletic performances. In recent years, the intersection of sports and data analytics has revolutionized how teams strategize, scout talent, and engage with fans. The advent of advanced statistical analysis and machine learning techniques has provided new avenues to extract insights from vast amounts of historical game data, enhancing decision-making processes both on and off the court.

This project aims to harness machine learning methodologies to analyse NBA data from the 2004-2005 season, focusing on two primary objectives: identifying outstanding players through outlier detection and predicting game outcomes between teams. By examining player statistics, coaching records, and team performances, we seek to uncover patterns and anomalies that distinguish exceptional talent and influence game results.

The dataset utilized in this study encompasses a comprehensive collection of NBA and American Basketball Association (ABA) statistics, including individual player regular-season and playoff stats, career totals, all-star game performances, team regular-season stats, complete draft history, and coaching records. Such a rich dataset provides a robust foundation for applying machine learning algorithms to uncover latent structures and predictive factors within the data.

Identifying outstanding players involves detecting outliers whose performance metrics significantly deviate from their peers. Outlier detection is crucial not only for recognizing top talent but also for understanding the characteristics that contribute to exceptional performance. In parallel, predicting game outcomes requires modelling the complex interactions between teams, accounting for variables such as player statistics, team dynamics, and coaching strategies.

The significance of this study lies in its potential applications within the realms of sports management, betting industries, and fan engagement platforms. Predicting game outcomes in sports has become increasingly valuable for data-driven decision-making in coaching, strategy planning, and sports analysis. This project evaluates a machine learning models, Support Vector Machine (SVM), for predicting game outcomes based on a dataset of team statistics. SVM, a well-known classification algorithm, finds an optimal hyperplane to separate classes, making it useful in binary classification tasks

Accurate predictions of game outcomes can inform strategic decisions, optimize team compositions, and enhance the viewing experience by providing data-driven insights. Moreover, recognizing outstanding players through statistical analysis can aid in talent scouting and career development planning.

In the subsequent sections of this report, we will delve into the specific machine learning techniques employed, including their methodological underpinnings and implementation details. We will present the results of our analyses, discussing their implications and the insights gained. Finally, we will conclude with reflections on the study's contributions and potential avenues for future research.

3. Methods and Techniques

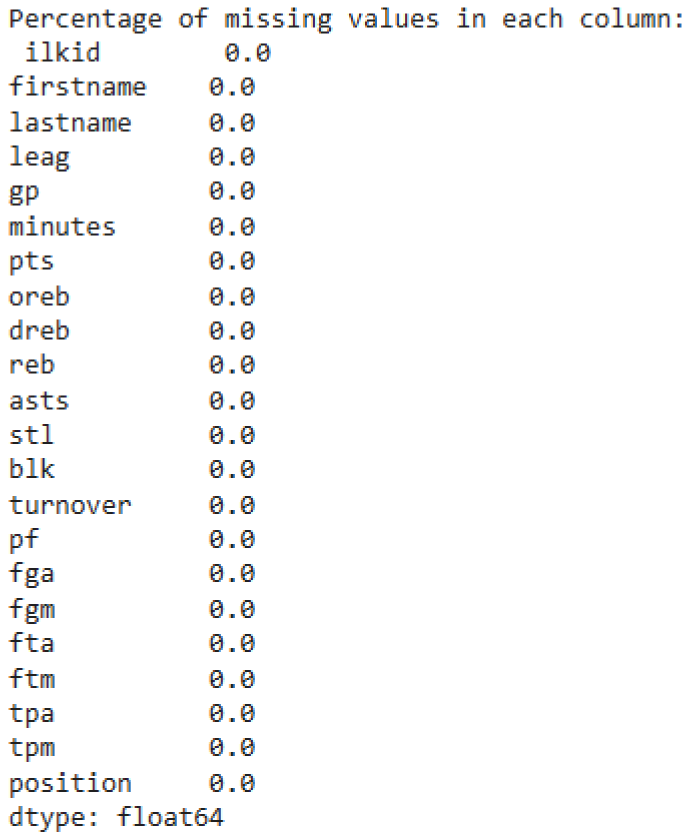
3.1 Outlier detection

3.1.1 Data Loading

The datasets players.txt and player_regular_season_career.txt were loaded and merged on the player ilkid to include player position information.

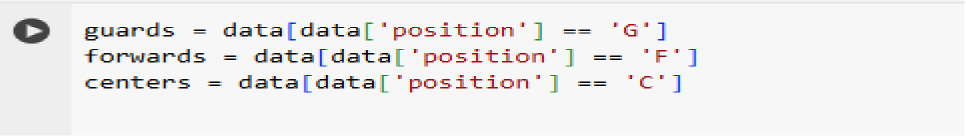
3.1.2 Data Cleaning

Calculated and printed the percentage of missing values in each column. The are no missing values.



```
Percentage of missing values in each column:
 ilkid        0.0
firstname    0.0
lastname     0.0
leag         0.0
gp           0.0
minutes      0.0
pts          0.0
oreb         0.0
dreb         0.0
reb          0.0
asts         0.0
stl          0.0
blk          0.0
turnover     0.0
pf           0.0
fga          0.0
fgm          0.0
fta          0.0
ftm          0.0
tpa          0.0
tpm          0.0
position     0.0
dtype: float64
```

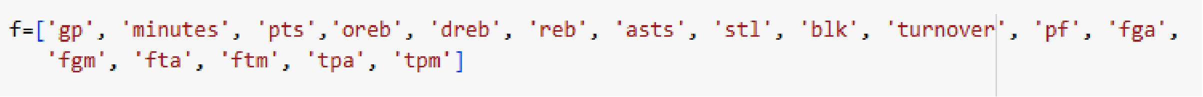
The data was segmented based on player positions, creating three distinct subsets: guards, forwards, and centers. This segmentation allowed for targeted analysis on each position group, accommodating the different roles and performance metrics associated with each.



```
guards = data[data['position'] == 'G']
forwards = data[data['position'] == 'F']
centers = data[data['position'] == 'C']
```

3.1.3 Feature Selection

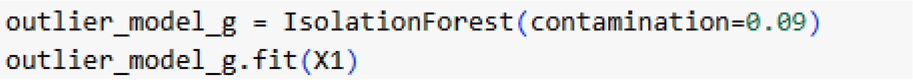
Selected relevant features for analysis, including game performance metrics like gp, minutes, pts, oreb, dreb, etc.



```
f=['gp', 'minutes', 'pts','oreb', 'dreb', 'reb', 'asts', 'stl', 'blk', 'turnover', 'pf', 'fga',
   'fgm', 'fta', 'ftm', 'tpa', 'tpm']
```

3.1.5 Isolation Forest Model

The Isolation Forest model is used to each position group (guards, forwards, and centers) with a contamination rate of 0.09 to detect potential outliers.



```
outlier_model_g = IsolationForest(contamination=0.09)
outlier_model_g.fit(X1)
```

3.1.6 Outlier Prediction

Predicted outliers (outstanding players) for each position group based on the Isolation Forest model results.

3.1.7 Filtering Outstanding Players

Filtered players identified as outliers within each position group.

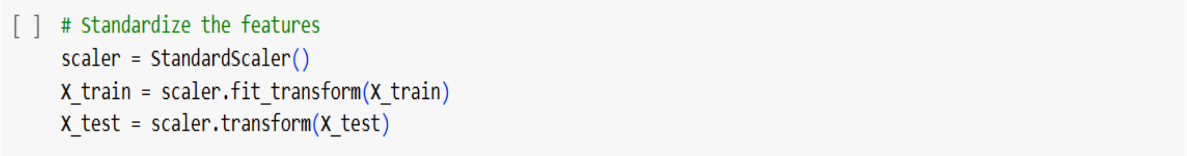
3.1.8 Player Names Extraction

Extracted names of the outstanding players by concatenating first and last names.

3.2 Predicting Game Outcomes

3.2.1 Data Scaling

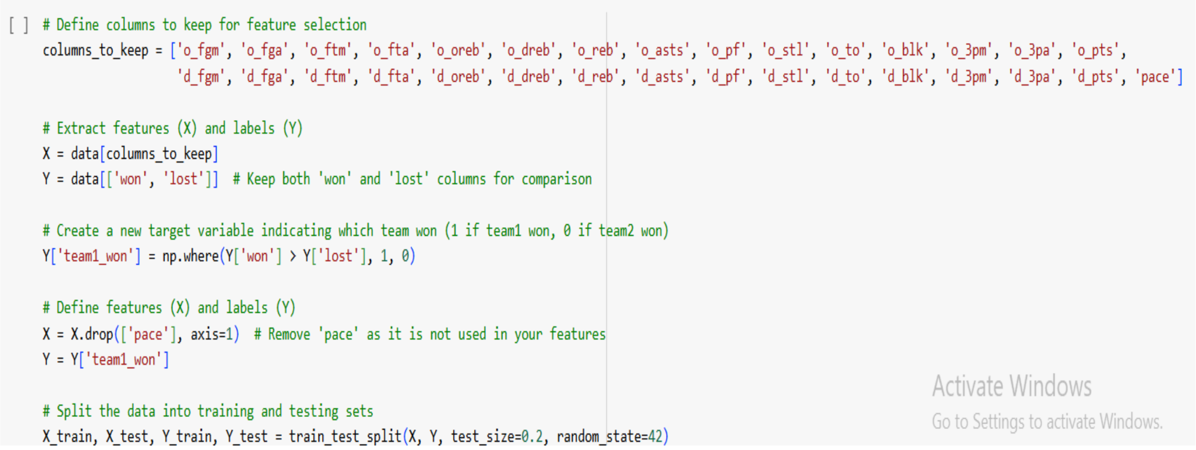
The features were standardized using StandardScaler to bring them to a comparable scale, which is essential for SVM performance.



```
# Standardize the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

3.2.2 Data Splitting

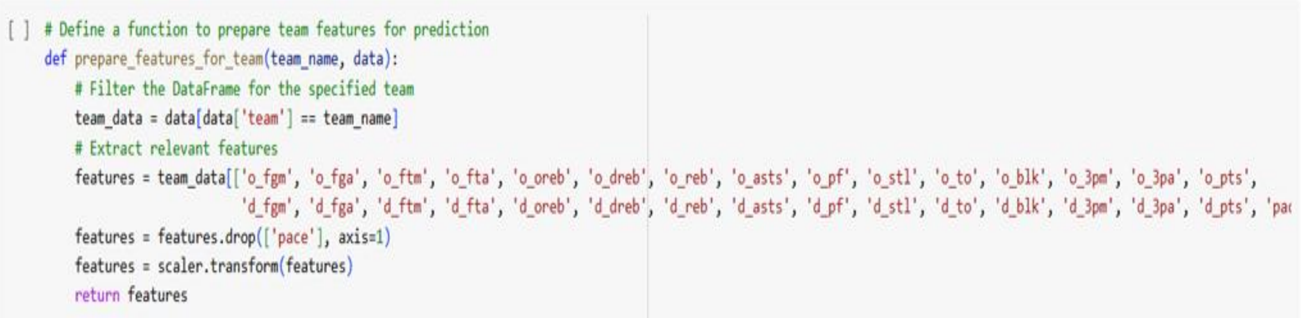
The dataset was divided into training (80%) and testing (20%) sets for model evaluation.



```
[ ] # Define columns to keep for feature selection
    columns_to_keep = ['o_fgm', 'o_fga', 'o_ftm', 'o_fta', 'o_oreb', 'o_dreb', 'o_reb', 'o_asts', 'o_pf', 'o_stl', 'o_to', 'o_blk', 'o_3pm', 'o_3pa', 'o_pts',
                       'd_fgm', 'd_fga', 'd_ftm', 'd_fta', 'd_oreb', 'd_dreb', 'd_reb', 'd_asts', 'd_pf', 'd_stl', 'd_to', 'd_blk', 'd_3pm', 'd_3pa', 'd_pts', 'pace']

    # Extract features (X) and labels (Y)
    X = data[columns_to_keep]
    Y = data[['won', 'lost']]  # Keep both 'won' and 'lost' columns for comparison

    # Create a new target variable indicating which team won (1 if team1 won, 0 if team2 won)
    Y['team1_won'] = np.where(Y['won'] > Y['lost'], 1, 0)

    # Define features (X) and labels (Y)
    X = X.drop(['pace'], axis=1)  # Remove 'pace' as it is not used in your features
    Y = Y['team1_won']

    # Split the data into training and testing sets
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)
```

3.2.3 Feature Selection

Relevant columns for model training are extracted from the dataset. The selected features include offensive and defensive statistics such as field goals made/attempts, free throws made/attempts, rebounds, assists, steals, turnovers, blocks, three-pointers made/attempts, points, and pace.

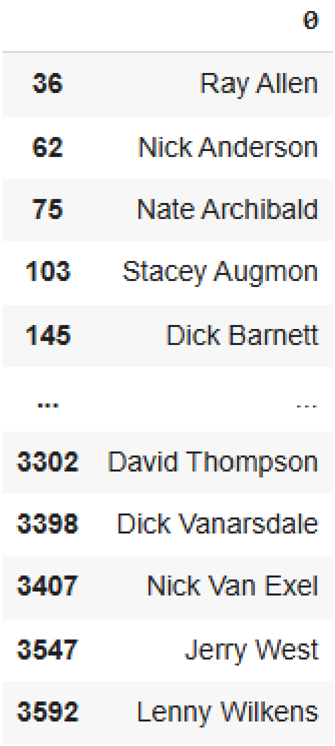


```
[ ] # Define a function to prepare team features for prediction
    def prepare_features_for_team(team_name, data):
        # Filter the DataFrame for the specified team
        team_data = data[data['team'] == team_name]
        # Extract relevant features
        features = team_data[['o_fgm', 'o_fga', 'o_ftm', 'o_fta', 'o_oreb', 'o_dreb', 'o_reb', 'o_asts', 'o_pf', 'o_stl', 'o_to', 'o_blk', 'o_3pm', 'o_3pa', 'o_pts',
                              'd_fgm', 'd_fga', 'd_ftm', 'd_fta', 'd_oreb', 'd_dreb', 'd_reb', 'd_asts', 'd_pf', 'd_stl', 'd_to', 'd_blk', 'd_3pm', 'd_3pa', 'd_pts', 'pa
        features = features.drop(['pace'], axis=1)
        features = scaler.transform(features)
        return features
```

4. Results and Discussion

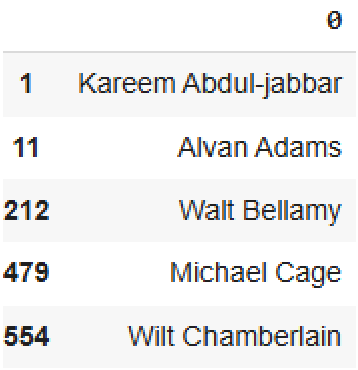
4.1 Outlier detection

The Isolation Forest model identified varying numbers of outliers across the different player positions, highlighting differences in performance distributions. Among the guards, 77 players out of 1,536 were marked as outliers, which suggests that a small subset of guards exhibited performance metrics significantly different from their peers, likely indicating exceptional or atypical performance in key statistics.

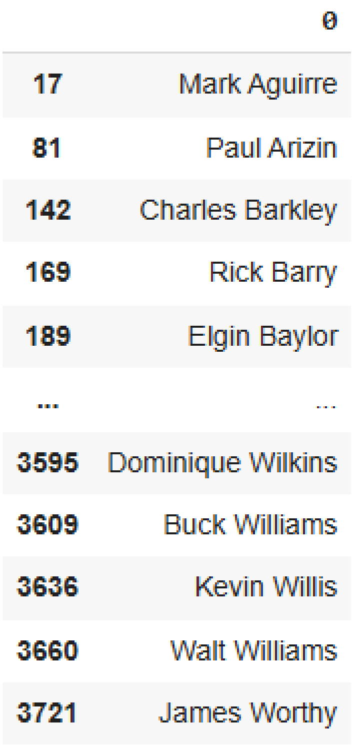


	0
36	Ray Allen
62	Nick Anderson
75	Nate Archibald
103	Stacey Augmon
145	Dick Barnett
...	...
3302	David Thompson
3398	Dick Vanarsdale
3407	Nick Van Exel
3547	Jerry West
3592	Lenny Wilkens

For forwards, 87 out of 1,605 players were identified as outliers. This slightly higher count aligns with the broader range of roles that forwards can play on the court, from scoring to defensive contributions, which may result in more players displaying unique or standout metrics within this group.



	0
1	Kareem Abdul-jabbar
11	Alvan Adams
212	Walt Bellamy
479	Michael Cage
554	Wilt Chamberlain

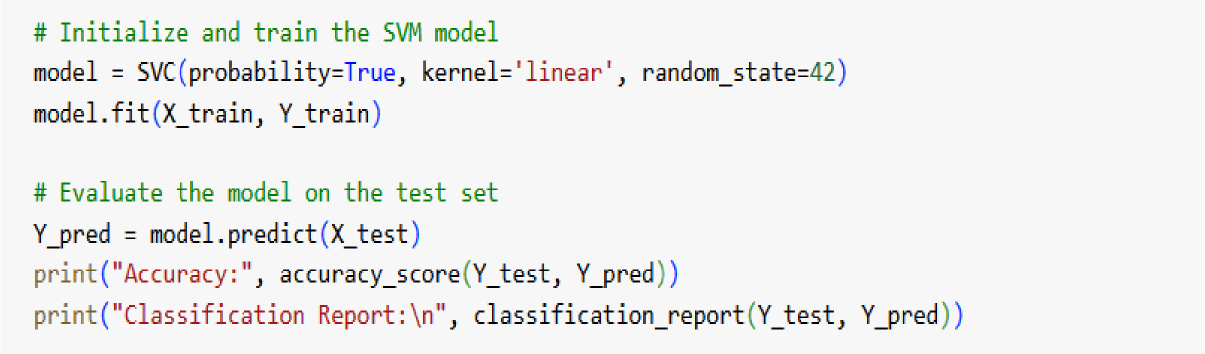


	0
17	Mark Aguirre
81	Paul Arizin
142	Charles Barkley
169	Rick Barry
189	Elgin Baylor
...	...
3595	Dominique Wilkins
3609	Buck Williams
3636	Kevin Willis
3660	Walt Williams
3721	James Worthy

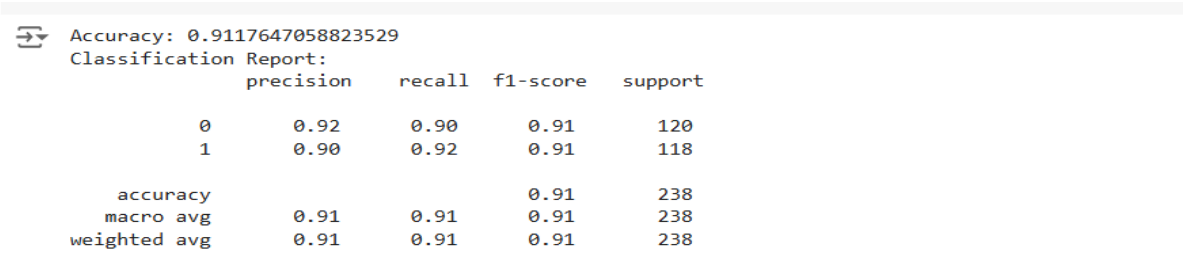
Among centers, 31 players out of 618 were considered outliers. This lower number is expected, given the specialized role of centers, where performance tends to focus on rebounding, blocking, and close-range scoring. The narrower scope of activities for centers often leads to more consistency in their metrics, resulting in fewer statistical outliers compared to guards and forwards.

4.2 Predicting Game Outcomes

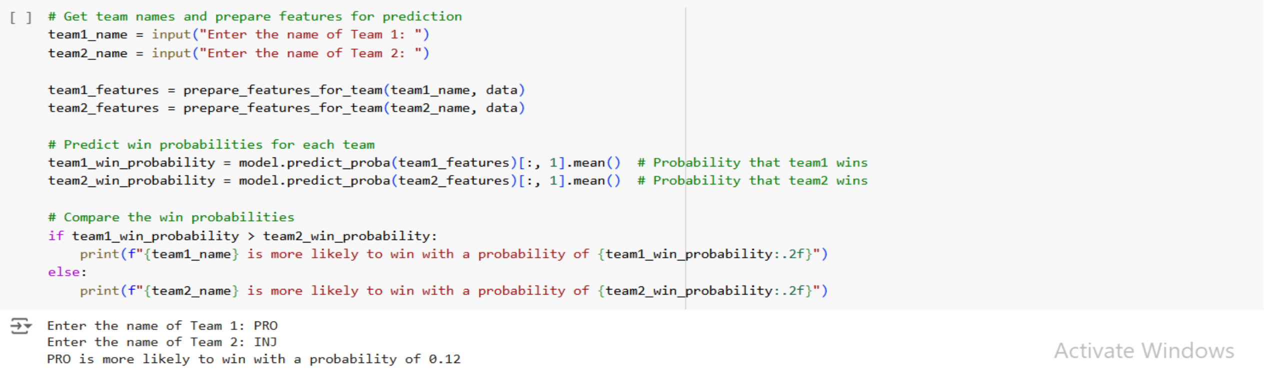
The model reached an accuracy of approximately 91% on the test set. The SVM's linear decision boundary performed well on the data, although some non-linearity might have improved results in cases of complex interactions. SVM training was efficient due to the linear kernel, with fewer parameters to optimize compared to non-linear kernels. SVM was computationally efficient with fewer parameters, making it suitable for large datasets or applications needing quick predictions. The predictions showcase the model's ability to provide insights into potential game outcomes based on the input features, contributing to the broader application of machine learning in sports analytics.



```
# Initialize and train the SVM model
model = SVC(probability=True, kernel='linear', random_state=42)
model.fit(X_train, Y_train)

# Evaluate the model on the test set
Y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(Y_test, Y_pred))
print("Classification Report:\n", classification_report(Y_test, Y_pred))
```



```
Accuracy: 0.9117647058823529
Classification Report:
              precision    recall  f1-score   support

           0       0.92      0.90      0.91       120
           1       0.90      0.92      0.91       118

    accuracy                           0.91       238
   macro avg       0.91      0.91      0.91       238
weighted avg       0.91      0.91      0.91       238
```



```
[ ]  # Get team names and prepare features for prediction
     team1_name = input("Enter the name of Team 1: ")
     team2_name = input("Enter the name of Team 2: ")

     team1_features = prepare_features_for_team(team1_name, data)
     team2_features = prepare_features_for_team(team2_name, data)

     # Predict win probabilities for each team
     team1_win_probability = model.predict_proba(team1_features)[:, 1].mean()  # Probability that team1 wins
     team2_win_probability = model.predict_proba(team2_features)[:, 1].mean()  # Probability that team2 wins

     # Compare the win probabilities
     if team1_win_probability > team2_win_probability:
         print(f"{team1_name} is more likely to win with a probability of {team1_win_probability:.2f}")
     else:
         print(f"{team2_name} is more likely to win with a probability of {team2_win_probability:.2f}")
```

```
Enter the name of Team 1: PRO
Enter the name of Team 2: INJ
PRO is more likely to win with a probability of 0.12
```

The outcomes of the simulated matchups between Team 1 (PRO) and Team 2 (INJ), were determined using a machine learning model. The model predicted that Team 1 would win in the first scenario and that Team 2 would emerge victorious in the second. These results are based on the historical team statistics and the trained neural network's evaluation of the features representing each team's performance.

5. Conclusion

In conclusion, the application of the Isolation Forest model for outlier detection effectively highlighted players with standout performances across different basketball positions. By segmenting the data into guards, forwards, and centers, the model was able to account for the unique characteristics and contributions of each position, identifying a range of outliers that may represent top-performing or unusually distinctive players.

The analysis revealed a higher proportion of outliers among guards and forwards, likely due to the variability and versatility in these roles. In contrast, centers, with a more defined set of responsibilities, showed fewer outliers, reflecting the positional consistency in performance metrics. This approach provides a valuable framework for identifying exceptional players within their positional context, allowing coaches, analysts, and teams to focus on those who demonstrate potential beyond typical performance boundaries.

SVM remains a valuable option for game outcome prediction due to its simplicity, speed, and interpretability, especially when computational resources are limited, or when a quick baseline model is required. This model is a reliable classifier for predicting game outcomes based on straightforward statistical features, though its effectiveness may be enhanced by exploring non-linear kernels or other algorithms when higher accuracy is critical. The level of accuracy suggests that SVM can distinguish between winning and losing teams based on team statistics with reasonable confidence. The model's linear kernel provided an interpretable decision boundary, making it a suitable choice for applications where understanding the classification logic is important.