IPL Winning Team Prediction

- *Chaturya Katragadda*
- *Mahima Chowdary Maddineni*

Introduction

The Indian Premier League (IPL), launched in 2008, has transformed the cricket landscape by making an enormous impact. Its popularity has skyrocketed, turning it into the premier cricket league globally. The primary objective of this project was to create a forecasting model to assist teams in predicting scores and winners during matches. Predictions were made using select features based on the available data at various stages. The study included seasons from 2008 to 2020, with the IPL 2020 season posing unique challenges due to its relocation from India to the UAE. This shift required data analysts to adapt, given that most previous matches were played in India.

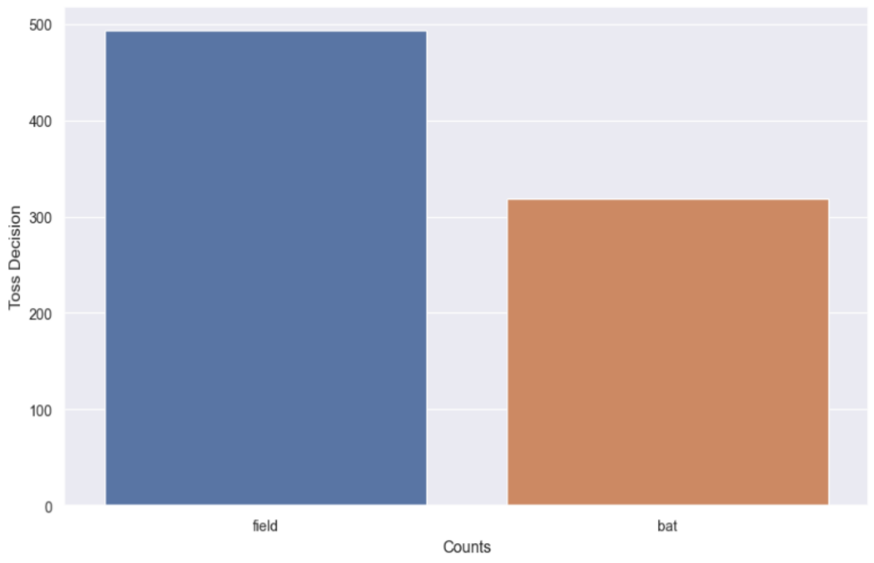
The 2022 season also brought significant changes in team dynamics and strategies. Looking ahead, with the addition of two new teams in 2022, teams have been allowed to retain up to four players, leading to intense competition for player acquisition.

Some of the useful features present in the dataset are year, venue, run(s) and wicket (if any) on every ball, toss decision, batsman and bowler, result of match with margin etc. The data contains some minor inconsistencies, such as missing city names, missing match results, and incorrectly spelled team names. However, these issues are relatively insignificant given the overall data size. To improve prediction accuracy, pre-processing was performed on the data, followed by feature engineering.

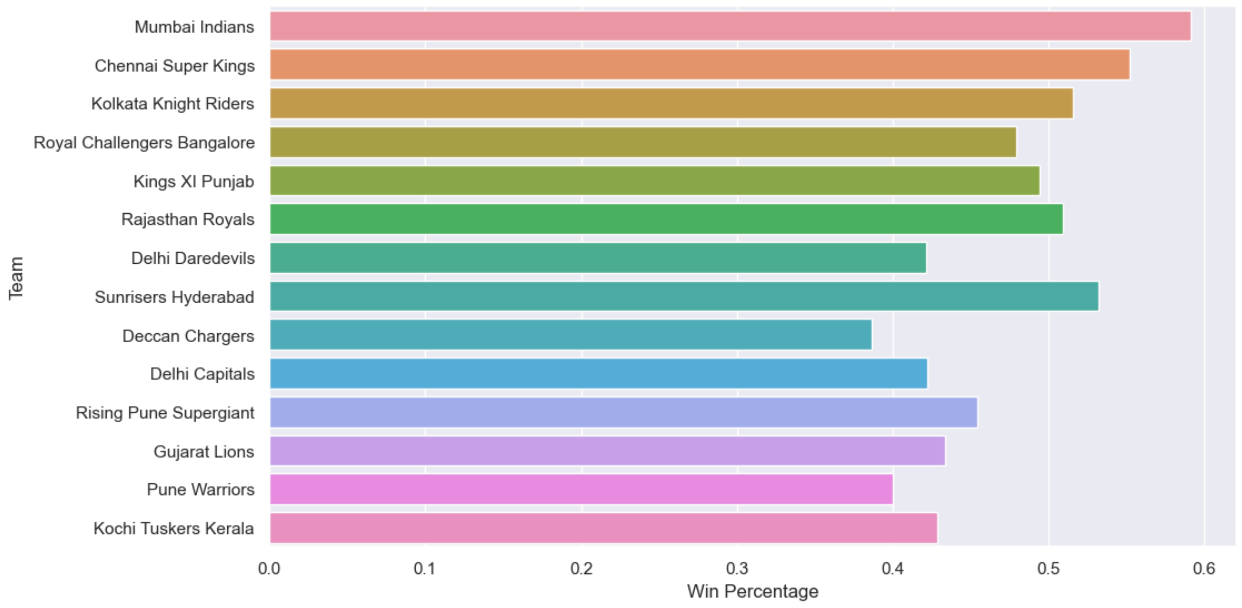
Exploratory Data Analysis

IPL is a very competitive league this is a very difficult task. So, for getting an understanding of which features to use for predictions EDA was done. Comprehensive and in-depth analysis of the data available was done to gain useful insights about teams and players so as to extract important features and discover interesting facts.

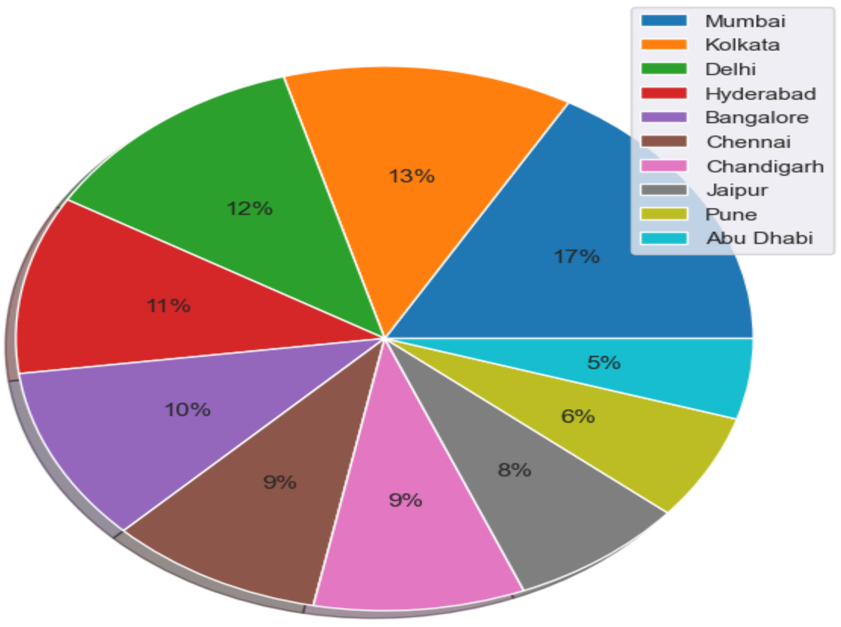
Most of the Wins are from Mumbai Indians and Chennai super kings, About 5 out of 8 teams i.e. almost 62% teams chose to bowl after winning the toss instead of batting first.



Most rivalry is between CSK and MI in which both the teams have performed equally well against each other. MI has highest win percentage, then followed by CSK and SRH. Deccan Charges has least win percentage as that team has be renamed to sun risers Hyderabad. The win percentage is indicated by the figure below.

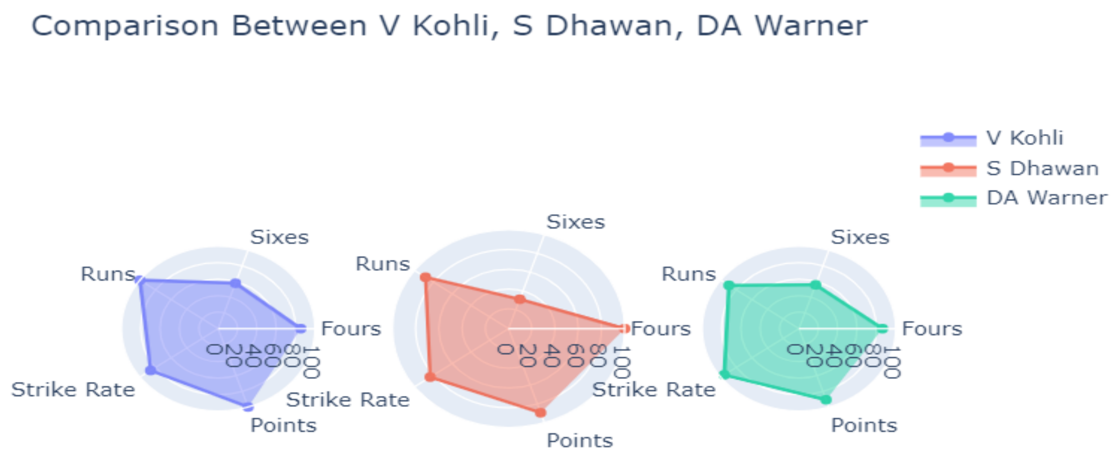


We can see Mumbai is the top city to host IPL matches with about 17% of the matches being held there. Only having one tournament in Abu Dhabi, UAE, also shows up in the top 10 list due to the limited number of grounds available in the country.



In the mega Auction that took place in 2022, teams will prioritize selecting players based on their skills to enhance team performance. Providing player statistics, as illustrated in the figure, can assist team owners and management in comparing different options and making crucial decisions regarding team selection more effectively.

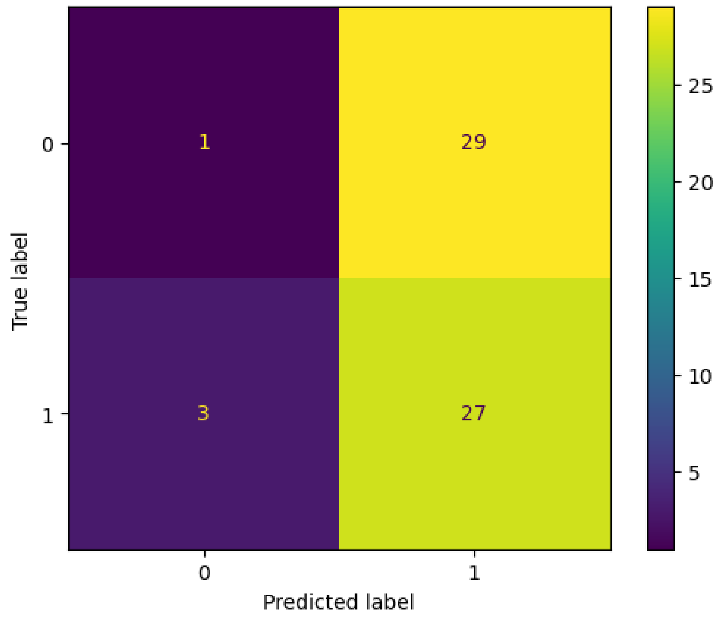
An observation made by us is that despite Shikhar Dhawan's higher frequency of scoring fours compared to sixes, he maintains a superior strike rate and accumulates more runs than Virat Kohli and David Warner. This aspect makes him an excellent choice for team selection, as he not only scores more runs but also has a lower probability of getting dismissed.

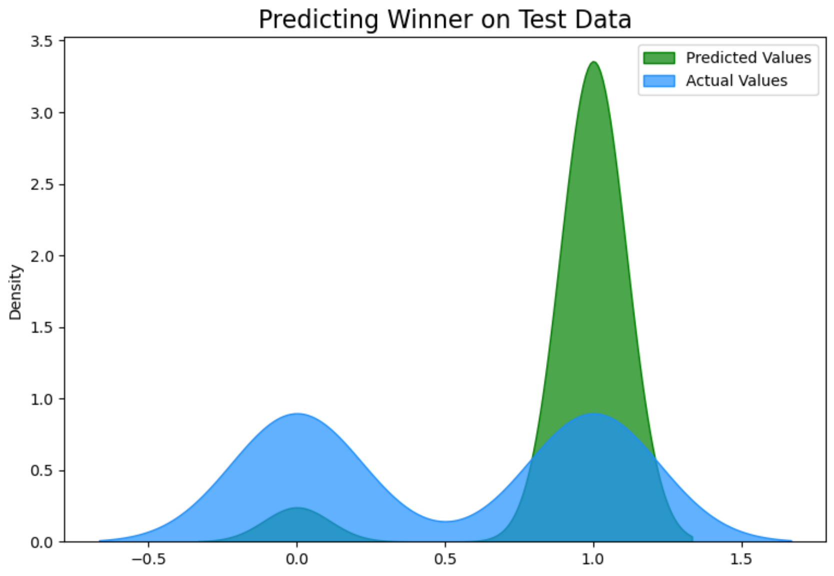


Feature Engineering and Code with less number of features

Logistic Regression

{'F1_score': 0.6363636363636364, 'Accuracy': 0.4666666666666667, 'Area under ROC curve': 0.4666666666666667, 'Precision': 0.4827586206896552, 'Recall': 0.9333333333333333}

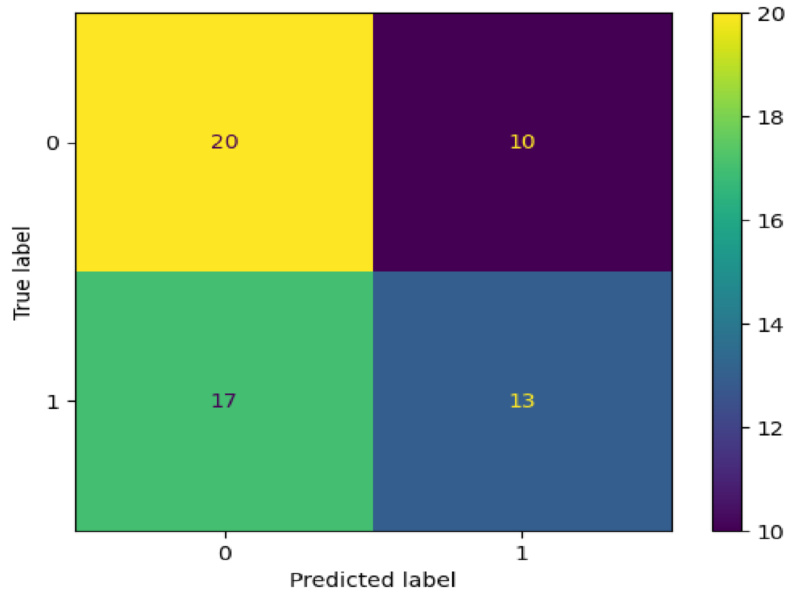


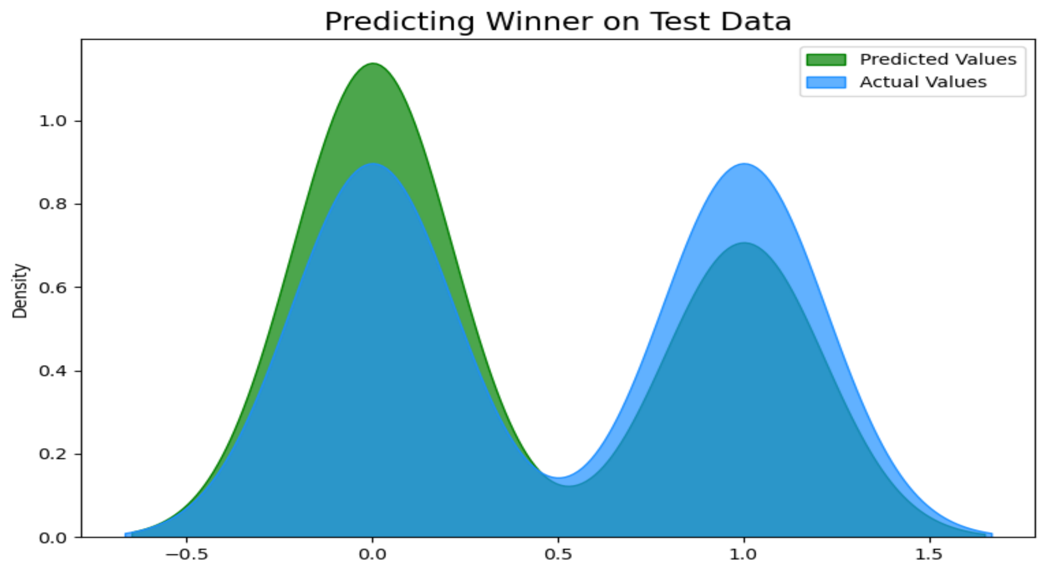


Decision Tree Classifier

{'F1_score': 0.49056603773584906, 'Accuracy': 0.55, 'Area under ROC curve': 0.55, 'Precision': 0.5652173913043478, 'Recall': 0.43333333333333335}

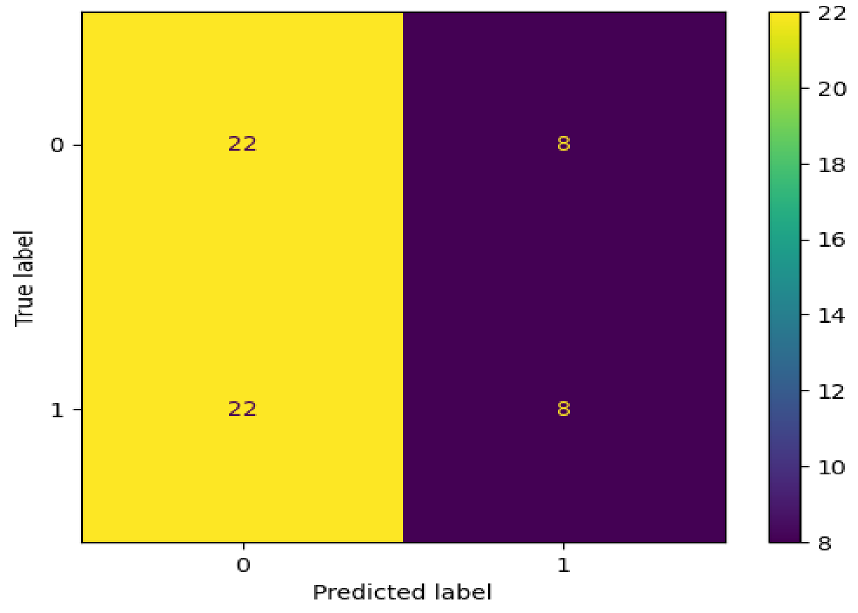
The Decision Tree Classifier is a supervised machine-learning algorithm that mimics human decision-making by employing a set of rules. It constructs a tree-like structure where each internal node corresponds to a feature, branches denote decision rules, and leaf nodes signify outcomes. During the tuning process, it was observed that the accuracy fluctuated randomly with changes in the maximum depth parameter. Interestingly, selecting a maximum depth beyond 10 yielded comparable results, indicating a saturation point in model performance.

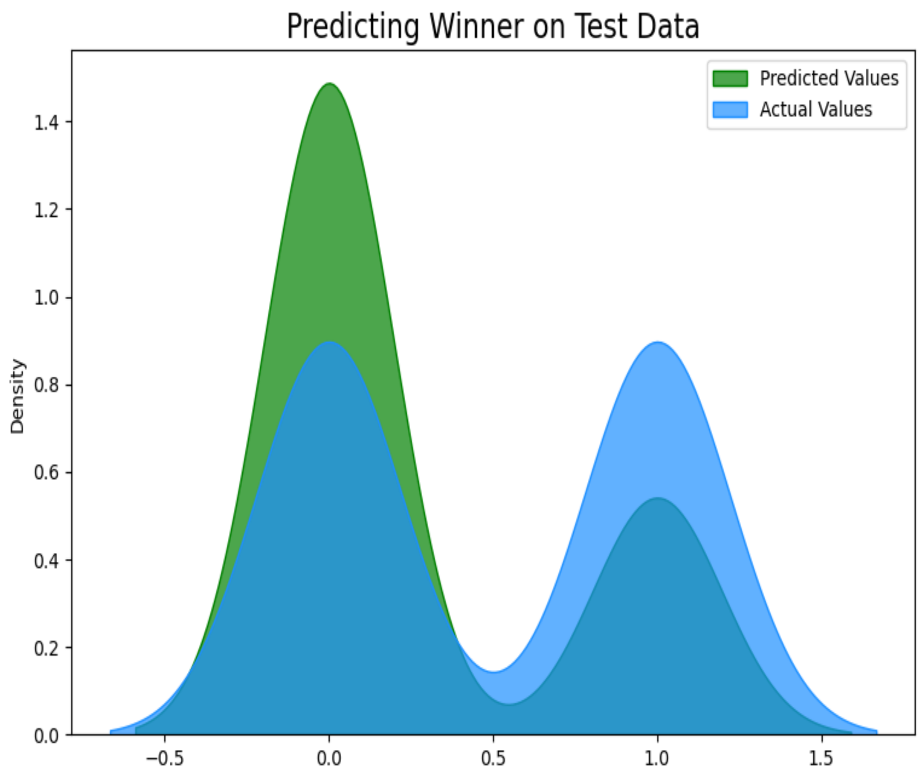




Randon Forest Classifier

```
{'F1_score': 0.3478260869565218, 'Accuracy': 0.5833, 'Area under ROC curve': 0.5, 'Precision':
0.5, 'Recall': 0.26666666666666666}
```



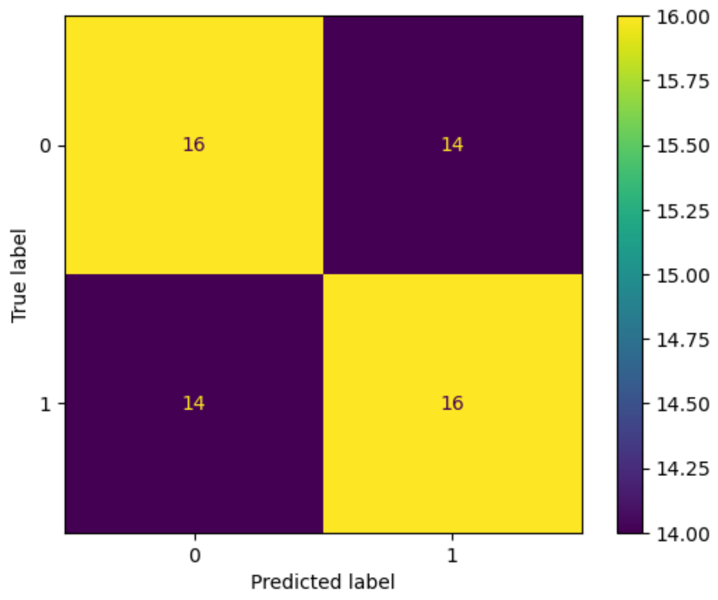


These were the initial values for the Random Forest Classifier, then tried fine tuning on lower n_estimators for RF Classifier.

We created a Random grid to search for the best hyperparameters, first created a base model to tune, searches across 100 different combinations and uses all available cores, then we fitted the random search model.

```
{'n_estimators': 50, 'min_samples_split': 5, 'min_samples_leaf': 4, 'max_features': 'sqrt', 'max_depth': 50, 'bootstrap': False}
```

The best score obtained was :0.6037290836653386

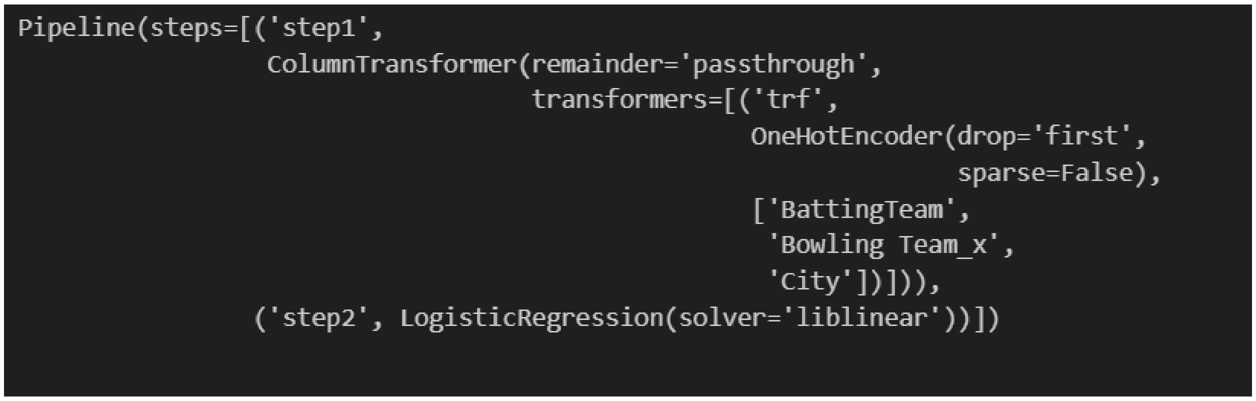


In our analysis, we've chosen to use fewer features during the first inning due to the absence of real-time data on second-inning players' performance. To improve accuracy, we suggest starting the winner prediction process at the beginning of the second inning to access updated information and gain better insights into the ongoing match.

Model	***Accuracy***
Logistic Regression	*0.46*
Decision Tree Classifier	*0.43*
Random Forest Classifier	*0.60*

Feature Engineering and Code when considering more Features.

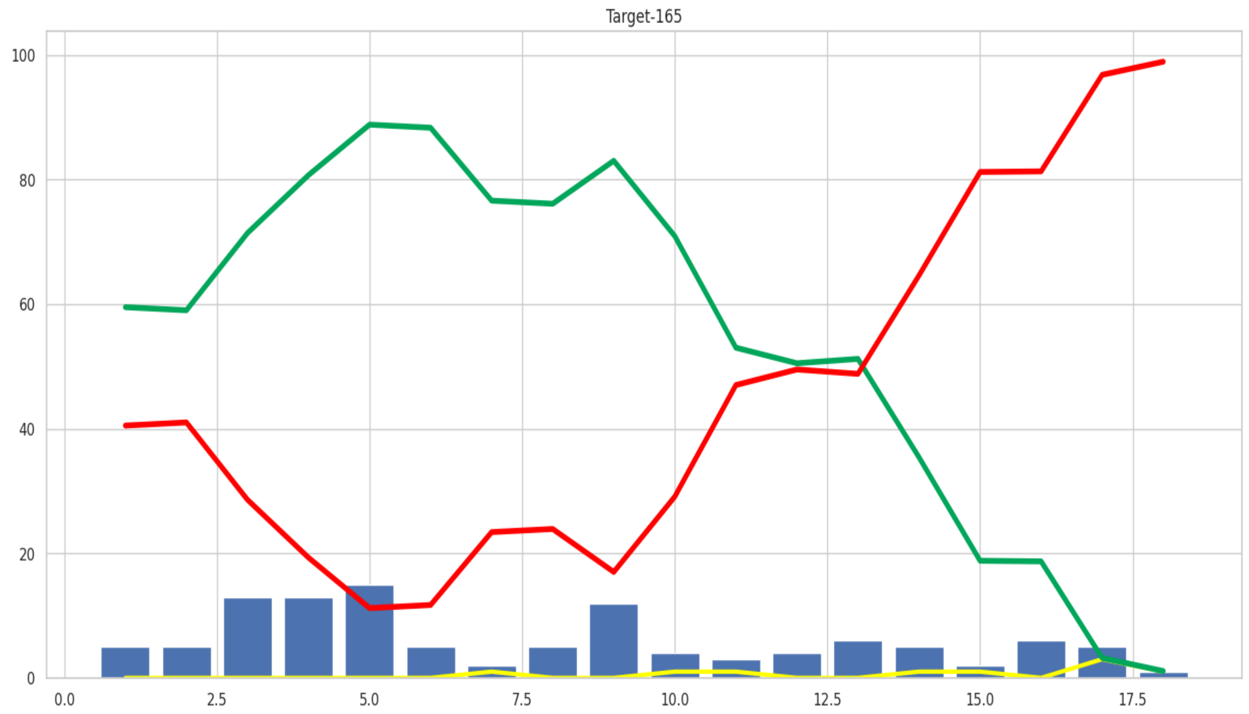
train first model is Logistic Regression then applied ColumnTransformer on 'BattingTeam','Bowling Team_x','City'



```
Pipeline(steps=[('step1',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('trf',
                                                  OneHotEncoder(drop='first',
                                                                sparse=False),
                                                  ['BattingTeam',
                                                   'Bowling Team_x',
                                                   'City'])])),
                ('step2', LogisticRegression(solver='liblinear'))])
```

I am training 3 models :- Logistic Regression, random forest classifier and decision tree classifier. In this 3 model was Logistic Regression accuracy is 81%, random forest classifier accuracy is 99% and decision tree classifier accuracy is 98%.

random forest classifier and decision tree classifier the accuracy was approximately 99% but i but Logistic Regression gave me an accuracy of 81% it was good but I expected to 90% but it is good in Logistic Regression model, given both the sides win probability was (win or loss) and the best model to finding win probability was (win or loss) so I am use this model.



This graph Represents over by over wining probability. In this graph green line is win and red line was loss and yellow line is wickets on over. Also, the target is 165. second inning will state the batting team win probability is 56% and bowling team probability is 44%. In this match 6 overs are left, the batting team winning probability is 84% and bowling team probability is 16%.

But when 11 overs are left the winning probability was 50-50 on both the teams. When, Batting team has 3 wickets, winning probability decreases e and bowling team winning probability was increases. In 14th and 15th over they have 2 more wickets for the batting team, the winning probability of the batting team is decreased and bowling team winning probability was increased that is, batting team winning probability is 17% and bowling team winning probability is 83% at the end of the 15 over

In over no 17 they give 3 wickets, winning probability of batting team is again decreased and bowling team winning probability was again increased, batting team winning probability is 3% and bowling team winning probability is 97% at the end of the 17 over and the end of the 18th over, batting team winning probability is 1% and bowling team winning probabilityis 99%

So the Bowling team will be winning the match.

Model	***Accuracy***
Logistic Regression	*0.81*
Decision Tree Classifier	*0.99*
Random Forest Classifier	*0.99*

Score Prediction

Custom accuracy was defined to consider the predicted score as correct if **|Predicted score-Actual score| ≤ 10** runs and wherever accuracy is mentioned ahead refers to this custom accuracy unless mentioned otherwise. The features considered are

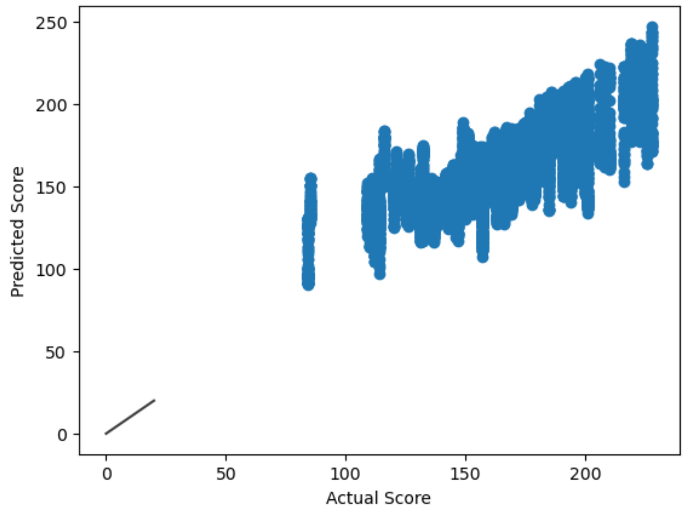
- Innings
- Overs
- Venue
- Batsman
- Bowler
- Current Runs
- Team 1
- Team 2
- Team1 _toss_win
- Team1_bat
- Runs in Last 5 overs
- Wickets in Last 5 overs

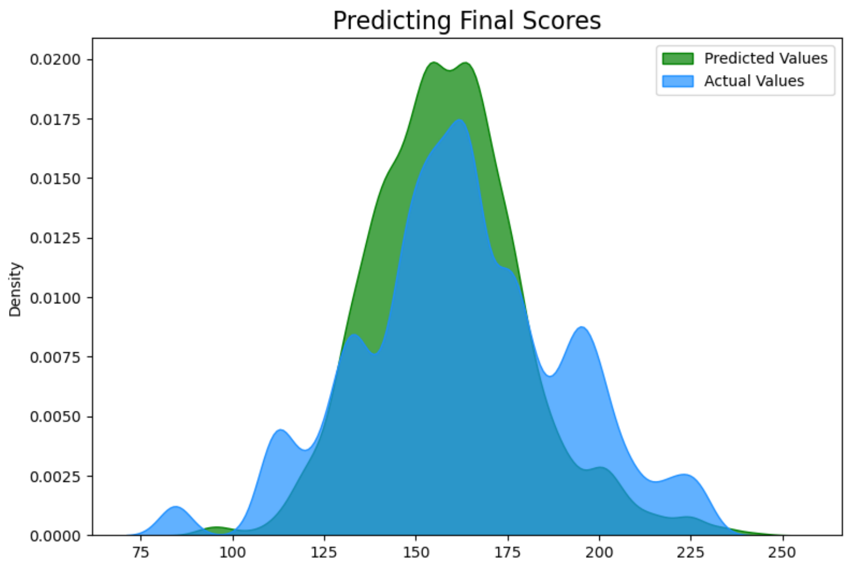
Also, gave these values respectively, Team1 toss win is 1 when Team 1 wins the toss and otherwise 0. Team1 bat is 1 if Team 1 is batting first and otherwise 0.

Linear Regression

Linear regression is a linear ap proach for modelling the relationship between the target response and one or more features which are being used for prediction. Ordinary Least Squares(OLS) Linear Regression was used.

```
{'MAE': 14.691412694278023, 'RMSE': 19.380890360733822, 'R2score': 0.5255066839745284, 'Custom Accuracy': 47.19979161239906}
```

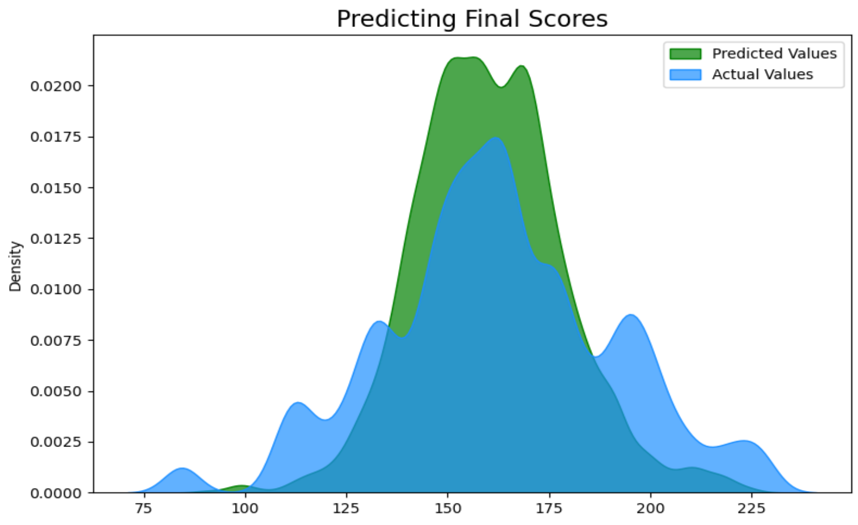




Support Vector Regression (SVR) :-

SVR, a supervised machine learning algorithm derived from Support Vector Machines (SVM), is commonly utilized for classification purposes. It revolves around the concept of a hyperplane, which is essentially a straight line used to fit the data. The primary goal of SVM is to identify a hyperplane within an n-dimensional space that effectively separates the data points into distinct classes. These data points closest to the hyperplane, known as Support Vectors, play a crucial role in determining the position and orientation of the hyperplane, thus contributing significantly to the construction of the SVM model.

{'MAE': 14.631978190368686, 'RMSE': 19.28656934930026, 'R2score': 0.5301138805268051, 'Custom Accuracy': 45.74976122254059}

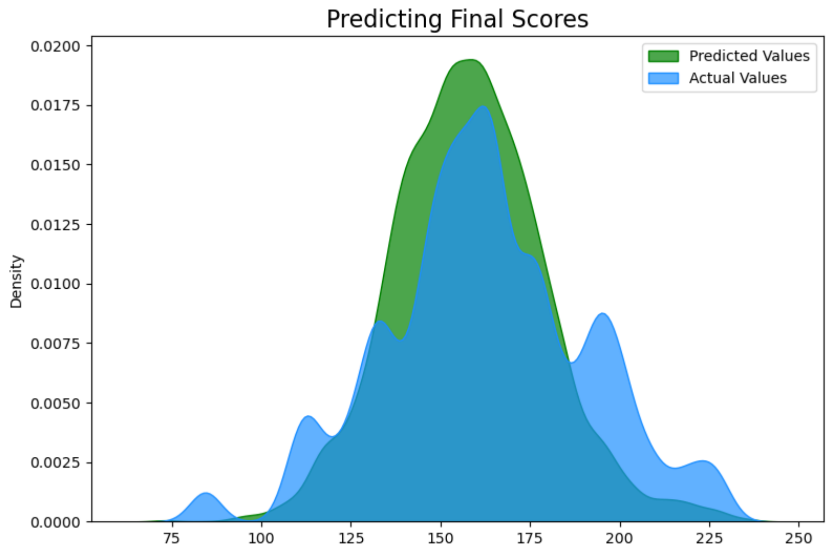


Random Forest Regression

Random Forest (RF) Regression is a supervised learning technique that employs ensemble learning for regression tasks. Ensemble learning combines predictions from various machine learning algorithms to enhance predictive accuracy compared to individual models. RF utilizes multiple decision trees as base models, which operate in parallel to reduce variance and improve accuracy. However, the RF model's higher complexity can lead to overfitting if its hyperparameters are not appropriately tuned.

The hyperparameters considered for tun ing were n estimators i.e. no. of trees in the forest and max depth which is the maximum depth of a tree. Inititally, Grid Search was run on the following param grid : {"n estimators": [250,500], "max depth": [50,100,250] }

{'MAE': 15.57745263523487, 'RMSE': 20.90106237926074, 'R2score': 0.4481521276444853, 'Custom Accuracy': 45.76712685595207}



We were unsuccessful for score prediction, when we were working on Winner prediction, we got an idea of doing the score prediction as well and tried implementing that but was able to get only 45 to 50 % accuracy which was less than flipping a coin.

We even have changed the accuracy function for plus or minus 10 for the score, still we couldnt get the accuracy better, also, we tried different models as well but unable to achieve a better accuracy.