STATISTICAL ANALYSIS OF IPL

Submitted by

ADITYA ROSHAN	**19BCE0911**
MAYANK TIWARY	**19BCE0918**
ANIKET MISHRA	**19BCE2062**
SAMRIDDHI AGARWAL	**19BCE2382**

Prepared For:

DATA VISUALIZATION (CSE3020)

SLOT: L41+L43 D1(THEORY)

Submitted To:

Ms. Nalini N

Assistant Professor Sr.

School of Computer Science and Engineering

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

ABSTRACT

With technology growing more and more advanced in the last few years, an in-depth acquisition of data has become relatively easy. With millions of people following the Indian Premier League (IPL), developing a model for predicting the outcome of its matches is a real-world problem. As a result, Machine Learning is becoming quite a trend in sports analytics because of the availability of live as well as historical data. Sports analytics is the process of collecting past matches data and analyzing them to extract the essential knowledge out of it, with a hope that it facilitates effective decision making.

Indian Premier League, IPL provides the most successful platform for cricket as it gives opportunities to young and talented players to showcase their talents on various pitches. Sports analytics and data visualization can play a crucial role in selecting the best players for a team and also help players increase their field performance**. Our project is about the toss related analysis and the breadth of data visualization, supporting our goal in predicting the match winner based on collective statistics from 2008 to 2019. The motive behind applying a visual analysis to it, is to gain insight into what is likely to happen in the future.**

OBJECTIVE

We will apply visualization techniques to the datasets to provide a deeper insight and pave way for recommendations to the player or team. The visual form of data is more easily understandable over numbers and text. And with the ease of obtaining and storing data, we will try to attempt and perform a toss related analytical and machine learning technique to engineer a model so that we can predict the match winner based on past records from 2008 to 2019.

INTRODUCTION

This paper presents a method that is aimed towards predicting a cricket player's upcoming match performance by implementing machine learning algorithms. The proposed model consists of statistical data of players of Bangladesh national cricket team which has been collected from trusted sports websites, feature selection algorithms such as recursive feature elimination and univariate selection and machine learning algorithms such as linear regression, support vector machine with linear and polynomial kernel. To implement the proposed model, the accumulated statistical data is processed into numerical value in order to implement those in the algorithms. Furthermore, aforementioned feature selection algorithms are applied for extracting the attributes that are more related to the output feature. Additionally, the machine learning algorithms are used to predict runs scored by a batsman and runs considered by a bowler in the upcoming match. The experimental setup demonstrates that the model gives up to 91.5% accuracy for batsman Tamim and up to 75.3% accuracy for bowler Mahmudullah whereas prediction accuracy for other players are also up to the mark. Therefore, this will help in calculating player's future performance and thus will ensure better team selection for forthcoming

cricket matches.

SPORTS statistical analysis use in sports has been growing quickly year by year. Due to which the ways in which game strategies are formed or the player's evaluation criteria has been changed but also has the got the more interest of audience towards cricket. Now Cricket has become one of the most followed team games in the world with billions of fans all across the globe. Cricket is a sports game that played globally across 106-member states of the International Cricket Council (ICC), which has 1.5 billion worldwide fans according to ICC. However, much of the global finance and interest is focused upon the 10 full ICC member nations and more specifically upon 'the big three' of England, Australia and India. Cricket has many evolutions over time. Today, there are three major formats in which cricket is being played internationally, One Day Internationals (ODIs) and the T20 cricket and Test Matches. Besides these international cricket matches, T20 League cricket is getting attention in the fans due to its shortest format and the most exciting format of the game. Indian Premiere League is one of most popular t20 cricket league in the world. Using Computer Intelligence to analyze and model the game of Cricket is a promising research area. The increased popularity and financial benefits

have made Cricket an interesting sport to be subjected to statistical analysis and machine learning. The dynamic nature of Cricket, complex rules governing Cricket makes the task a challenging one. The various approaches taken and what has been disclosed from available work is neither very clear nor properly documented due to the differences in the approaches. If the good and the drawbacks of the existing work is properly analyzed and documented, it will assist in future researches. This paper presents an analysis of the existing work related to match outcome prediction in the Cricket domain. This paper is a result of an ongoing research, by the end of the research we hope to address the missing links and the drawbacks that will be explored in this paper.

DATASETS AND DESCRIPTIONS OF DATASETS USED

While dealing with data, Kaggle: Your Home for Data Science is the to-go platform to get any sort of dataset.

We will use the dataset https://www.kaggle.com/nowke9/ipldata for visualizing the different parameters throughout the IPL season. The dataset has 2 files: *matches.csv* having every match detail from 2008 to 2019 and *deliveries.csv* having ball by ball detail for every match.

For the prediction of the winner of the match we made use of the datasets from https://www.kaggle.com/veeralakrishna/ipl-points-table-2008-to-2019 which contains 12 datasets having the points table from the year 2008 to 2019.

LITERATURE SURVEY

[1]Bangladesh Cricket Squad Prediction Using Statistical Data and Genetic Algorithm

Cricket is the most famous game in Bangladesh and has turned into an indistinguishable piece of our way of life. This game is played between two groups where each group has 11 players. Every one of these players plays an alternate part, like Batsman, Bowler, All-rounder and Wicket keeper. Individuals of Bangladesh are very worry about the public cricket crew before any match or any series. Other than these ideal crew and mix choice is expected to dominate any cricket game or any series. In this article a model is proposed to choose an ideal cricket crew utilising statistical information and hereditary calculation. The most recent 2 years performance of all Bangladeshi players in public association and international matches is considered to predict every player performance and select top 30 players utilising statistical information. Then, at that point, hereditary calculation is applied on these 30 players to predict last Bangladeshi public cricket crew of 14 individuals. The predicted crew in this proposed model is just substantial for one day international(ODI) match. Yet, this model is nonexclusive and with appropriate information t20 and test crew will likewise be predicted.

Choice of a decent crew in cricket is essential to its prosperity.
The presentation of a cricket crew relies upon its group part execution and their blend. To foresee performance of each cricket player or a group, there are multiple ways to do. This exhibition is anticipated in view of recorded information.These

model for cricket squad selection is based on statistical analysis or genetic algorithm or analytical hierarchy process or evolutionary multi-objective optimization or random forest algorithm. In statistical analysis only statistical data is considered to predict the team and in genetic algorithm team is predicted by applying genetic algorithm on some selected player to predict the final squad from these players.

Strengths -

1. They considered all 3 years international matches data of past 3 years of all Bangladeshi national level cricket players.
2. They preprocess and determine each player's performance using java and statistical data of each player. From these performances select top 30 players (where 10 specialist batsmen, 12 special- ist bowlers, 6 all-rounders and 2 Wicket keepers included) by predicting their future performance.
3. Then we have applied GA (which includes selection, crossover, mutation and survivor selection) on these 30 players for selecting final 1 4 squad. We have calculated fitness value for each squad and run GA several times on our system.

Weaknesses -

They calculated squad is nearly similar to our current Bangladesh ODI squad of 14 members. There are 2 all- rounders, 6 batsmen, 5 bowlers and 1 wicket keeper in our predicted squad. This list include 4 top order batsmen and 2 middle order batsmen, 3 fast bowlers, 1 off spinner and 1 leg spinner. All rounder will perform as middle or lower order batsman and off-break spinner bowler. The accuracy could be a bit more

[2] A Study on Performance of Cricket Players using Factor Analysis Approach

In this paper, they have made an attempt to study the performance of Cricket Players

using Factor Analysis technique. For this , the dataset of 85 batsmen, 85 bowlers; and 95 batsmen, 95 bowlers have been considered from IPL9, 2016 (20 overs) and ICC World Cup, 2015 (50 overs) respectively, and the findings of this study reveals that batting capability dominates over bowling capability which is in conformity with an earlier study on same kind of game.

They have done factor analysis is a statistical technique to study the interrelationship among variables in an effort to find a new set of factors, fewer in number than the original variables. It is one of the widely used methods of multivariate data analysis .Their purpose of factor analysis is to obtain a reduced set of uncorrelated latent variables using a set of linear combinations of the original variables to maximize the variance of these components. The factor analysis which was performed using Principal Component Analysis (PCA) as they explain items validity as well as groups of items into meaningful clusters, and for suitable rotation policy, an orthogonal varimax rotation was used as it assists in optimizing the number of variables.

Strengths -

We found out that the variance explained by factor1 (batting) is much higher than the variance explained by factor2 (bowling).

They took the latest dataset

Weaknesses

Their accuracy level could be more

[3] Data Mining and Machine Learning in Cricket Match Outcome Prediction: Missing Links

This paper, started gaining the required domain knowledge with the literature survey and the authors identified 3 main problem domains that are linked with Cricket match outcome prediction,

1. Cricket Player Performance analysis.

2. Cricket match simulation.

3. Cricket Team Selection.

The existing work on predicting the outcome of a Cricket match there has been substantial interest around the above identified problem.

Strength-

Researchers have taken various approaches to solving the problem, in which they identified 2 main approaches used.

1. Using historical Cricket data.

a. Classification using team and categorical data.

b. Match Simulation.

c. Classification using team composition data.

2. Using Collective knowledge (Social Media).

Weaknesses-

Dealing with the lack of data about certain players: When taking a team composition approach, where the relative strength of a team is calculated from the statistics of the players in the team, a new player having only played few games or a playing making a debut can lead to inaccurate results, this issue was not addressed in any of the studies we explored.

Not giving enough consideration to the conditions the game is played in: As the game of Cricket has showed us time and time again, conditions in which the match is played play a pivotal role in the outcome, where the home team has an advantage over the visiting team. Therefore, it's vital that this is taken into consideration in the feature selection and engineering phase.

Taking an ensemble approach: they identified 2 main approaches taken by researchers so far, using historical cricket data and using collective knowledge from social media. No attempts have been made so far to use these two approaches in

tandem to improve the accuracy of the predictions made.

[4] Player's Performance Prediction in ODI Cricket Using Machine Learning Algorithms

This paper presents a method that is aimed towards predicting a cricket player's upcoming match performance by implementing machine learning algorithms. The proposed model consists of statistical data of players of Bangladesh national cricket team which has been collected from trusted sports websites, feature selection algorithms such as recursive feature elimination and univariate selection and machine learning algorithms such as linear regression, support vector machine with linear and polynomial kernel. The proposed model, the accumulated statistical data is processed into numerical value in order to implement those in the algorithms.
Furthermore, aforementioned feature selection algorithms are applied for extracting the attributes that are more related to the output feature. Additionally, the machine learning algorithms are used to predict runs scored by a batsman and runs considered by a bowler in the upcoming match.
Strenghts-
The experimental setup demonstrates that the model gives up to 91.5% accuracy for batsman Tamim and up to 75.3% accuracy for bowler Mahmudullah whereas prediction accuracy for other players are also up to the mark. Therefore, this will help in calculating player's future performance and thus will ensure better team selection for forthcoming cricket matches.

[5] The Cricket Winner Prediction With Application Of Machine Learning And Data Analytics

Daniel Mago Vistro, Faizan Rasheed, Leo Gertrude David

SPORTS statistical analysis use in sports has been growing quickly year by year. Due to which the ways in which game strategies are formed or the player's evaluation criteria has been changed but also has the got the more interest of audience towards cricket. Winning a cricket match depends on multiple factors like batting, bowling, fielding, team performances and player performances. To predict the winner of a cricket match is never an easy task. But there are always some kinds of unique aspects or match conditions that may favor to some team and sometime does not such as home advantage, Key Players, Pitch Conditions and weather conditions.

The SEMMA process was developed by the SAS Institute that considers a cycle with 5 stages for the process: Sample, Explore, Modify, Model, and Assess. Data mining is the process of discovering predictive information from the analysis of large databases. As the results by Decision Tree model were not perfect according to the requirements so we need to fine tune the parameters of Decision Trees. A machine learning model consists of various parameters which decide how different

computations will be performed in selected models. Usually the predictions of data are made by parameters that has been already set by default in models but in some cases the results are not good enough because of different nature of data.

On selected features several machine learning models has been applied on the to predict the winner and the results were outstanding. First of all, Decision Tree model was applied which predicted the match winner with good accuracy 76.9%. To improve the performances of model, we fine-tuned the parameters of Decision Tree model and achieve good results. Model performance was enhanced by 76 % to 94%. Then we applied Random Forest model on the selected features and the predicted the winner with 71% accuracy which was not good enough, so Random Forest Model was also tuned by parameter's tuning and results got better with 80 % accuracy.

[6] Cricket Match Analytics Using the Big Data Approach

Mazhar Javed Awan 1,* , Syed Arbaz Haider Gilani 1 , Hamza Ramzan 1 , Haitham Nobanee 2,3,4,* , Awais Yasin 5 , Azlan Mohd Zain 6 and Rabia Javed

The experimental results are measured through accuracy, the root mean square error (RMSE), mean square error (MSE), and mean absolute error (MAE), respectively 95%, 30.2, 1350.34, and 28.2 after applying linear regression in Spark ML. Furthermore, their approach can be applied to other sports. This learning is categorized into significant types such as clustering, associations, and dimensionality. The primary objective of these machine learning algorithms and models is to get valuable data patterns within complex and massive datasets. The Apache Spark framework is a highly effective and highly recommended framework by most analyst communities and in research. It is an open-source, easy-to-use, and freely accessible framework with a high-performance rate.

In this paper, we have built a prediction model to predict the winner of the match. For this reason, we have made two models using traditional methods and frameworks. Then, we checked the best model with high accuracy—the model with high accuracy was selected for further work. Based on a high accuracy rate, the model decided on will be built again using the Spark machine learning framework, in order to check which one is better suited to make a prediction model.

Linear regression is a predictive analysis algorithm that is used to predict something according to your choice. Moreover, it is a commonly used basic prediction algorithm, which is used by most of the analyst community in the prediction process.

The primary reason for building this model is to check which framework is better in terms of accuracy. In this model, we have done the same data visualization again and applied the descriptive data analysis on our dataset by checking the covariance of the data attributes that are here considered. The accuracy of the model using this Apache

framework is better than the traditional machine learning Scikit learn. In the Spark machine learning framework, the model's accuracy is increased by 96% as training and 94.5% as testing accuracy.

[7]

One of the primary approaches used in sport analytics research is machine learning. Machine learning techniques are utilised to predict the match result variable by developing classification models based on certain independent variables such as player's position, weather, location, etc. . The process involves training the model based on previous matches played, then the developed model gets evaluated on an independent future match to measure its effectiveness . Often, machine learning models' effectiveness is measured using metrics such as predictive accuracy and error rate among others [30]. Since cricket matches are recorded using multiple independent variables within a historical dataset and one dependant variable, (the outcome of the match) this problem can be dealt with using predictive analytics (classification methods) within machine learning.

To answer the research question different machine learning approaches are experimentally evaluated including probabilistic, Random Forest, statistical and Decision Trees. We used 10 years' data collected from the IPL-T20 tournaments [. More details on the data and empirical results. For this research, all experiments have been conducted on WEKA (The Waikato Environment for Knowledge Analysis), a machine learning tool which has automated intelligent techniques . To visually explore the IPL data insights, visualisation tool Tableau has been used. Different algorithms are adopted to deal with the research problem including Naïve Bayes, Random Forest, K-nearest neighbour and Model Decision Tree.

The FP rate for the Decision Tree algorithm is around 48% due to 147 instances misclassified as ‘‘Win” instead of ‘‘Lose”, while 144 instances were incorrectly predicted as ‘‘Lose” that were supposed to be ‘‘Win” by this algorithm. Similarly, for Naïve Bayes a lower rate was observed in terms of Precision and Recall rates, 49% of ‘‘Win” were misclassified leading to the Precision rate being 6% lower compared to that achieved on the Home team subset. KNN performs extremely well when processing the Toss Winner features set.

[8] The authors in discussed various key performance indicators to study the player performance in IPL from different countries. Cluster analysis has been applied on the datasets of players of IPL season 2010. The study reveals that players of England had performed well as a group and New Zealand players were the lowest performers.
[9] To predict the player performance in ODI using various Machine Learning Algorithm techniques is done in . Naïve Bayes, Decision tree, multiclass SVM and

Random forest are used to generate the prediction models for batsmen score and bowler's wickets for both the teams.

[10] The authors in predicted the result of a match by comparing the strengths of two teams. A performance of individual players from each team is measured by them. They implemented algorithms to predict the performances of batsmen and bowlers from past and recent career data.

[11] The authors in highlights the player performance especially batsmen and addresses the analysis that is done for Maximum Man of the Matches, Maximum Centuries Scored by Batsmen, Top Batsmen, Batsmen with Top Strike Rate, Top 10 Players with Maximum Runs. Statistics of 696 matches have been used in this experiment and even for toss related analysis such as Count of Toss wins, Decision taken by each team after winning the toss, Toss Decision Season Wise, Toss Decision Team Wise.

Sports analysis is a huge cluster of specific data and statistics. Sports analytics are the present and future of the professional sports era. The authors in [5] discussed a factor analysis approach, study the performance of cricket players and findings of his study say that batting capability dominates over bowling. The study reveals that the performance of bowlers is one of the crucial and significant factors which may change the scenario of matches. Coaches and selectors can include good all-rounder players to improve team results.

[12] The work in compared cricketers batting and bowling performances using graphical methods. Batsman and Bowlers record of season 2008 has been utilized for the analysis and interpretation of the graphs. Twelve bowlers and twelve batsmen were selected who bowled at least 100 balls and took at least four wickets and batsmen faced at least 100 balls had at least four completed innings. Random Forest gives the most accurate results for both the scenarios out of all the four techniques used.

PROPOSED WORK

Data Collection

While dealing with data, Kaggle: Your Home for Data Science is the to-go platform.

We used the dataset https://www.kaggle.com/nowke9/ipldata for visualizing the different parameters throughout the IPL season. The dataset has 2 files: matches.csv having every match detail from 2008 to 2019 and deliveries.csv having ball by ball detail for every match.

For the prediction of the winner of the match we made use of the datasets from h ttps://www.kaggle.com/veeralakrishna/ipl-points-table-2008-to-2019 which contains 12 datasets having the points table from the year 2008 to 2019.

Tools Used

Here, we used Python, Numpy, Pandas, Matplotlib and Jupyter notebook as our playground and hence the files are stored there.

Methodology

Visualizing the Datasets

- In the first phase, we filtered and cleaned the matches and deliveries datasets. This was a major pre-processing done for the collected data as most of the entries were empty.
- The datasets containing the details of every match from 2008 to 2019 are represented in the form of charts and graphs to get a visual presentation of the data.
- The data is visualized to get a better and clear understanding about all the parameters of the season, the teams, all-rounders, batsmen and bowlers, average wins and toss results affecting the match winners.
- Different new features were introduced such as the number of Total Matches played by the Teams for all the 11 Seasons, Maximum Man of the Matches, Maximum Runs and Centuries scored by Batsmen, Maximum player of the Match Awards, Maximum count of Toss wins by Different Teams, Decision taken by each Team after Winning the Toss, etc.

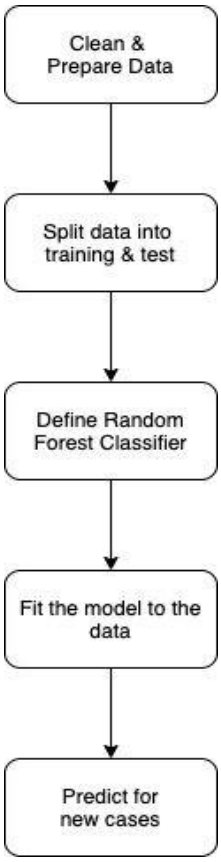
Match Winner Prediction

- We took points table from year 2008 to 2019 as an input, after dropping the redundant columns, we choose some custom parameters that are very crucial like 'toss_decision', 'field', 'bat', 'toss_win', 'team 1', 'team 2' etc and thus obtained a new points table.
- We then applied a Random Classifier and created a model to find out which team would win if a match was supposed to happen between the two teams.

We took points table from year 2008 to 2019 as an input, then we drop the redundant columns and at the end we choose some custom parameters that are very crucial like 'toss_decision', 'field', 'bat', 'toss_win', 'team 1', 'team 2' etc… and thus we obtain a new points table. Now we apply Random Forest Classifier and create a model to find out which team would win if a match was supposed to happen between the two teams.
The procedure followed is:

1. From the visualization choose most influencing factors (team points, run-rate, toss decision and toss-win)
2. Wrangle data and process into a new single dataframe
3. Split data into training and testing sets for training
4. Define model (Random Forest Classifier here)
5. Fit the model to the data

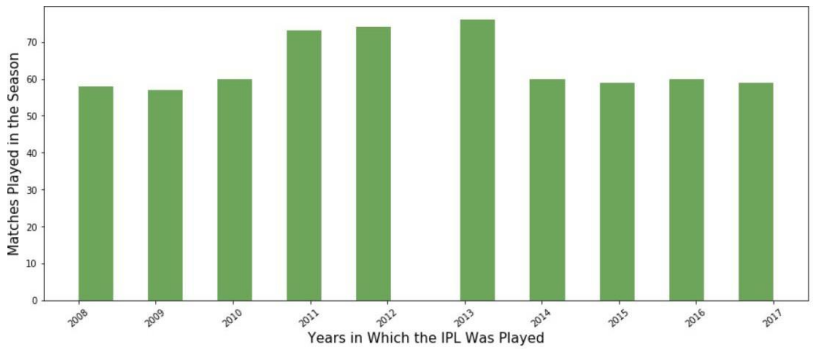
6. Create a prediction function to reuse again and again.
7. Finally, pass two team names, toss outcome and winner of toss to the prediction function and the model will return the predicted winner.



EXPERIMENTS/RESULTS

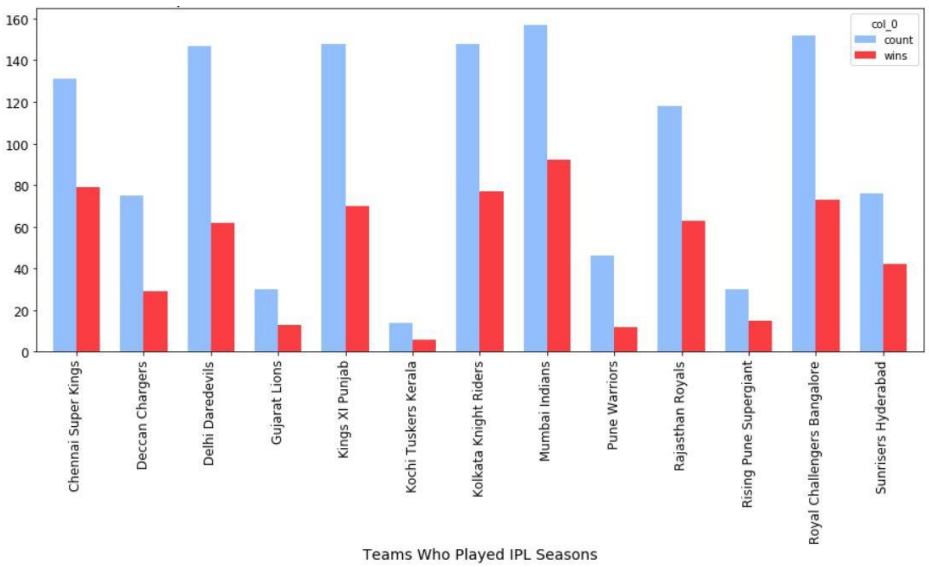
Results obtained through Visual Techniques applied to the given datasets

```
matches = pd.read_csv("matches.csv")
deliveries = pd.read_csv("deliveries.csv")
```



Histogram to show number of matches played in each season of the IPL

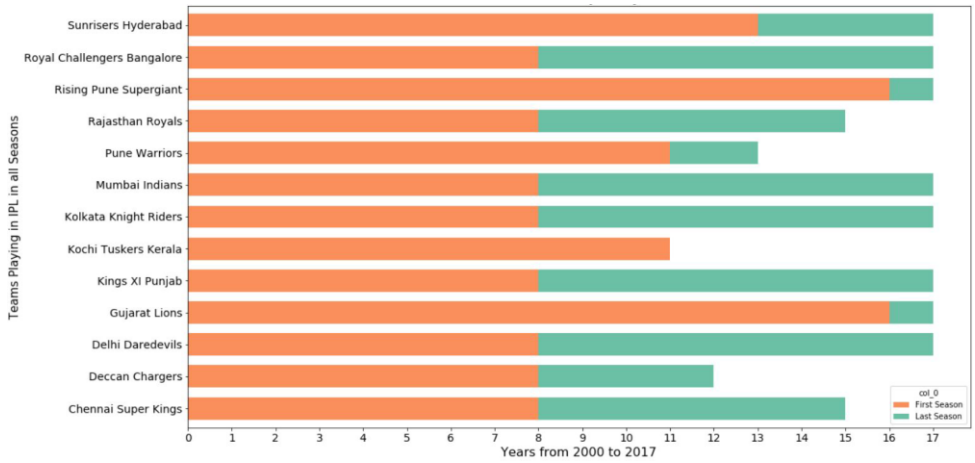
Analysis: From the, we see that the year 2011 to 2013 having 9 IPL franchises have the maximum number of matches during those seasons, which brings the average matches played per season to 63.6.



Grouped bar plot for each total matches and wins of each team in all seasons

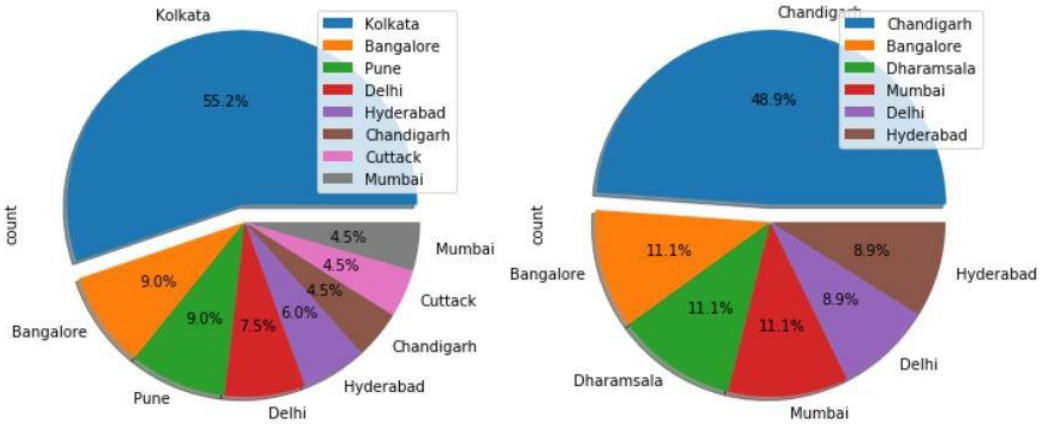
Analysis: We found the total number of teams with the help of function *unique()*. We then calculated the number of matches played as team1 and team2 by the respective Teams using

crosstab(). We then found the total matches won by the teams and grouped it accordingly.



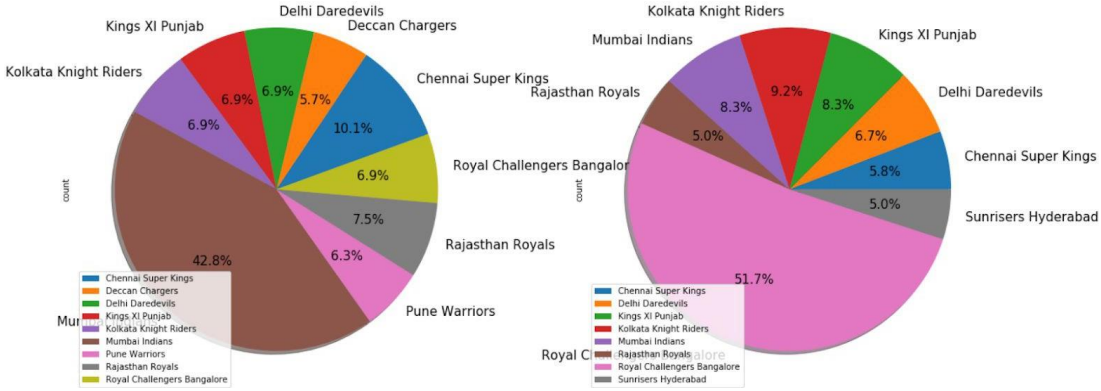
Horizontal Stacked Bar chart for seasons played by each IPL Team (Year v/s Time)

Analysis: We then sorted the teams according to the average matches played and won throughout the 11 seasons. And plotted a Stacked Bar chart for the same against the years played.



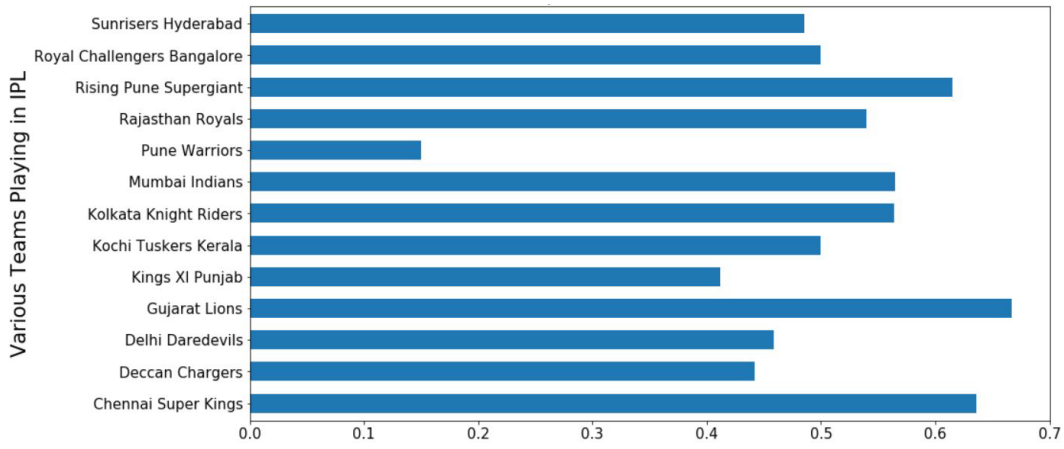
3D Pie Chart for Matches won by the teams I. Kolkata Knight Riders and II.Kings XI Punjab in different cities.

Analysis: We took a separate data array to store all the cities in which the matches were played. Removed any redundant values, sorted it accordingly to the number of matches played in the respective cities. We then plotted the cities in which each Team won in the form of a pie chart. shows the same for Teams KKR and KXIP.



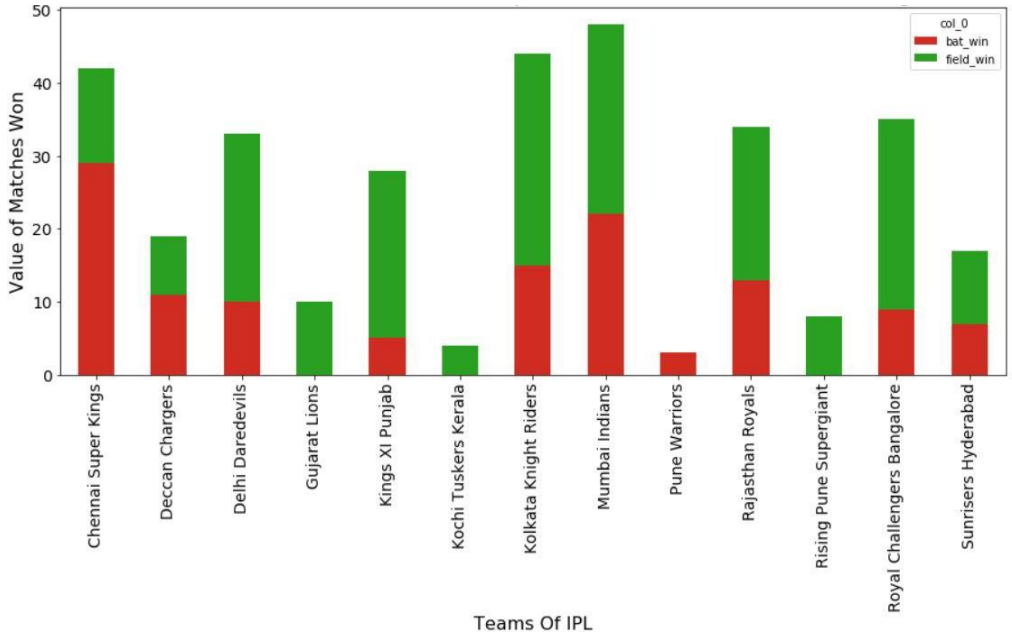
Cities Mumbai and Bangalore hosting matches of different Teams.

Analysis: We followed the same for 6 cities viz, Mumbai, Bangalore, Chennai, Delhi, Kolkata and Chandigarh. Wherein we calculated the total matches played by the Teams in the respective Cities. This figure depicts the Pie Chart for Cities Mumbai and Bangalore.



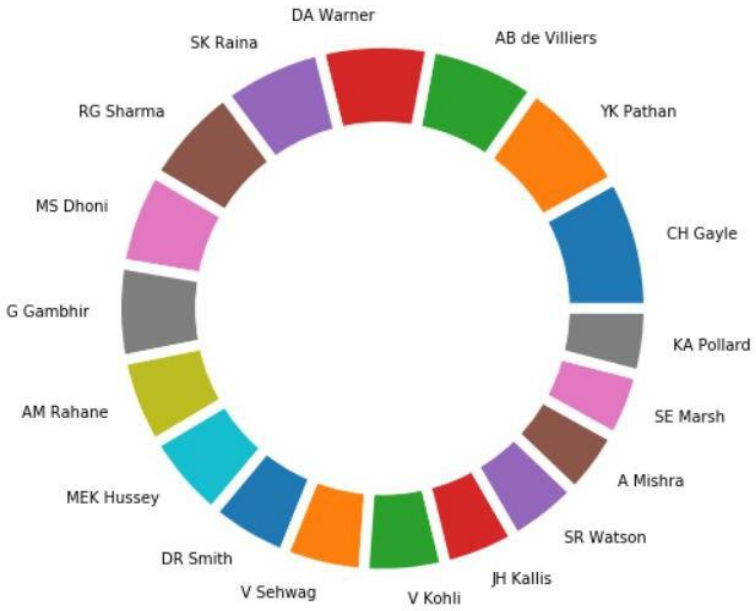
Horizontal Bar chart for Percentage win of the team when it won the Toss

Analysis:, we found out the total wins of each team against their toss wins and plotted the same above.



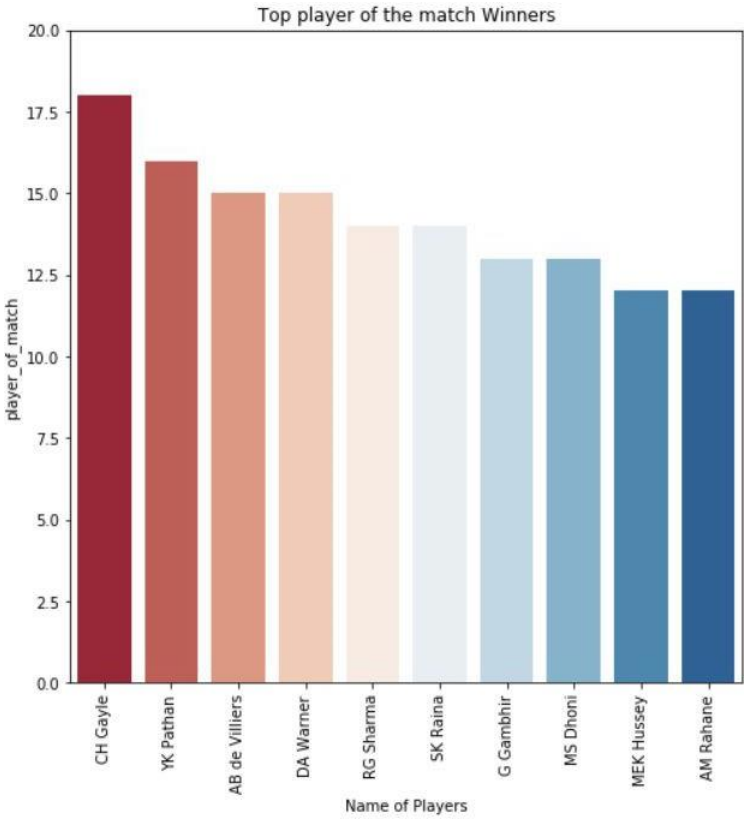
Stacked Bar Chart for the matches won wrt preference to field or bat after winning the Toss

Analysis: Here, we analysed the Toss Winning team's decision to field or bat against the Outcome of the Match.



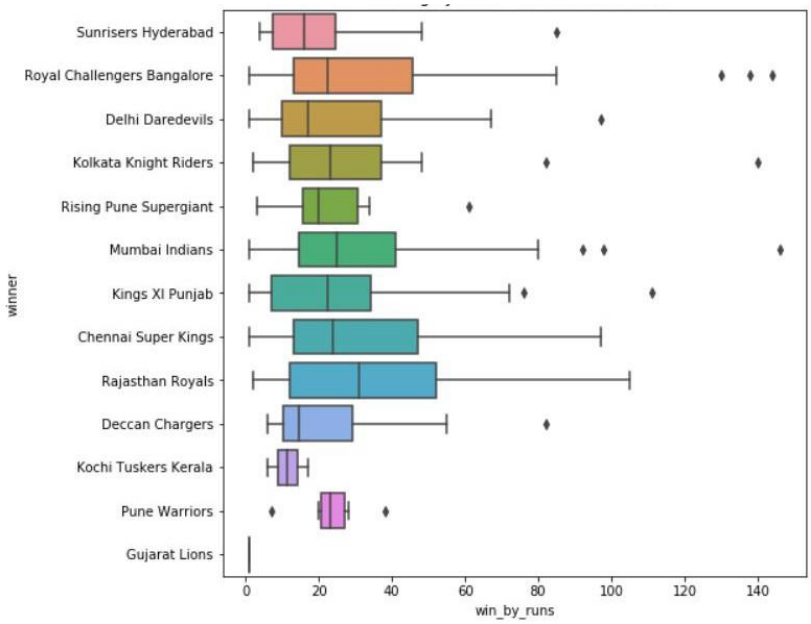
Doughnut Chart for Top 15 Cricketers who have been Man of the Matches

Analysis: In Fig, We found out the Top players from every Team based on the Title viz "Man of the Match" acquired by them.

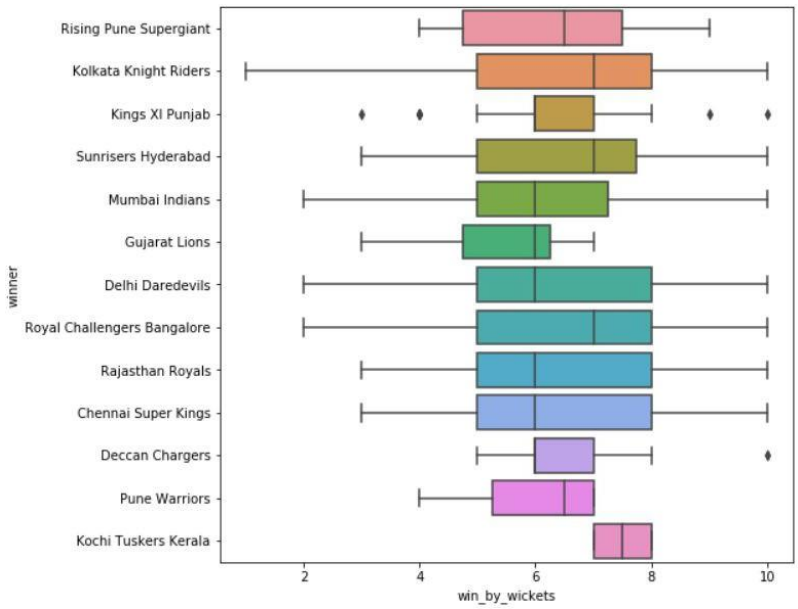


Top Players of the Match Winners

Analysis: Here, we have compared the Top players with the help of a Bar chart as shown in Fig. We can see that CH Gayle tops the list followed by YK Pathan.



Box Plot showing Team Performance - Winning by Runs



Box Plot showing Team Performance - Winning by Wickets

Analysis: In Fig , we have attempted to plot a performance chart with the help of their win rates as calculated earlier and the method through which they won.

Model

```
In [81]: RC = RandomForestClassifier(random_state=815)
```

```
Out[84]: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                                max_depth=None, max_features='auto', max_leaf_nodes=None,
                                min_impurity_decrease=0.0, min_impurity_split=None,
                                min_samples_leaf=1, min_samples_split=2,
                                min_weight_fraction_leaf=0.0, n_estimators=10,
                                n_jobs=None, oob_score=False, random_state=815,
                                verbose=0, warm_start=False)
```

Fitting

```
In [84]: RC.fit(X=match[['Team1_pts','Team1_RR','Team2_pts','Team2_RR','toss_decision','toss_win']],y=match.win)
```

Creating a function for Prediction

```
In [85]: def Predicter(team1,team2,toss_winner,toss_decision):
             Team1_pts = points.Pts[team1]
             Team1_RR = points['Net RR'][team1]
             Team2_pts = points.Pts[team2]
             Team2_RR = points['Net RR'][team2]
             pred = RC.predict([[Team1_pts,Team1_RR,Team2_pts,Team2_RR,toss_decision,toss_winner]])
             if pred == 'team 1':
                 return team1
             else:
                 return team2
```

Result Prediction

```
In [87]: Predicter('Sunrisers Hyderabad','Rajasthan Royals',toss_decision=1,toss_winner=2)
Out[87]: 'Rajasthan Royals'
```

and Principles and Practice of Knowledge Discovery in Databases (ECMLPKDD), 2016.

[4] Rabindra Lamsal and Ayesha Choudhary, “Predicting outcome of Indian Premier League(IPL) Matches Using Machine Learning”, Jawaharlal Nehru University, Sept 2020.

[5] Sricharan Shah, et al., “A Study on Performance of Cricket Players using Factor Analysis Approach,” International Journal of Advanced Research in Computer Science, vol. 8, no. 3, 2017.

[6] Van Staden, P. J., “Comparison of cricketers’ bowling and batting performances using graphical displays,” Current Science, vol. 96, no. 6, 2009.

Conclusion

The dataset contained all the match data since the beginning of Indian Premier League till 2019. We used a trained model to predict the outcome of any IPL match, immediately after the toss. We have successfully been able to predict the winner of an IPL match using Python Libraries and Machine Learning Model. Even though our prediction might not always be accurate, it gives a basic idea about the strategies and methodologies applied into designing a solution as such.

Future Works

- Since the dawn of the IPL in 2008, it has attracted viewers all around the globe. High level of uncertainty and last moment nail biters has urged fans to watch the matches. Within a short period, IPL has become the highest revenue generating league of cricket. Data Analytics has been a part of sports entertainment for a long time.
- In a cricket match, we might have seen the scoreline showing the probability of the team winning based on the current match situation. This is and will continue being Data Analytics in action.
- The predictions for a match have taken a step further in today's world. Thus, the need of experts to predict the outcome of games and performances of individual players has come to the fore.
- 'Fantasy cricket' is the new feature for the fans that has developed in the recent years and has been engaging a lot of people since its inception. Websites and mobile applications like Dream11, iplfantasy.com and many more have created a user interface for such predictions and people have been earning money through it.

REFERENCES

[1] Hossain, M. J., Kashem, M. A., Islam, M. S., & E-Jannat, M. (2018). Bangladesh Cricket Squad Prediction Using Statistical Data and Genetic Algorithm. 2018 4th International Conference on Electrical Engineering and Information & Communication Technology (ICEEiCT), Electrical Engineering and Information & Communication Technology (ICEEiCT), 2018 4th International Conference On, 178–181. https://doi.org/10.1109/CEEICT.2018.8628076

[2] Wang, J., Qing, G., Ou, X., Hu, W., Hu, C., & Zhang, Z. (2019). The impact of target dosimetry on patients' locoregional recurrence in nasopharyngeal carcinoma: A propensity score-matched analysis. Radiotherapy and Oncology, 141, 67–71. https://doi.org/10.1016/j.radonc.2019.09.001

[3] Hatharasinghe, M. M., & Poravi, G. (2019). Data Mining and Machine Learning in Cricket Match Outcome Prediction: Missing Links. 2019 IEEE 5th International Conference for Convergence in Technology (I2CT), Convergence in Technology (I2CT), 2019 IEEE 5th International Conference For, 1–4. https://doi.org/10.1109/I2CT45611.2019.9033698

[4] Anik, A. I., Yeaser, S., Hossain, A. G. M. I., & Chakrabarty, A. (2018). Player's Performance Prediction in ODI Cricket Using Machine Learning Algorithms. 2018 4th International Conference on Electrical Engineering and Information & Communication Technology (ICEEiCT), Electrical Engineering and Information & Communication Technology (ICEEiCT), 2018 4th International Conference On, 500–505. https://doi.org/10.1109/CEEICT.2018.8628118Chirag Goyal, "IPL-2017 Cross Country Cluster Analysis," International Journal of Computer Science Trends and Technology (IJCST), vol. 5, no. 4, 2017.

[5] The Cricket Winner Prediction With Application Of Machine Learning And

Data Analytics Daniel Mago Vistro, Faizan Rasheed, Leo Gertrude David

[6] Cricket Match Analytics Using the Big Data Approach
Mazhar Javed Awan 1,* , Syed Arbaz Haider Gilani 1 , Hamza Ramzan 1 , Haitham Nobanee 2,3,4,* , Awais Yasin 5 , Azlan Mohd Zain 6 and Rabia Javed

[7] Sport analytics for cricket game results using machine learning: An experimental study
Sinha, A. Application of Machine Learning in Cricket and Predictive Analytics of IPL 2020. Preprints 2020, 2020100436 (doi: 10.20944/preprints202010.0436.v1).

[8] Chirag Goyal, “IPL-2017 Cross Country Cluster Analysis,” International Journal of Computer Science Trends and Technology (IJCST), vol. 5, no. 4, 2017.
[9] Kalapdrum Passi and Niravkumar Pandey, “Predicting players performance in one day international cricket matches using Machine learning,” Computer Science & Information Technology (CS & IT), 2017.
[10] M. G. Jhanwar and V. Pudi, "Predicting the Outcome of ODI Cricket Matches: A Team Composition Based Approach," European Conference on Machine Learning
and Principles and Practice of Knowledge Discovery in Databases (ECMLPKDD), 2016.
[11] Rabindra Lamsal and Ayesha Choudhary, “Predicting outcome of Indian Premier League(IPL) Matches Using Machine Learning”, Jawaharlal Nehru University, Sept 2020.
[12] Sricharan Shah, et al., “A Study on Performance of Cricket Players using Factor Analysis Approach,” International Journal of Advanced Research

in Computer Science, vol. 8, no. 3, 2017.
[13] Van Staden, P. J., “Comparison of cricketers’ bowling and batting performances using graphical displays,” Current Science, vol. 96, no. 6, 2009.

ENCLOSED:

APPENDIX A(CODING)

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
from sklearn.ensemble import RandomForestClassifier,RandomForestRegressor
```

In [3]:
```
matches = pd.read_csv("matches.csv")
deliveries = pd.read_csv("deliveries.csv")
```

In [4]:
```
matches.head(3)
```

Out[4]:

	id	season	city	date	team1	team2	toss_winner	toss_decision	result	dl_app
0	1	2017	Hyderabad	2017-04-05	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	field	normal	
1	2	2017	Pune	2017-04-06	Mumbai Indians	Rising Pune Supergiant	Rising Pune Supergiant	field	normal	
2	3	2017	Rajkot	2017-04-07	Gujarat Lions	Kolkata Knight Riders	Kolkata Knight Riders	field	normal	

In [5]:
```
deliveries.head()
```

Out[5]:

	match_id	inning	batting_team	bowling_team	over	ball	batsman	non_striker	bowler	is_supe
0	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	1	DA Warner	S Dhawan	TS Mills	
1	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	2	DA Warner	S Dhawan	TS Mills	
2	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	3	DA Warner	S Dhawan	TS Mills	
3	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	4	DA Warner	S Dhawan	TS Mills	
4	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	5	DA Warner	S Dhawan	TS Mills	

5 rows × 21 columns

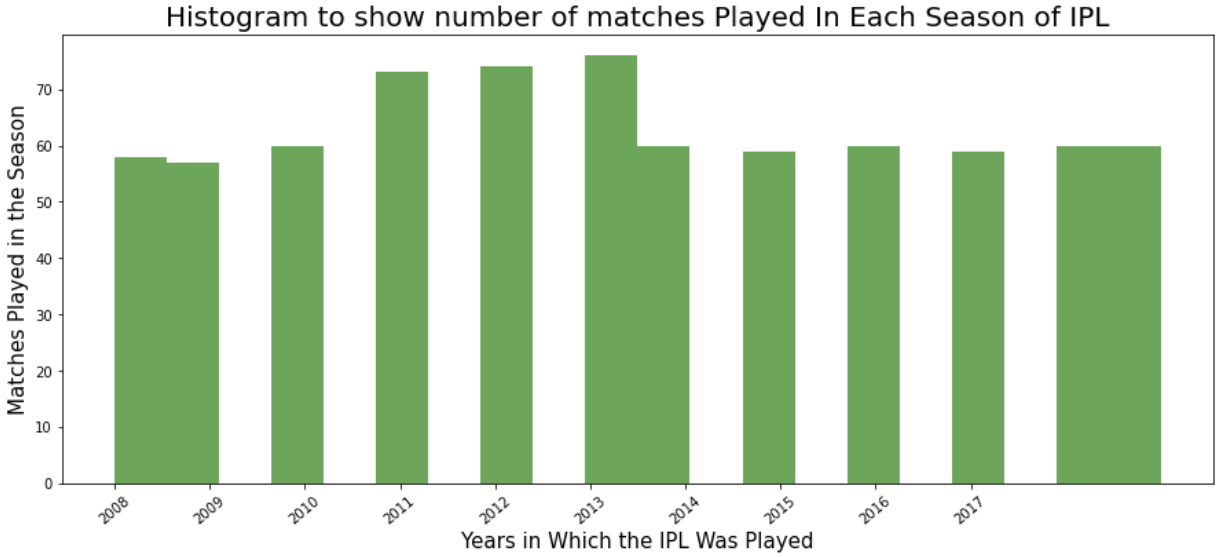
In [6]:
```
#Finding number of seasons in total
```

```
season = np.sort(matches['season'].unique())
print("The Number of Seasons in all are",len(season), "which are" , season)
```

```
The Number of Seasons in all are 12 which are [2008 2009 2010 2011 2012 2013 2014 20
15 2016 2017 2018 2019]
```

In [7]:

```
# Plotting a Histogram to get the number of seasons and no. of matches played in eac

plt.rcParams["figure.figsize"][0] = 15
plt.rcParams["figure.figsize"][1] = 6
matches['season'].hist(color = '#6DA65A',bins = 20,  grid = False)
plt.title("Histogram to show number of matches Played In Each Season of IPL",fontsiz
plt.xlabel("Years in Which the IPL Was Played", fontsize = 15)
plt.ylabel("Matches Played in the Season", fontsize = 15)
plt.xticks(np.arange(2008,2018, step = 1), rotation=40)
plt.show()
```



In [8]:

```
#Finding average number of matches played per season

i = 0;
match_no = []
while(i < len(season)):
    match_no.append(len(matches[matches['season'] == season[i]]['id']))
    i = i + 1;
mean = np.mean(match_no)
print("The Average Number of Matches played in all seasons combined", mean)
```

```
The Average Number of Matches played in all seasons combined 63.0
```

In [9]:

```
#finding the number of teams in all seasons of IPL

team1 = matches['team1'].unique()
matches = matches.replace("Rising Pune Supergiants", "Rising Pune Supergiant")
print("The total teams that participated in all seasons of IPLs are", len(team1))
matches['team1'].unique()
```

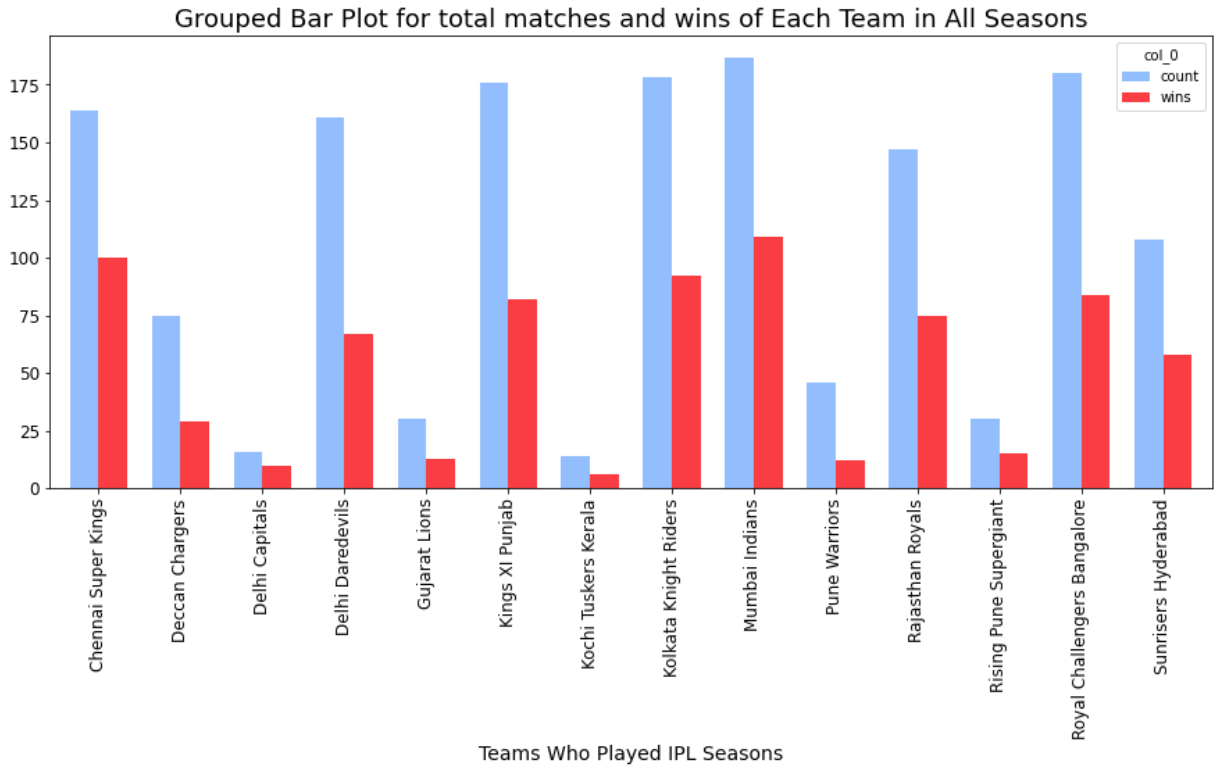
```
The total teams that participated in all seasons of IPLs are 15
```

Out[9]:

```
array(['Sunrisers Hyderabad', 'Mumbai Indians', 'Gujarat Lions',
       'Rising Pune Supergiant', 'Royal Challengers Bangalore',
       'Kolkata Knight Riders', 'Delhi Daredevils', 'Kings XI Punjab',
       'Chennai Super Kings', 'Rajasthan Royals', 'Deccan Chargers',
       'Kochi Tuskers Kerala', 'Pune Warriors', 'Delhi Capitals'],
      dtype=object)
```

In [10]:
```
# Finding the number of matches played by each team.

#Finding the unique teams in the IPL Matches
team1 = matches['team1'].unique()

# Finding the number of matches played as team1 and team2 by the teams
team1_all = pd.crosstab(index = matches['team1'], columns = 'count')
team2_all = pd.crosstab(index = matches['team2'], columns = 'count')

# Adding both the numbers to get the total number of matches
final_team = team1_all.add(team2_all)

# Finding the total matches won by the team
winner = pd.crosstab(index = matches['winner'], columns = "count")
final_team['wins'] = winner['count']
final_team[['count', 'wins']].plot(kind = 'bar' ,  width = 0.7,   color=['#93BEFE' ,
plt.xlabel("Teams Who Played IPL Seasons", fontsize = 14)
plt.title("Grouped Bar Plot for total matches and wins of Each Team in All Seasons",
```

Out[10]:
```
Text(0.5, 1.0, 'Grouped Bar Plot for total matches and wins of Each Team in All Seas
ons')
```



In [11]:
```
#Calculating the percentage of the wins for all the teams
final_team['percent'] = final_team['wins']/final_team['count'] * 100
final_team.sort_values(by = 'percent', ascending = False)
```

Out[11]:

col_0	count	wins	percent
team1			
Delhi Capitals	16	10	62.500000
Chennai Super Kings	164	100	60.975610
Mumbai Indians	187	109	58.288770
Sunrisers Hyderabad	108	58	53.703704

col_0	count	wins	percent
team1			
Kolkata Knight Riders	178	92	51.685393
Rajasthan Royals	147	75	51.020408
Rising Pune Supergiant	30	15	50.000000
Royal Challengers Bangalore	180	84	46.666667
Kings XI Punjab	176	82	46.590909
Gujarat Lions	30	13	43.333333
Kochi Tuskers Kerala	14	6	42.857143
Delhi Daredevils	161	67	41.614907
Deccan Chargers	75	29	38.666667
Pune Warriors	46	12	26.086957

In [12]:
```
#Calculating the average number of matches played by each team

mean = final_team['count'].mean()
print("The average no. of matches are", mean)
print("The teams who have played less than average matches are" ,sum(final_team['cou
```

```
The average no. of matches are 108.0
The teams who have played less than average matches are 6
```

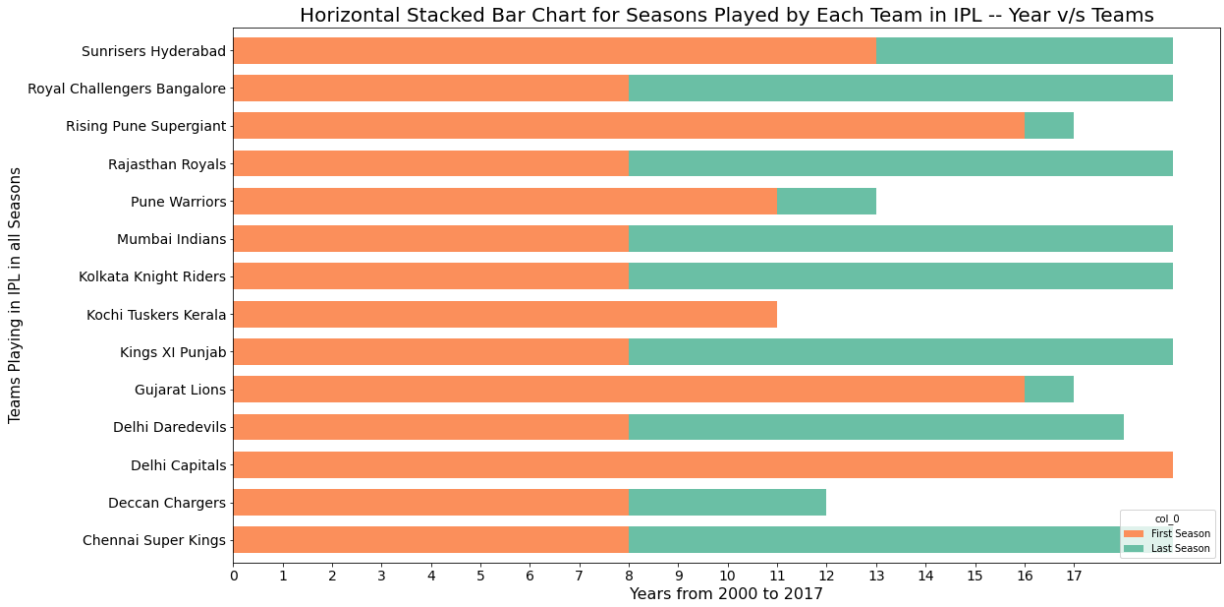
In [13]:
```
#Calculating average number of wins
mean_win = final_team['percent'].mean()
print("The average percent win is", mean_win)
print("The teams who have won more than average winning percentage are" ,sum(final_t
```

```
The average percent win is 48.14217620740895
The teams who have won more than average winning percentage are 7
```

In [14]:
```
# Determining the first and the last seasons of IPL
i = 0;
team_order = np.sort(matches['team1'].unique())
array = []
first = []
last = []
while i < 14:
    array = np.sort(list(matches[matches['team1'] == team_order[i]]['season'].unique
    first.append(array[0] % 2000)
    last.append(array[len(array)-1] - array[0])
    i = i +1
final_team['First Season']= first
final_team['Last Season'] = last
final_team['seasons'] = [x+1 for x in last]
plt.rcParams["figure.figsize"][0] = 18
plt.rcParams["figure.figsize"][1] = 10
final_team[['First Season' , 'Last Season']].plot(kind = 'barh', stacked = True, rot
                                                  fontsize = 14,width = 0.7,   color
plt.title('Horizontal Stacked Bar Chart for Seasons Played by Each Team in IPL -- Ye
plt.xlabel("Years from 2000 to 2017", fontsize = 16)
plt.ylabel("Teams Playing in IPL in all Seasons", fontsize  = 15)
```

```
Text(0, 0.5, 'Teams Playing in IPL in all Seasons')
```

Out[14]:



In [15]:

```
final_team = final_team.drop(columns = ['First Season' , 'Last Season'])
final_team.sort_values(['seasons', 'percent'], ascending = [False, False])
```

Out[15]:

col_0	count	wins	percent	seasons
team1				
Chennai Super Kings	164	100	60.975610	12
Mumbai Indians	187	109	58.288770	12
Kolkata Knight Riders	178	92	51.685393	12
Rajasthan Royals	147	75	51.020408	12
Royal Challengers Bangalore	180	84	46.666667	12
Kings XI Punjab	176	82	46.590909	12
Delhi Daredevils	161	67	41.614907	11
Sunrisers Hyderabad	108	58	53.703704	7
Deccan Chargers	75	29	38.666667	5
Pune Warriors	46	12	26.086957	3
Rising Pune Supergiant	30	15	50.000000	2
Gujarat Lions	30	13	43.333333	2
Delhi Capitals	16	10	62.500000	1
Kochi Tuskers Kerala	14	6	42.857143	1

In [16]:

```
#Finding the Cities in which matches were played
null = matches.isnull()
print("The entries which have no cities mentioned are" , sum(null['city']))
print(matches[null['city'] == True]['id'])
# Removing all these matches by these specific indices
matches_clean = matches.drop(matches.index[[461,462,466,468,469,474,476]])
```

```
The entries which have no cities mentioned are 7
461    462
```

```
462    463
466    467
468    469
469    470
474    475
476    477
Name: id, dtype: int64
```

In [17]:

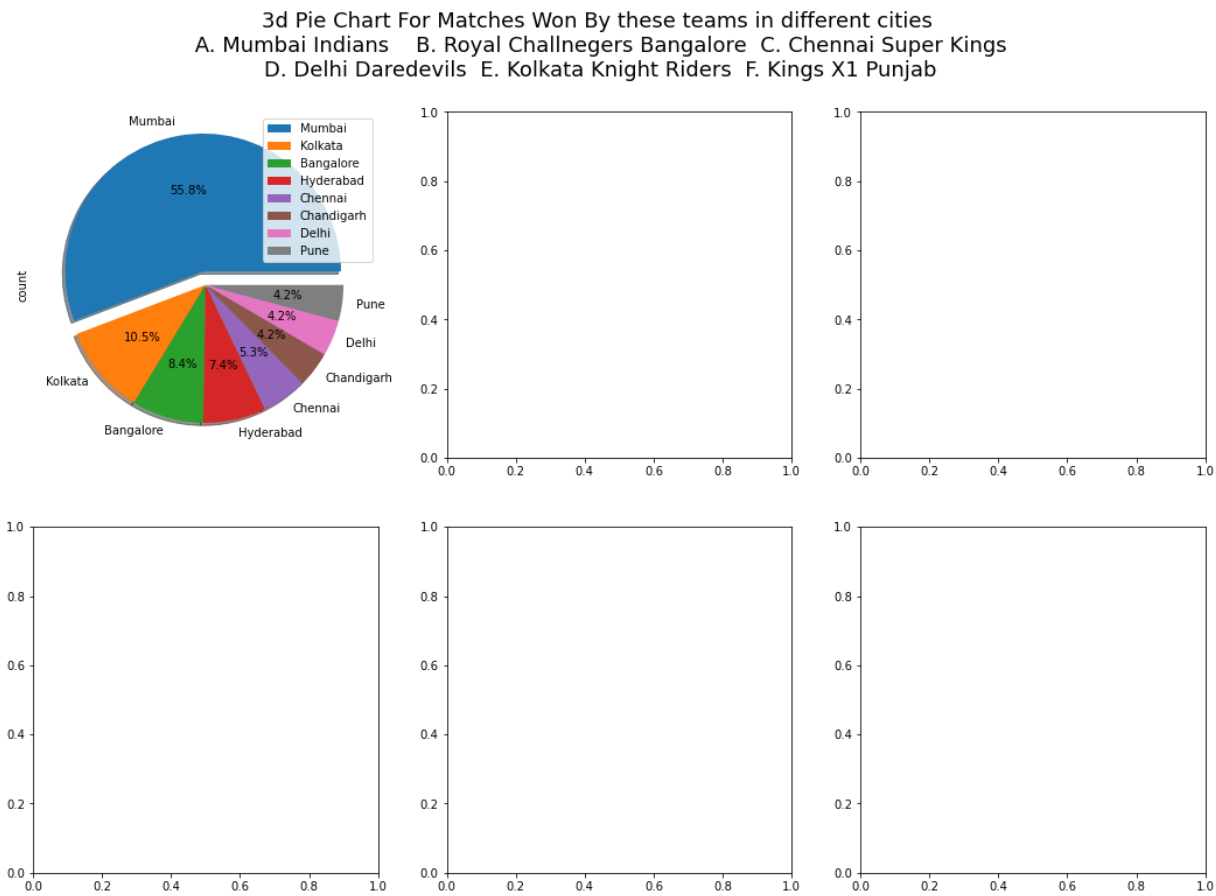
```
final_team_city = final_team[final_team['count'] > 70]
final_team_city = final_team_city.sort_values(by = 'count', ascending = False)
cities = pd.crosstab(index = matches_clean['city'], columns = 'count').sort_values(b
cities_analyse = cities[cities['count'] > 40]
```

In [18]:

```
#Finding the cities in Which MUMBAI INDIANS WON for PIE CHART 1
matcheswin = matches_clean[matches_clean['winner'] == "Mumbai Indians"]
winner_MI = pd.crosstab(index = matcheswin['city'],columns= 'count' )
winner_MI = winner_MI[winner_MI['count'] > 2].sort_values(by = 'count' , ascending =

#Finding the cities in Which RCB WON -- PIE CHART 1
matcheswin2 = matches_clean[matches_clean['winner'] == "Royal Challengers Bangalore"
winner_RCB = pd.crosstab(index = matcheswin2['city'],columns= 'count' )
winner_RCB = winner_RCB[winner_RCB['count'] > 3].sort_values(by = 'count' , ascendin

#Finding the cities in Which CSK WON -- PIE CHART 1
matcheswin3 = matches_clean[matches_clean['winner'] == "Chennai Super Kings"]
winner_CSK = pd.crosstab(index = matcheswin3['city'],columns= 'count' )
winner_CSK = winner_CSK[winner_CSK['count'] > 2].sort_values(by = 'count' , ascendin

#Finding the cities in Which DD WON -- PIE CHART 1
matcheswin4 = matches_clean[matches_clean['winner'] == "Delhi Daredevils"]
winner_DD = pd.crosstab(index = matcheswin4['city'],columns= 'count' )
winner_DD = winner_DD[winner_DD['count'] > 2].sort_values(by = 'count' , ascending =

#Finding the cities in Which KKR WON -- PIE CHART 1
matcheswin5 = matches_clean[matches_clean['winner'] == "Kolkata Knight Riders"]
winner_KKR = pd.crosstab(index = matcheswin5['city'],columns= 'count' )
winner_KKR = winner_KKR[winner_KKR['count'] > 2].sort_values(by = 'count' , ascendin

#Finding the cities in Which KIP WON -- PIE CHART 1
matcheswin6 = matches_clean[matches_clean['winner'] == "Kings XI Punjab"]
winner_KIP = pd.crosstab(index = matcheswin6['city'],columns= 'count' )
winner_KIP = winner_KIP[winner_KIP['count'] > 3].sort_values(by = 'count' , ascendin
```

In [19]:

```
plt.rcParams["figure.figsize"][0] = 18
plt.rcParams["figure.figsize"][1] = 12
fig, axs = plt.subplots(2,3)
plt.suptitle("3d Pie Chart For Matches Won By these teams in different cities \n "
             "A. Mumbai Indians    B. Royal Challnegers Bangalore  C. Chennai Super
             "D. Delhi Daredevils  E. Kolkata Knight Riders  F. Kings X1 Punjab", fo
explode1 = (0.1, 0, 0 , 0 , 0 ,0 , 0 , 0)
explode2 = (0.1,0,0,0)
explode3 = (0.1, 0 , 0 , 0 ,0 , 0 , 0 )
explode4 = (0.1, 0,0,0,0,0)
winner_MI.plot(kind = 'pie' , subplots = True, ax=axs[0][0], explode = explode1,  au

plt.show()
```



In [20]:

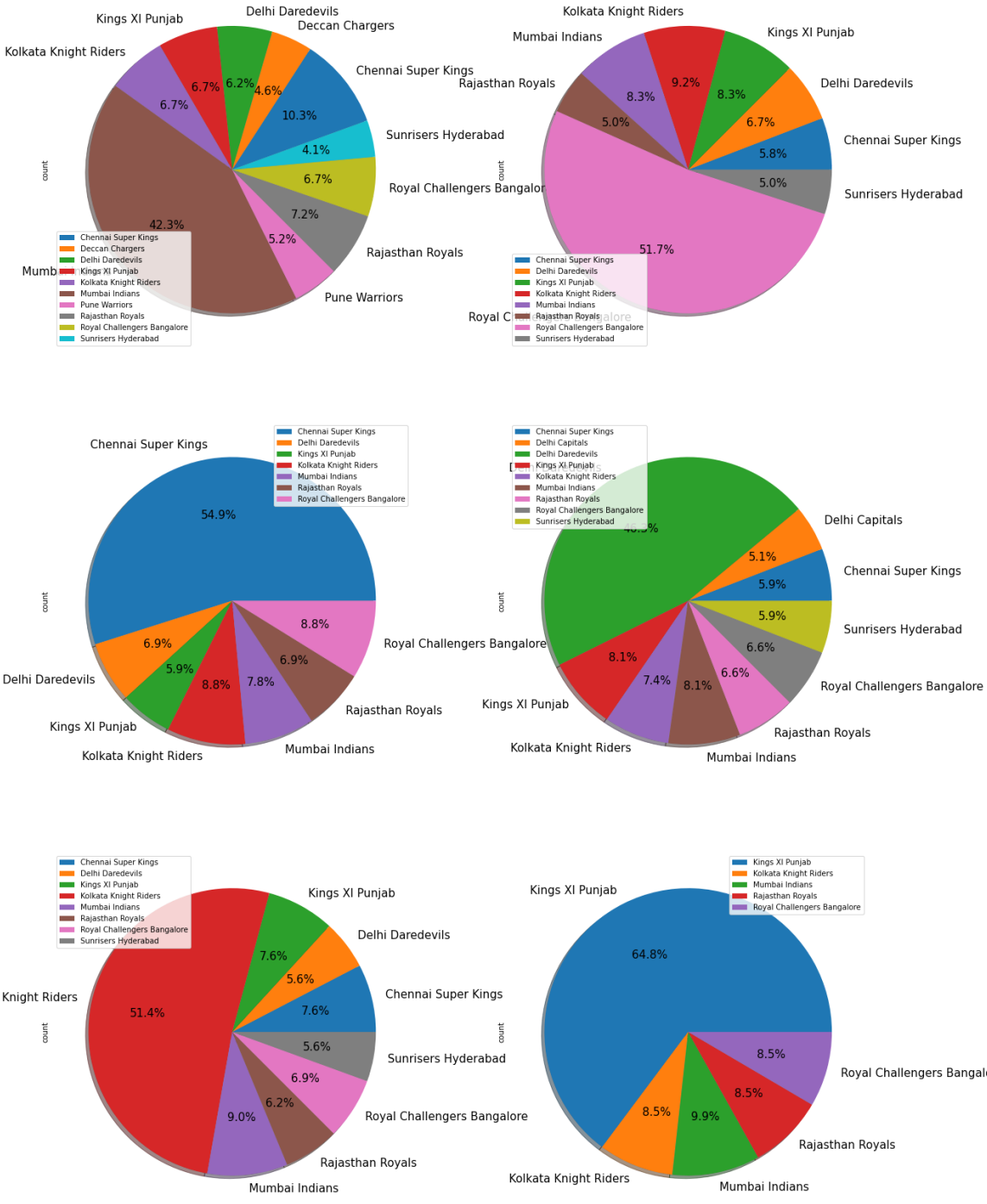
```
# Finding the matches played in Mumbai City FOR PIE CHART 2
matches_city1 = matches[matches['city'] == "Mumbai"]
# Finding the number of matches played as team1 and team2 by the teams
team1_mumbai = pd.crosstab(index = matches_city1['team1'], columns = 'count')
team2_mumbai = pd.crosstab(index = matches_city1['team2'], columns = 'count')
# Adding both the numbers to get the total number of matches
team_city1 = team1_mumbai.add(team2_mumbai)
team_city1 = team_city1[team_city1['count'] > 5]

# Finding the matches played in Bangalore City -- PIE CHART 2
matches_city2 = matches[matches['city'] == "Bangalore"]
# Finding the number of matches played as team1 and team2 by the teams
team1_bangalore = pd.crosstab(index = matches_city2['team1'], columns = 'count')
team2_bangalore = pd.crosstab(index = matches_city2['team2'], columns = 'count')
# Adding both the numbers to get the total number of matches
team_city2 = team1_bangalore.add(team2_bangalore)
team_city2 = team_city2[team_city2['count'] > 5]

# Finding the matches played in Chennai City -- PIE CHART 2
matches_city3 = matches[matches['city'] == "Chennai"]
# Finding the number of matches played as team1 and team2 by the teams
team1_chennai = pd.crosstab(index = matches_city3['team1'], columns = 'count')
team2_chennai = pd.crosstab(index = matches_city3['team2'], columns = 'count')
# Adding both the numbers to get the total number of matches
team_city3 = team1_chennai.add(team2_chennai)
team_city3 = team_city3[team_city3['count'] > 5]

# Finding the matches played in Delhi City -- PIE CHART 2
matches_city4 = matches[matches['city'] == "Delhi"]
# Finding the number of matches played as team1 and team2 by the teams
team1_delhi = pd.crosstab(index = matches_city4['team1'], columns = 'count')
team2_delhi = pd.crosstab(index = matches_city4['team2'], columns = 'count')
# Adding both the numbers to get the total number of matches
team_city4 = team1_delhi.add(team2_delhi)
team_city4 = team_city4[team_city4['count'] > 5]
```

```
# Finding the matches played in Kolkata City -- PIE CHART 2
matches_city5 = matches[matches['city'] == "Kolkata"]
# Finding the number of matches played as team1 and team2 by the teams
team1_kolkata = pd.crosstab(index = matches_city5['team1'], columns = 'count')
team2_kolkata = pd.crosstab(index = matches_city5['team2'], columns = 'count')
# Adding both the numbers to get the total number of matches
team_city5 = team1_kolkata.add(team2_kolkata)
team_city5 = team_city5[team_city5['count'] > 5]

## Finding the matches played in Chandigarh City -- PIE CHART 2
matches_city6 = matches[matches['city'] == "Chandigarh"]
# Finding the number of matches played as team1 and team2 by the teams
team1_chandigarh = pd.crosstab(index = matches_city6['team1'], columns = 'count')
team2_chandigarh = pd.crosstab(index = matches_city6['team2'], columns = 'count')
# Adding both the numbers to get the total number of matches
team_city6 = team1_chandigarh.add(team2_chandigarh)
team_city6 = team_city6[team_city6['count'] > 5]
```

In [21]:

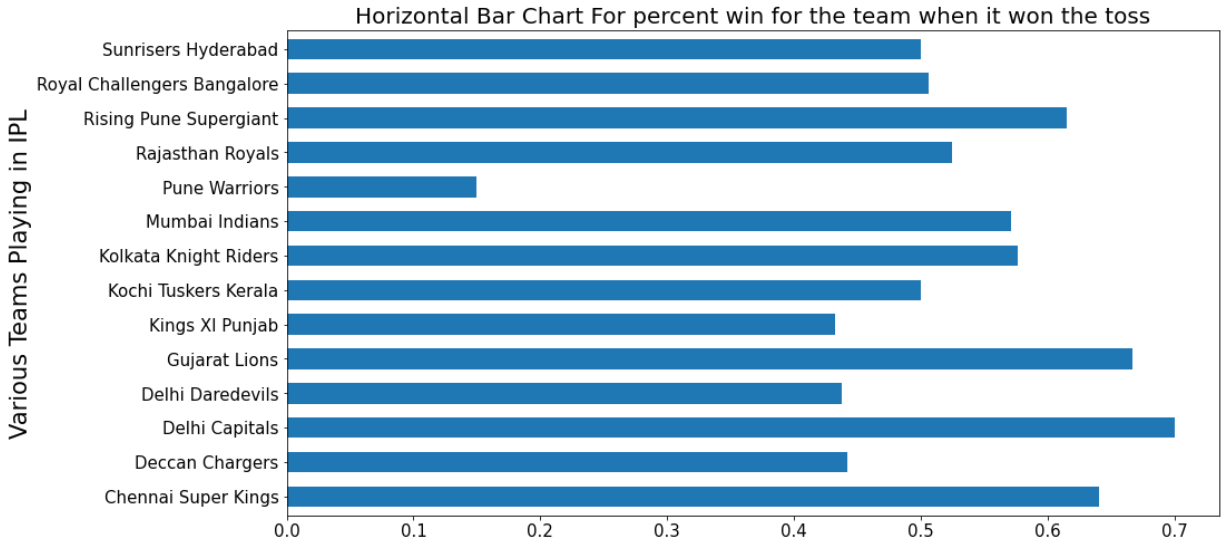
```
plt.rcParams["figure.figsize"][0] = 20
plt.rcParams["figure.figsize"][1] = 30
fig, axs = plt.subplots(3,2)
plt.suptitle("Cities Hosting Matches of Different Teams  \n "
             "A. Mumbai    B. Bangalore  C. Chennai \n"
             "D. Delhi     E. Kolkata    F.Chandigarh", fontsize = 24)
team_city1.plot(kind = 'pie' , subplots = True, ax = axs[0][0],  autopct = "%1.1f%%"
team_city2.plot(kind = 'pie' , subplots = True, ax = axs[0][1],  autopct = "%1.1f%%"
team_city3.plot(kind = 'pie' , subplots = True, ax = axs[1][0],  autopct = "%1.1f%%"
team_city4.plot(kind = 'pie' , subplots = True, ax = axs[1][1],  autopct = "%1.1f%%"
team_city5.plot(kind = 'pie' , subplots = True, ax = axs[2][0],  autopct = "%1.1f%%"
team_city6.plot(kind = 'pie' , subplots = True, ax = axs[2][1],  autopct = "%1.1f%%"
plt.show()
```

Cities Hosting Matches of Different Teams
A. Mumbai B. Bangalore C. Chennai
D. Delhi E. Kolkata F.Chandigarh



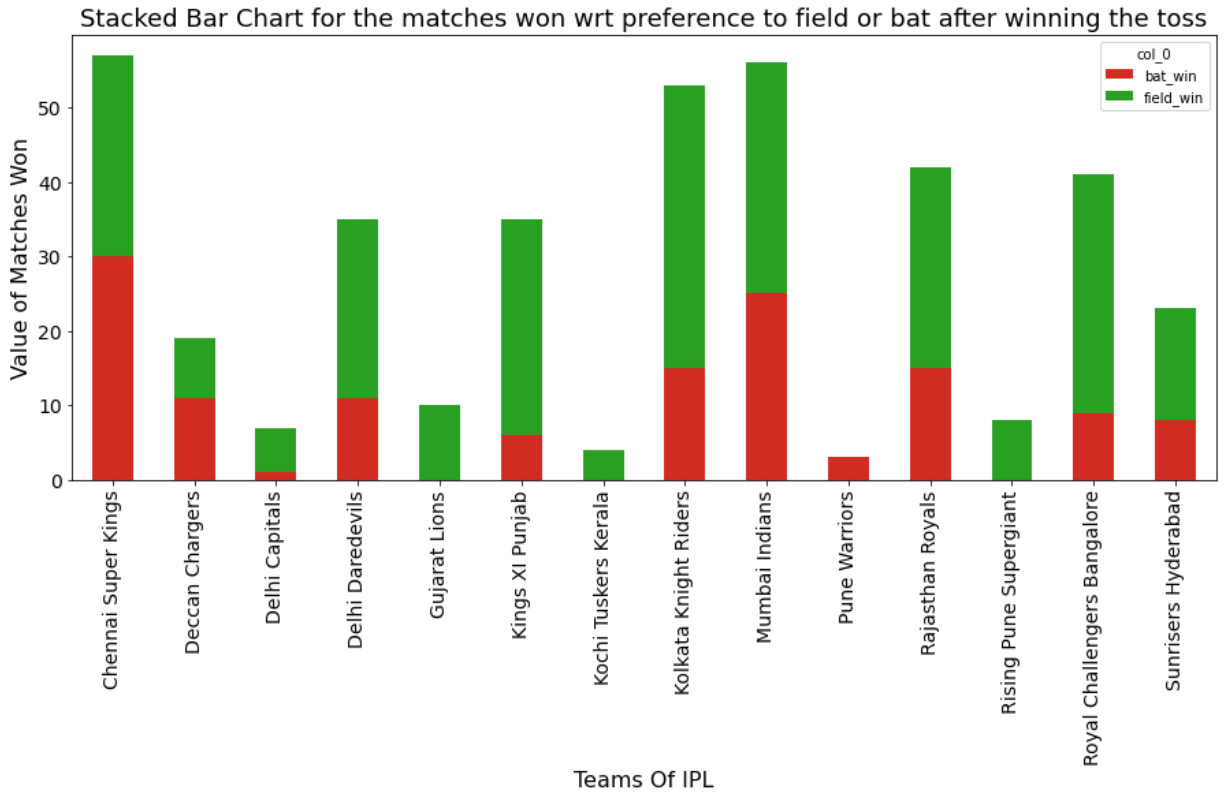
In [22]:

```
win_game = matches[matches['toss_winner'] == matches['winner']]
toss_win = pd.crosstab(index = win_game['winner'], columns = 'count')
toss_all = pd.crosstab(index = matches['toss_winner'], columns = 'count')
toss_all['won_match'] = toss_win['count']
toss_all['percent'] = toss_win['count'] / toss_all['count']
plt.rcParams["figure.figsize"][0] = 15
plt.rcParams["figure.figsize"][1] = 8
toss_all['percent'].plot(kind = 'barh', width = 0.6, fontsize = 15)
plt.ylabel("Various Teams Playing in IPL", fontsize =21)
plt.title("Horizontal Bar Chart For percent win for the team when it won the toss" ,
plt.show()
```



In [23]:

```
win_game.head(4)
field_win = win_game[win_game['toss_decision'] == 'field']
bat_win = win_game[win_game['toss_decision'] == 'bat']
bat = pd.crosstab(index = bat_win['winner'] , columns = 'count')
field = pd.crosstab(index = field_win['winner'], columns = 'count')
toss_all['bat_win'] = bat['count']
toss_all['field_win'] = field['count']
plt.rcParams["figure.figsize"][0] = 15
plt.rcParams["figure.figsize"][1] = 6
toss_all[['bat_win', 'field_win']].plot(kind = 'bar', stacked = True, fontsize = 14,
plt.title("Stacked Bar Chart for the matches won wrt preference to field or bat afte
plt.ylabel("Value of Matches Won", fontsize = 16)
plt.xlabel("Teams Of IPL", fontsize = 16)
plt.show()
```



In [24]:

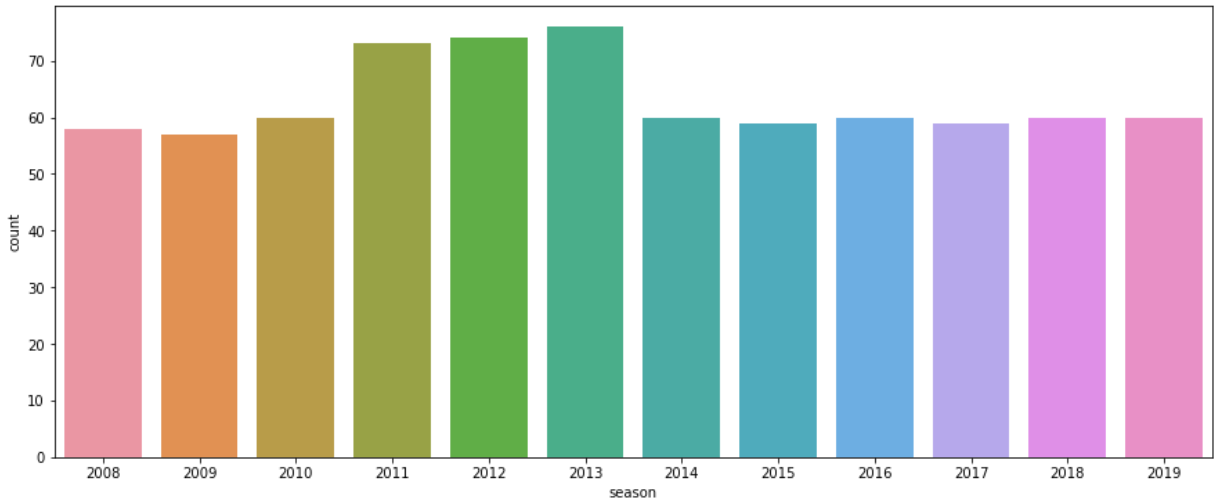
```
matches.describe()
```

Out[24]:

	id	season	dl_applied	win_by_runs	win_by_wickets

	id	season	dl_applied	win_by_runs	win_by_wickets
count	756.000000	756.000000	756.000000	756.000000	756.000000
mean	1792.178571	2013.444444	0.025132	13.283069	3.350529
std	3464.478148	3.366895	0.156630	23.471144	3.387963
min	1.000000	2008.000000	0.000000	0.000000	0.000000
25%	189.750000	2011.000000	0.000000	0.000000	0.000000
50%	378.500000	2013.000000	0.000000	0.000000	4.000000
75%	567.250000	2016.000000	0.000000	19.000000	6.000000
max	11415.000000	2019.000000	1.000000	146.000000	10.000000

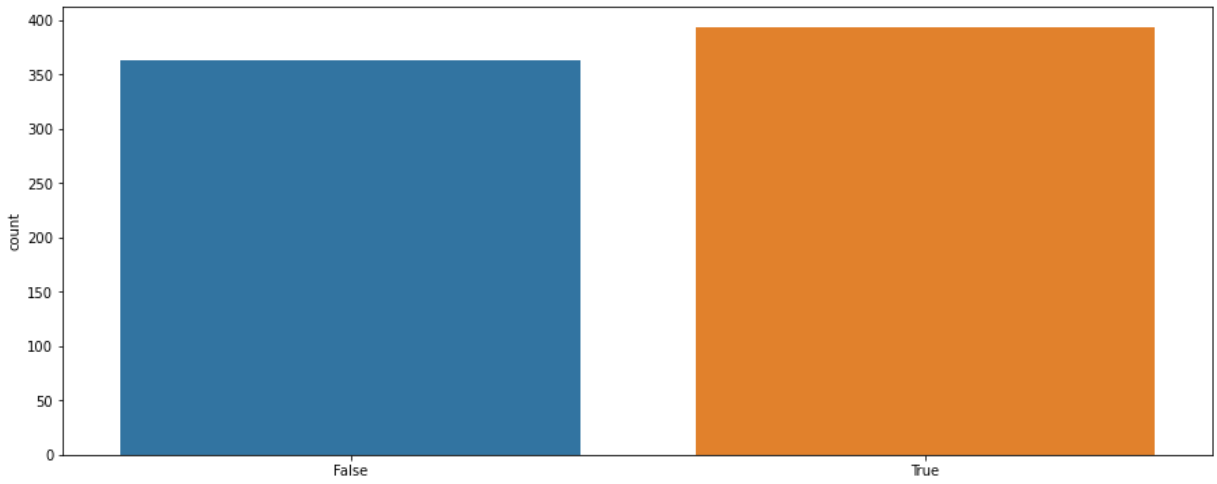
In [25]:
```
sb.countplot(x = 'season', data = matches)
plt.show()
```



In [26]:
```
winneroft = matches['toss_winner'] == matches['winner']
winneroft.groupby(winneroft).size()
sb.countplot(winneroft)
```

```
C:\SOFTWARES\ANACONDA\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pa
ss the following variable as a keyword arg: x. From version 0.12, the only valid pos
itional argument will be `data`, and passing other arguments without an explicit key
word will result in an error or misinterpretation.
  warnings.warn(
```

Out[26]: <AxesSubplot:ylabel='count'>

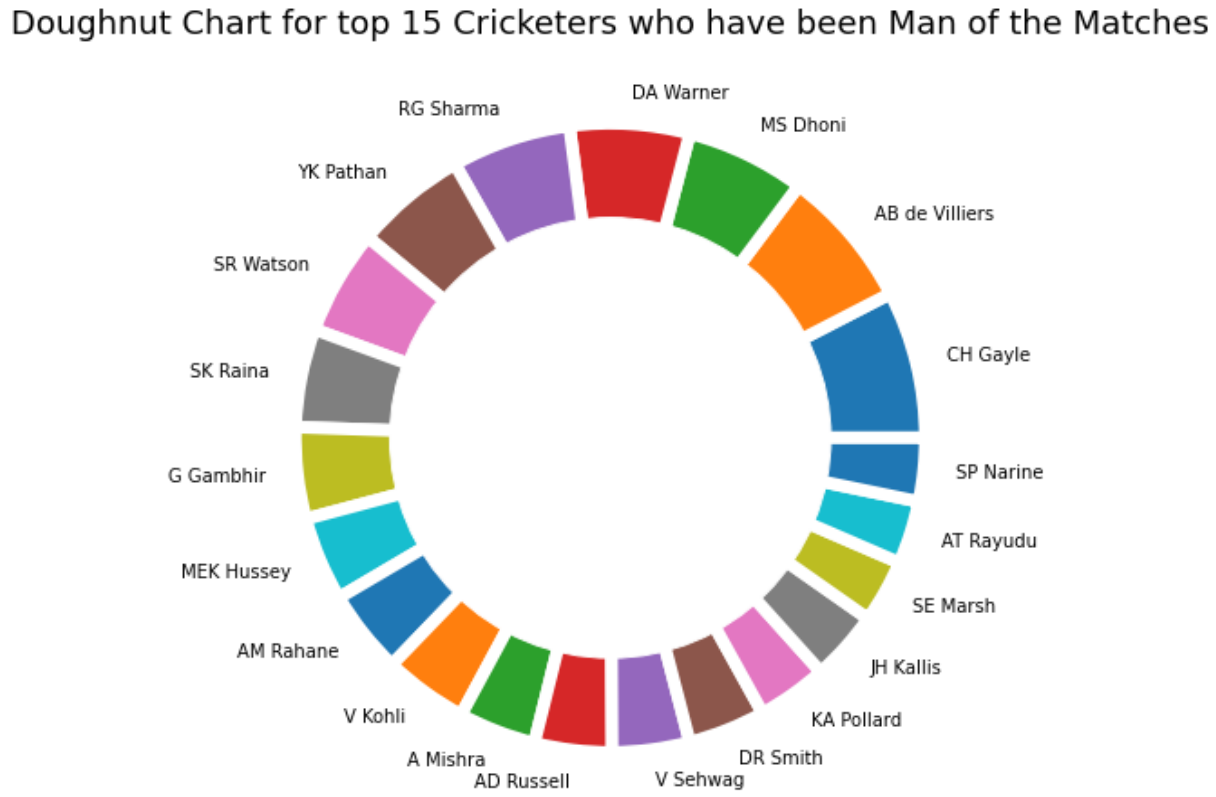


In [27]:

```
#Finding the count of each player being the man of the match and selecting top 15 to
player = matches['player_of_match']
from collections import Counter
count = Counter(player).values()
list1 = pd.DataFrame({'Player' :matches['player_of_match'].unique()})
list1['count'] = count
list1 = list1[list1['count'] > 8].sort_values(by = 'count', ascending = False)
#plotting the Doughnut Chart
plt.rcParams["figure.figsize"][0] = 8
plt.rcParams["figure.figsize"][1] = 8
plt.pie(list1['count'], labels = list1['Player'], wedgeprops = {'linewidth' : 7, 'ed
my_circle = plt.Circle(( 0,0) , 0.7 , color = 'White')
p = plt.gcf()
p.gca().add_artist(my_circle)
plt.title("Doughnut Chart for top 15 Cricketers who have been Man of the Matches", f
```

Out[27]:

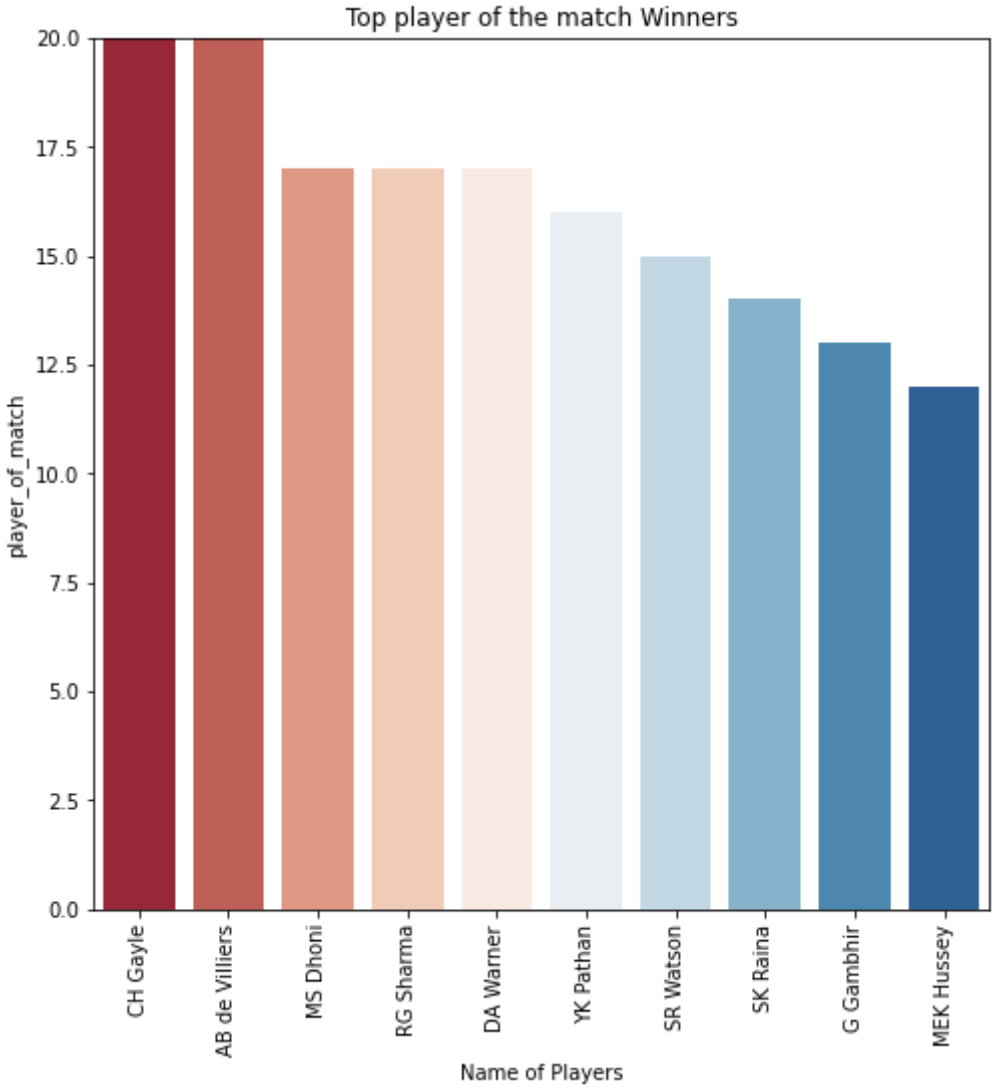
```
Text(0.5, 1.0, 'Doughnut Chart for top 15 Cricketers who have been Man of the Matche
s')
```



In [28]:

```
top_players = matches.player_of_match.value_counts()[:10]
```

```
#sns.barplot(x="day", y="total_bill", data=tips)
fig, ax = plt.subplots()
ax.set_ylim([0,20])
ax.set_ylabel("Number of Awards")
ax.set_xlabel("Name of Players")
ax.set_title("Top player of the match Winners")
#top_players.plot.bar()
sb.barplot(x = top_players.index, y = top_players, orient='v', palette="RdBu");
plt.xticks(rotation = 'vertical')
plt.show()
```



In [29]:

```
## BATSMEN DATA GROUPED BY MATCH
# Here the data is grouped to provide deeper depth of statistics and later for the t

batsman_grp = deliveries.groupby(["match_id", "inning", "batting_team", "batsman"])
batsmen = batsman_grp["batsman_runs"].sum().reset_index()

# Ignore the wide balls.
balls_faced = deliveries[deliveries["wide_runs"] == 0]
balls_faced = balls_faced.groupby(["match_id", "inning", "batsman"])["batsman_runs"]
balls_faced.columns = ["match_id", "inning", "batsman", "balls_faced"]
batsmen = batsmen.merge(balls_faced, left_on=["match_id", "inning", "batsman"],
                        right_on=["match_id", "inning", "batsman"], how="left")

fours = deliveries[ deliveries["batsman_runs"] == 4]
sixes = deliveries[ deliveries["batsman_runs"] == 6]

fours_per_batsman = fours.groupby(["match_id", "inning", "batsman"])["batsman_runs"]
```

```
sixes_per_batsman = sixes.groupby(["match_id", "inning", "batsman"])["batsman_runs"]

fours_per_batsman.columns = ["match_id", "inning", "batsman", "4s"]
sixes_per_batsman.columns = ["match_id", "inning", "batsman", "6s"]

batsmen = batsmen.merge(fours_per_batsman, left_on=["match_id", "inning", "batsman"]
                        right_on=["match_id", "inning", "batsman"], how="left")
batsmen = batsmen.merge(sixes_per_batsman, left_on=["match_id", "inning", "batsman"]
                        right_on=["match_id", "inning", "batsman"], how="left")
batsmen['SR'] = np.round(batsmen['batsman_runs'] / batsmen['balls_faced'] * 100, 2)

for col in ["batsman_runs", "4s", "6s", "balls_faced", "SR"]:
    batsmen[col] = batsmen[col].fillna(0)

dismissals = deliveries[ pd.notnull(deliveries["player_dismissed"])]
dismissals = dismissals[["match_id", "inning", "player_dismissed", "dismissal_kind",
dismissals.rename(columns={"player_dismissed": "batsman"}, inplace=True)
batsmen = batsmen.merge(dismissals, left_on=["match_id", "inning", "batsman"],
                        right_on=["match_id", "inning", "batsman"], how="left")

batsmen = matches[['id','season']].merge(batsmen, left_on = 'id', right_on = 'match_
batsmen.head(10)
```

Out[29]:

	season	match_id	inning	batting_team	batsman	batsman_runs	balls_faced	4s	6s	SR
0	2017	1	1	Sunrisers Hyderabad	BCJ Cutting	16	6.0	0.0	2.0	266.67
1	2017	1	1	Sunrisers Hyderabad	DA Warner	14	8.0	2.0	1.0	175.00
2	2017	1	1	Sunrisers Hyderabad	DJ Hooda	16	12.0	0.0	1.0	133.33
3	2017	1	1	Sunrisers Hyderabad	MC Henriques	52	37.0	3.0	2.0	140.54
4	2017	1	1	Sunrisers Hyderabad	S Dhawan	40	31.0	5.0	0.0	129.03
5	2017	1	1	Sunrisers Hyderabad	Yuvraj Singh	62	27.0	7.0	3.0	229.63
6	2017	1	2	Royal Challengers Bangalore	A Choudhary	6	2.0	0.0	1.0	300.00
7	2017	1	2	Royal Challengers Bangalore	CH Gayle	32	21.0	2.0	3.0	152.38
8	2017	1	2	Royal Challengers Bangalore	KM Jadhav	31	16.0	4.0	1.0	193.75
9	2017	1	2	Royal Challengers Bangalore	Mandeep Singh	24	16.0	5.0	0.0	150.00

In [30]:

```
# Data is grouped for bowlers to provide greater depth of information. Very importan

bowler_grp = deliveries.groupby(["match_id", "inning", "bowling_team", "bowler", "ov
bowlers = bowler_grp["total_runs", "wide_runs", "bye_runs", "legbye_runs", "noball_r
```

```
bowlers["runs"] = bowlers["total_runs"] - (bowlers["bye_runs"] + bowlers["legbye_run
bowlers["extras"] = bowlers["wide_runs"] + bowlers["noball_runs"]

del( bowlers["bye_runs"])
del( bowlers["legbye_runs"])
del( bowlers["total_runs"])

dismissal_kinds_for_bowler = ["bowled", "caught", "lbw", "stumped", "caught and bowl
dismissals = deliveries[deliveries["dismissal_kind"].isin(dismissal_kinds_for_bowler
dismissals = dismissals.groupby(["match_id", "inning", "bowling_team", "bowler", "ov
dismissals.rename(columns={"dismissal_kind": "wickets"}, inplace=True)

bowlers = bowlers.merge(dismissals, left_on=["match_id", "inning", "bowling_team", "
                        right_on=["match_id", "inning", "bowling_team", "bowler", "o
bowlers["wickets"] = bowlers["wickets"].fillna(0)

bowlers_over = bowlers.groupby(['match_id', 'inning', 'bowling_team', 'bowler'])['ov
bowlers = bowlers.groupby(['match_id', 'inning', 'bowling_team', 'bowler']).sum().re
bowlers = bowlers_over.merge(bowlers, on=["match_id", "inning", "bowling_team", "bow
bowlers['Econ'] = np.round(bowlers['runs'] / bowlers['over'] , 2)
bowlers = matches[['id','season']].merge(bowlers, left_on = 'id', right_on = 'match_

bowlers.head(10)
```

```
<ipython-input-30-d8faef0f6cd5>:4: FutureWarning: Indexing with multiple keys (impli
citly converted to a tuple of keys) will be deprecated, use a list instead.
  bowlers = bowler_grp["total_runs", "wide_runs", "bye_runs", "legbye_runs", "noball
_runs"].sum().reset_index()
```

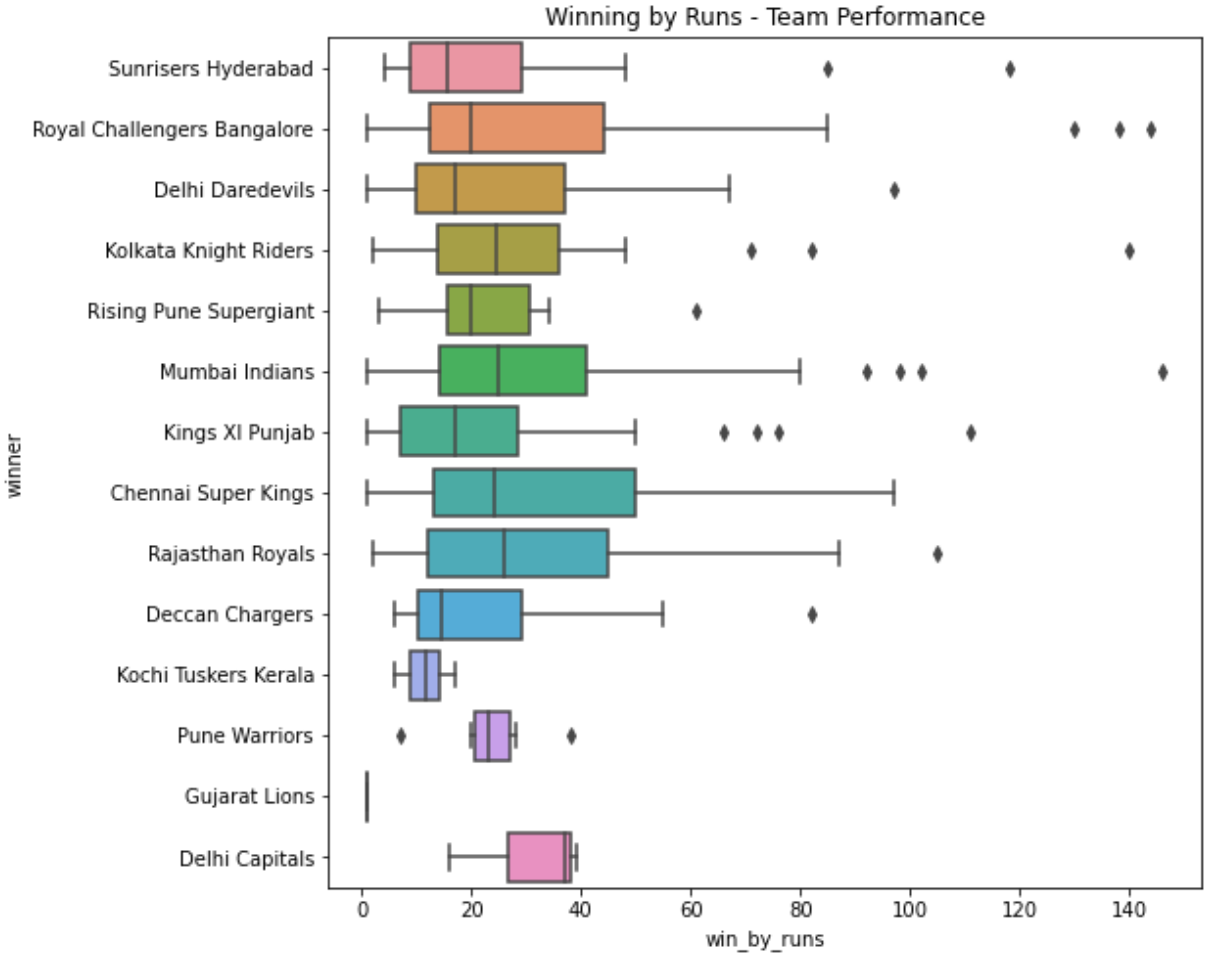
Out[30]:

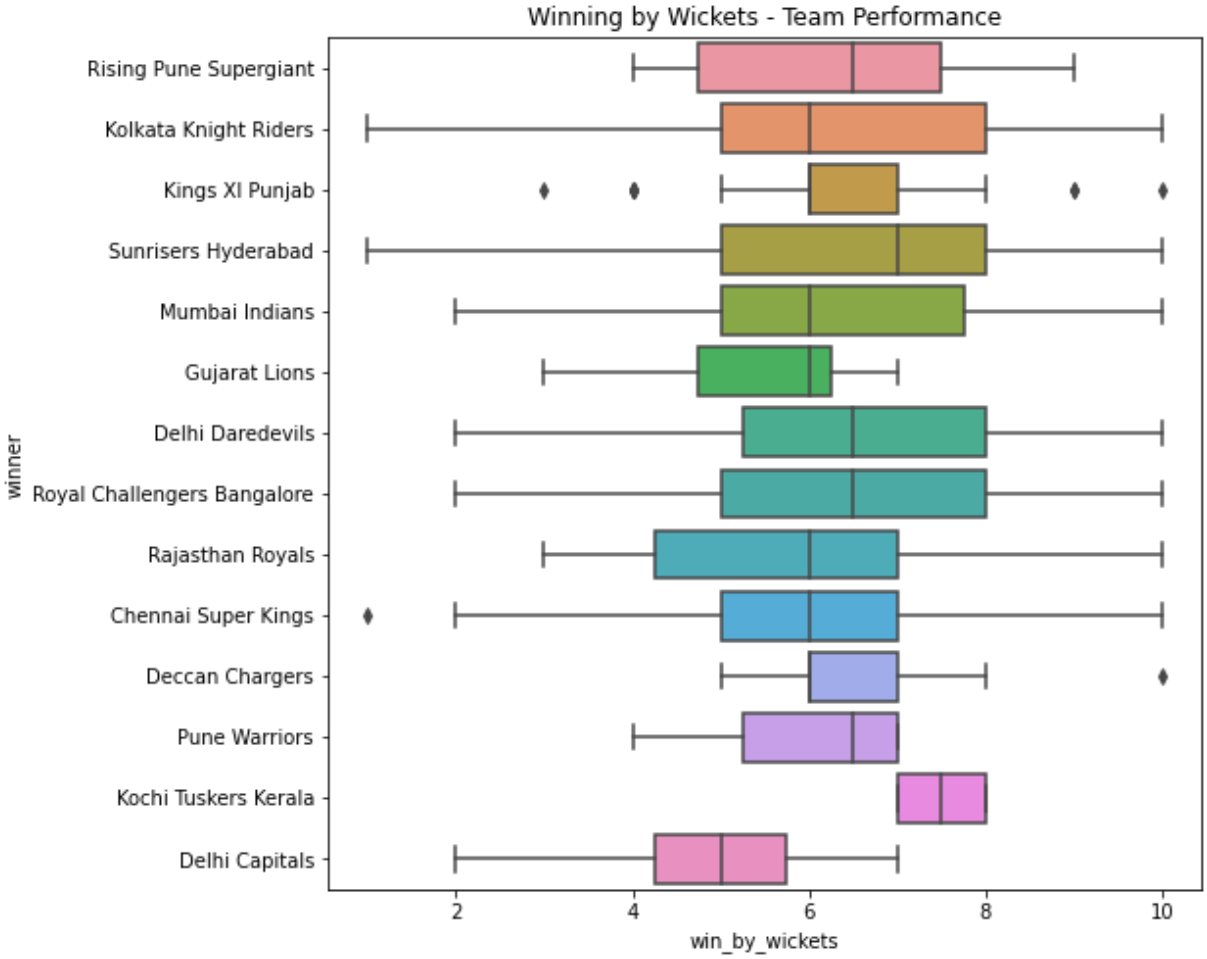
	season	match_id	inning	bowling_team	bowler	over	wide_runs	noball_runs	runs	extras
0	2017	1	1	Royal Challengers Bangalore	A Choudhary	4	3	1	55	4
1	2017	1	1	Royal Challengers Bangalore	S Aravind	3	0	0	36	0
2	2017	1	1	Royal Challengers Bangalore	SR Watson	3	0	0	41	0
3	2017	1	1	Royal Challengers Bangalore	STR Binny	1	0	0	10	0
4	2017	1	1	Royal Challengers Bangalore	TM Head	1	0	0	11	0
5	2017	1	1	Royal Challengers Bangalore	TS Mills	4	2	0	31	2
6	2017	1	1	Royal Challengers Bangalore	YS Chahal	4	0	0	22	0
7	2017	1	2	Sunrisers Hyderabad	A Nehra	4	1	0	42	1
8	2017	1	2	Sunrisers Hyderabad	B Kumar	4	1	0	27	1

	season	match_id	inning	bowling_team	bowler	over	wide_runs	noball_runs	runs	extras
9	2017	1	2	Sunrisers Hyderabad	BCJ Cutting	4	2	0	35	2

In [32]:

```
## Barplot of Runs

#sb.barplot(x="day", y="total_bill", data=tips)
fig, ax = plt.subplots()
#fig.figsize = [16,10]
#ax.set_ylim([0,20])
ax.set_xlabel("Runs")
ax.set_title("Winning by Runs - Team Performance")
#top_players.plot.bar()
sb.boxplot(y = 'winner', x = 'win_by_runs', data=matches[matches['win_by_runs']>0],
plt.show()

## Barplot of Wickets Win

#sns.barplot(x="day", y="total_bill", data=tips)
fig, ax = plt.subplots()
#fig.figsize = [16,10]
#ax.set_ylim([0,20])
ax.set_title("Winning by Wickets - Team Performance")
#top_players.plot.bar()
sb.boxplot(y = 'winner', x = 'win_by_wickets', data=matches[matches['win_by_wickets'
plt.show()
```





In [33]:
```
#WINNER PREDICTION

matches[pd.isnull(matches['winner'])]
```

Out[33]:

	id	season	city	date	team1	team2	toss_winner	toss_decision	result
300	301	2011	Delhi	2011-05-21	Delhi Daredevils	Pune Warriors	Delhi Daredevils	bat	no result
545	546	2015	Bangalore	2015-04-29	Royal Challengers Bangalore	Rajasthan Royals	Rajasthan Royals	field	no result
570	571	2015	Bangalore	2015-05-17	Delhi Daredevils	Royal Challengers Bangalore	Royal Challengers Bangalore	field	no result
744	11340	2019	Bengaluru	30/04/19	Royal Challengers Bangalore	Rajasthan Royals	Rajasthan Royals	field	no result

In [34]:
```
matches['winner'].fillna('Draw', inplace=True)
```

In [35]:
```
matches.replace(['Mumbai Indians','Kolkata Knight Riders','Royal Challengers Bangalo
                 'Rajasthan Royals','Delhi Daredevils','Gujarat Lions','Kings XI Pun
                 'Sunrisers Hyderabad','Rising Pune Supergiants','Rising Pune Superg
                ,['MI','KKR','RCB','DC','CSK','RR','DD','GL','KXIP','SRH','RPS','RPS

encode = {'team1': {'MI':1,'KKR':2,'RCB':3,'DC':4,'CSK':5,'RR':6,'DD':7,'GL':8,'KXIP
```

```
            'team2': {'MI':1,'KKR':2,'RCB':3,'DC':4,'CSK':5,'RR':6,'DD':7,'GL':8,'KXIP
            'toss_winner': {'MI':1,'KKR':2,'RCB':3,'DC':4,'CSK':5,'RR':6,'DD':7,'GL':8
            'winner': {'MI':1,'KKR':2,'RCB':3,'DC':4,'CSK':5,'RR':6,'DD':7,'GL':8,'KXI
matches.replace(encode, inplace=True)
matches.head(2)
```

Out[35]:

	id	season	city	date	team1	team2	toss_winner	toss_decision	result	dl_applied	wir
0	1	2017	Hyderabad	2017-04-05	10	3	3	field	normal	0	
1	2	2017	Pune	2017-04-06	1	11	11	field	normal	0	

In [36]:

```
#Find cities which are null
matches[pd.isnull(matches['city'])]
```

Out[36]:

	id	season	city	date	team1	team2	toss_winner	toss_decision	result	dl_applied	winn
461	462	2014	NaN	2014-04-19	1	3	3	field	normal	0	
462	463	2014	NaN	2014-04-19	2	7	2	bat	normal	0	
466	467	2014	NaN	2014-04-23	5	6	6	field	normal	0	
468	469	2014	NaN	2014-04-25	10	7	10	bat	normal	0	
469	470	2014	NaN	2014-04-25	1	5	1	bat	normal	0	
474	475	2014	NaN	2014-04-28	3	9	9	field	normal	0	
476	477	2014	NaN	2014-04-30	10	1	1	field	normal	0	

In [37]:

```
#remove any null values, winner has hence fill the null value in winner as draw
#City is also null
```

```
matches['city'].fillna('Dubai',inplace=True)
matches.describe()
```

Out[37]:

	id	season	dl_applied	win_by_runs	win_by_wickets
count	756.000000	756.000000	756.000000	756.000000	756.000000
mean	1792.178571	2013.444444	0.025132	13.283069	3.350529
std	3464.478148	3.366895	0.156630	23.471144	3.387963
min	1.000000	2008.000000	0.000000	0.000000	0.000000
25%	189.750000	2011.000000	0.000000	0.000000	0.000000
50%	378.500000	2013.000000	0.000000	0.000000	4.000000
75%	567.250000	2016.000000	0.000000	19.000000	6.000000
max	11415.000000	2019.000000	1.000000	146.000000	10.000000

In [38]:

```
dicVal = encode['winner']
print(dicVal['MI']) #key value
print(list(dicVal.keys())[list(dicVal.values()).index(1)]) #find key by value search
```

```
1
MI
```

In [39]:

```
matches = matches[['team1','team2','city','toss_decision','toss_winner','venue','win
matches.head(2)
```

Out[39]:

	team1	team2	city	toss_decision	toss_winner	venue	winner
0	10	3	Hyderabad	field	3	Rajiv Gandhi International Stadium, Uppal	10
1	1	11	Pune	field	11	Maharashtra Cricket Association Stadium	11

In [40]:

```
df = pd.DataFrame(matches)
df.describe()
```

Out[40]:

	team1	team2	city	toss_decision	toss_winner	venue	winner
count	756	756	756	756	756	756	756
unique	14	14	33	2	14	41	15
top	1	2	Mumbai	field	1	Eden Gardens	1
freq	101	95	101	463	98	77	109

In [41]:

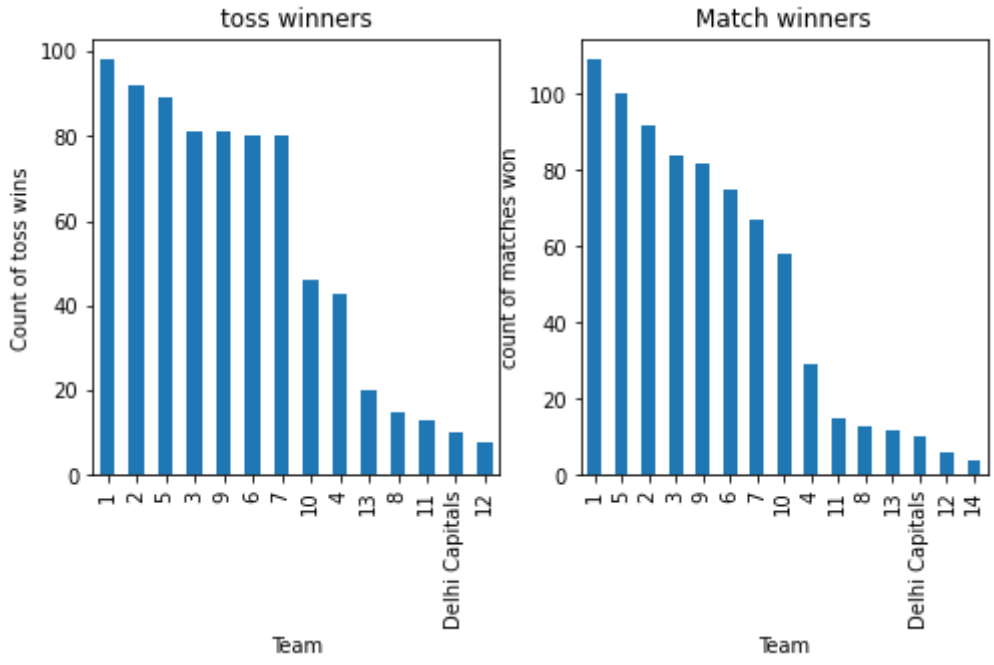
```
temp1=df['toss_winner'].value_counts(sort=True)
temp2=df['winner'].value_counts(sort=True)
#Mumbai won most toss and also most matches
print('No of toss winners by each team')
for idx, val in temp1.iteritems():
   print('{} -> {}'.format(list(dicVal.keys())[list(dicVal.values()).index(idx)],val
print('No of match winners by each team')
for idx, val in temp2.iteritems():
   print('{} -> {}'.format(list(dicVal.keys())[list(dicVal.values()).index(idx)],val
```

```
No of toss winners by each team
MI -> 98
KKR -> 92
CSK -> 89
RCB -> 81
KXIP -> 81
RR -> 80
DD -> 80
SRH -> 46
DC -> 43
PW -> 20
GL -> 15
RPS -> 13
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-41-08b5a73c00f8> in <module>
      4 print('No of toss winners by each team')
      5 for idx, val in temp1.iteritems():
----> 6     print('{} -> {}'.format(list(dicVal.keys())[list(dicVal.values()).index(i
dx)],val))
      7 print('No of match winners by each team')
      8 for idx, val in temp2.iteritems():

ValueError: 'Delhi Capitals' is not in list
```

In [43]:

```
fig = plt.figure(figsize=(8,4))
ax1 = fig.add_subplot(121)
ax1.set_xlabel('Team')
ax1.set_ylabel('Count of toss wins')
ax1.set_title("toss winners")
temp1.plot(kind='bar')

ax2 = fig.add_subplot(122)
temp2.plot(kind = 'bar')
ax2.set_xlabel('Team')
ax2.set_ylabel('count of matches won')
ax2.set_title("Match winners")
```

Out[43]: Text(0.5, 1.0, 'Match winners')



In [44]:

```
df.apply(lambda x: sum(x.isnull()),axis=0)
    #find the null values in every column
```

Out[44]:
```
team1            0
team2            0
city             0
toss_decision    0
toss_winner      0
venue            0
winner           0
dtype: int64
```

In [45]:
```
#Find cities which are null
df[pd.isnull(df['city'])]
```

Out[45]:

	team1	team2	city	toss_decision	toss_winner	venue	winner

In [46]:
```
#building predictive model
from sklearn.preprocessing import LabelEncoder
var_mod = ['city','toss_decision','venue']
le = LabelEncoder()
for i in var_mod:
    df[i] = le.fit_transform(df[i])
df.dtypes
```

Out[46]:
```
team1            object
team2            object
city              int32
toss_decision     int32
toss_winner      object
venue             int32
winner           object
dtype: object
```

In [47]:
```
# PREDCITION
```

In [48]:
```
points2008 = pd.read_csv('IPL 2008 PT.csv')
points2009 = pd.read_csv('IPL 2009 PT.csv')
points2010 = pd.read_csv('IPL 2010 PT.csv')
points2011 = pd.read_csv('IPL 2011 PT.csv')
points2012 = pd.read_csv('IPL 2012 PT.csv')
points2013 = pd.read_csv('IPL 2013 PT.csv')
points2014 = pd.read_csv('IPL 2014 PT.csv')
points2015 = pd.read_csv('IPL 2015 PT.csv')
points2016 = pd.read_csv('IPL 2016 PT.csv')
points2017 = pd.read_csv('IPL 2017 PT.csv')
points2018 = pd.read_csv('IPL 2018 PT.csv')
points2019 = pd.read_csv('IPL 2019 PT.csv')
```

In [49]:
```
points2019
```

Out[49]:

	Team	Pld	Won	Lost	Tied	N/R	Net RR	For	Against	Pts
0	Mumbai Indians MI	14	9	5	0	0	0.421	2,380/275.1	2,282/277.2	18
1	Chennai Super Kings CSK	14	9	5	0	0	0.131	2,043/274.1	2,012/274.5	18
2	Delhi Capitals DC	14	9	5	0	0	0.044	2,207/272.5	2,238/278.1	18
3	Sunrisers Hyderabad SRH	14	6	8	0	0	0.577	2,288/269.2	2,200/277.5	12

	Team	Pld	Won	Lost	Tied	N/R	Net RR	For	Against	Pts
4	Kolkata Knight Riders KKR	14	6	8	0	0	0.028	2,466/270.4	2,419/266.2	12
5	Kings XI Punjab KXIP	14	6	8	0	0	-0.251	2,429/276.3	2,503/277	12
6	Rajasthan Royals RR	14	5	8	0	1	-0.449	2,153/257	2,192/248.2	11
7	Royal Challengers Bangalore RCB	14	5	8	0	1	-0.607	2,146/258.4	2,266/254.3	11

In [50]:
```
point=points2008.append([points2009,points2010,points2011,points2012,points2013,poin
```

In [51]:
```
points = point.groupby('Team').mean()
```

In [52]:
```
new_matches = pd.read_csv('matches1234.csv')
```

In [53]:
```
match = new_matches.drop(['city','venue','umpire1','win_by_runs','win_by_wickets','s
```

In [54]:
```
def string_remover(df,list1=[],list2=[],drop=[],exclude=[],include=[]):
    a = df.select_dtypes(include='object')
    for i in a.columns:
        list1=[]
        d=0
        if not i in exclude and i in include:
            list1.append(i)
            for x in a.index:
                try:
                    c = list1.index(a[i].iloc[x:x+1][x])
                    a[i].iloc[x:x+1][x] = c
                except:
                    list1.append(a[i].iloc[x:x+1][x])
                    a[i].iloc[x:x+1][x] = len(list1)-1
            list2.append(list1)
            d+=1
    a.fillna(len(list1))
    d = df.select_dtypes(exclude='object').fillna(0)
    try:
        return pd.concat([d,a],axis=1).fillna(0).drop([drop],axis=1)
    except:
        return pd.concat([d,a],axis=1).fillna(0)
```

In [55]:
```
match['win']=None
match['toss_win']=None
for i in match.index:
    if match.winner[i] == match.team1[i]:
        match['win'][i] = 'team 1'
    else:
        match['win'][i] = 'team 2'
    if match.toss_winner[i] == match.team1[i]:
        match['toss_win'][i] = 'team 1'
    else:
        match['toss_win'][i] = 'team 2'
```

In [56]:
```
a=[]
b=['toss_decision', 'field', 'bat', 'toss_win', 'team 1', 'team 2']
```

In [57]:
```
match
```

Out[57]:

	team1	team2	toss_winner	toss_decision	winner	win	toss_win
0	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	field	Sunrisers Hyderabad	team 1	team 2
1	Mumbai Indians	Rising Pune Supergiants	Rising Pune Supergiants	field	Rising Pune Supergiants	team 2	team 2
2	Gujarat Lions	Kolkata Knight Riders	Kolkata Knight Riders	field	Kolkata Knight Riders	team 2	team 2
3	Rising Pune Supergiants	Kings XI Punjab	Kings XI Punjab	field	Kings XI Punjab	team 2	team 2
4	Royal Challengers Bangalore	Delhi Daredevils	Royal Challengers Bangalore	bat	Royal Challengers Bangalore	team 1	team 1
...	...	...	...	...	...	...	...
631	Delhi Daredevils	Royal Challengers Bangalore	Royal Challengers Bangalore	field	Royal Challengers Bangalore	team 2	team 2
632	Gujarat Lions	Royal Challengers Bangalore	Royal Challengers Bangalore	field	Royal Challengers Bangalore	team 2	team 2
633	Sunrisers Hyderabad	Kolkata Knight Riders	Kolkata Knight Riders	field	Sunrisers Hyderabad	team 1	team 2
634	Gujarat Lions	Sunrisers Hyderabad	Sunrisers Hyderabad	field	Sunrisers Hyderabad	team 2	team 2
635	Sunrisers Hyderabad	Royal Challengers Bangalore	Sunrisers Hyderabad	bat	Sunrisers Hyderabad	team 1	team 1

636 rows × 7 columns

In [58]:
```
match = string_remover(match,include=['toss_decision','toss_win'],list1=b,list2 = a)
```

In [59]:
```
b
```

Out[59]:
```
['toss_decision', 'field', 'bat', 'toss_win', 'team 1', 'team 2']
```

In [60]:
```
match
```

Out[60]:

	team1	team2	toss_winner	toss_decision	winner	win	toss_win
0	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	1	Sunrisers Hyderabad	team 1	1

	team1	team2	toss_winner	toss_decision	winner	win	toss_win
1	Mumbai Indians	Rising Pune Supergiants	Rising Pune Supergiants	1	Rising Pune Supergiants	team 2	1
2	Gujarat Lions	Kolkata Knight Riders	Kolkata Knight Riders	1	Kolkata Knight Riders	team 2	1
3	Rising Pune Supergiants	Kings XI Punjab	Kings XI Punjab	1	Kings XI Punjab	team 2	1
4	Royal Challengers Bangalore	Delhi Daredevils	Royal Challengers Bangalore	2	Royal Challengers Bangalore	team 1	2
...	...	...	...	...	...	...	...
631	Delhi Daredevils	Royal Challengers Bangalore	Royal Challengers Bangalore	1	Royal Challengers Bangalore	team 2	1
632	Gujarat Lions	Royal Challengers Bangalore	Royal Challengers Bangalore	1	Royal Challengers Bangalore	team 2	1
633	Sunrisers Hyderabad	Kolkata Knight Riders	Kolkata Knight Riders	1	Sunrisers Hyderabad	team 1	1
634	Gujarat Lions	Sunrisers Hyderabad	Sunrisers Hyderabad	1	Sunrisers Hyderabad	team 2	1
635	Sunrisers Hyderabad	Royal Challengers Bangalore	Sunrisers Hyderabad	2	Sunrisers Hyderabad	team 1	2

636 rows × 7 columns

In [61]:
```
points.drop('Tied',inplace=True,axis=1)
```

In [62]:
```
points.index = ['Chennai Super Kings', 'Deccan Chargers', 'Delhi Capitals',
       'Gujarat Lions', 'Kings XI Punjab', 'Kochi Tuskers Kerala',
       'Kolkata Knight Riders', 'Mumbai Indians',
       'Pune Warriors India', 'Rajasthan Royals',
       'Rising Pune Supergiants', 'Royal Challengers Bangalore',
       'Sunrisers Hyderabad']
```

In [63]:
```
points.index.name = 'Team'
```

In [64]:
```
for i in [match[match['team2']=='Delhi Daredevils'].index,match[match['team2']=='Pun
    for i in i:
        match.drop(i,inplace=True)

for i in [match[match['team1']=='Delhi Daredevils'].index,match[match['team1']=='Pun
    for i in i:
        match.drop(i,inplace=True)
```

In [65]:
```
points.to_csv('IPL_Points_Table.csv')
```

In [66]:
```
match['Team1_pts'] = None
match['Team1_RR'] = None
match['Team2_pts'] = None
match['Team2_RR'] = None
for i in match.index:
    match['Team1_pts'][i] = points.Pts[match.team1[i]]
    match['Team1_RR'][i] = points['Net RR'][match.team1[i]]
    match['Team2_pts'][i] = points.Pts[match.team2[i]]
    match['Team2_RR'][i] = points['Net RR'][match.team2[i]]
for i in match.index:
    match['Team2_pts'][i] = points.Pts[match.team2[i]]
    match['Team2_RR'][i] = points['Net RR'][match.team2[i]]
```

```
<ipython-input-66-7567bd35d126>:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  match['Team1_pts'][i] = points.Pts[match.team1[i]]
<ipython-input-66-7567bd35d126>:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  match['Team1_RR'][i] = points['Net RR'][match.team1[i]]
<ipython-input-66-7567bd35d126>:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  match['Team2_pts'][i] = points.Pts[match.team2[i]]
<ipython-input-66-7567bd35d126>:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  match['Team2_RR'][i] = points['Net RR'][match.team2[i]]
<ipython-input-66-7567bd35d126>:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  match['Team2_pts'][i] = points.Pts[match.team2[i]]
<ipython-input-66-7567bd35d126>:12: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  match['Team2_RR'][i] = points['Net RR'][match.team2[i]]
```

In [67]:
```
points
```

Out[67]:

Team	Pld	Won	Lost	N/R	Net RR	Pts
Chennai Super Kings	14.400000	8.700000	5.500000	0.200000	0.358400	17.600000
Deccan Chargers	14.400000	5.400000	8.800000	0.200000	-0.169600	11.000000
Delhi Capitals	14.333333	6.333333	7.750000	0.250000	-0.173417	12.916667
Gujarat Lions	14.000000	6.500000	7.500000	0.000000	-0.393000	13.000000
Kings XI Punjab	14.333333	6.750000	7.583333	0.000000	-0.197417	13.500000
Kochi Tuskers Kerala	14.000000	6.000000	8.000000	0.000000	-0.214000	12.000000

Team	Pld	Won	Lost	N/R	Net RR	Pts
Kolkata Knight Riders	14.333333	7.166667	6.833333	0.333333	0.083167	14.666667
Mumbai Indians	14.333333	8.250000	6.000000	0.083333	0.313333	16.583333
Pune Warriors India	15.333333	4.000000	11.000000	0.333333	-0.563667	8.333333
Rajasthan Royals	14.400000	7.200000	6.700000	0.500000	-0.097900	14.900000
Rising Pune Supergiants	14.000000	7.000000	7.000000	0.000000	0.095500	14.000000
Royal Challengers Bangalore	14.333333	6.583333	7.250000	0.500000	-0.050583	13.666667
Sunrisers Hyderabad	14.285714	7.714286	6.428571	0.142857	0.152857	15.571429

In [68]:
```
RC = RandomForestClassifier(random_state=815)
```

In [69]:
```
match.win = match['win']
```

In [70]:
```
match[['Team1_pts','Team1_RR','Team2_pts','Team2_RR',]]
```

Out[70]:

	Team1_pts	Team1_RR	Team2_pts	Team2_RR
0	15.571429	0.152857	13.666667	-0.050583
1	16.583333	0.313333	14.0	0.0955
2	13.0	-0.393	14.666667	0.083167
3	14.0	0.0955	13.5	-0.197417
5	13.0	-0.393	15.571429	0.152857
...	...	...	...	...
630	14.666667	0.083167	15.571429	0.152857
632	13.0	-0.393	13.666667	-0.050583
633	15.571429	0.152857	14.666667	0.083167
634	13.0	-0.393	15.571429	0.152857
635	15.571429	0.152857	13.666667	-0.050583

449 rows × 4 columns

In [71]:
```
RC.fit(X=match[['Team1_pts','Team1_RR','Team2_pts','Team2_RR','toss_decision','toss_
```

Out[71]:
```
RandomForestClassifier(random_state=815)
```

In [72]:
```
def Predicter(team1,team2,toss_winner,toss_decision):
    Team1_pts = points.Pts[team1]
    Team1_RR = points['Net RR'][team1]
    Team2_pts = points.Pts[team2]
    Team2_RR = points['Net RR'][team2]
    pred = RC.predict([[Team1_pts,Team1_RR,Team2_pts,Team2_RR,toss_decision,toss_win
    if pred == 'team 1':
```

```
        return team1
    else:
        return team2
```

In [73]:
```
Predicter('Sunrisers Hyderabad','Rajasthan Royals',toss_decision=1,toss_winner=2)
```

Out[73]: 'Rajasthan Royals'

In [ ]:

In [ ]:

In [ ]: