A Mini-Project Report

on

"Multi-Match Predictive Analysis and Visualization"

Submitted to the

Pune Institute of Computer Technology, Pune

In partial fulfillment for the award of the Degree of

Bachelor of Engineering

in

Information Technology

by

RUCHIKA PANDE	71829042K
AASHISH PRASAD	71829105M
VEDANT PURANIK	71829113B
SAKSHI TANTAK	71829281C

Under the guidance of

Prof. R. B. Murumkar

Department Of Information Technology

Pune Institute of Computer Technology College of Engineering

Sr. No 27, Pune-Satara Road, Dhankawadi, Pune - 411 043.

2019-2020

CERTIFICATE

This is to certify that the project report entitled

MULTI-MATCH PREDICTIVE ANALYSIS AND VISUALIZATION

Submitted by,

RUCHIKA PANDE	71829042K
AASHISH PRASAD	71829105M
VEDANT PURANIK	71829113B
SAKSHI TANTAK	71829281C

is a bonafide work carried out by them under the supervision of Prof. R. B. Murumkar and it is approved for the partial fulfillment of the requirement of Software Laboratory Course-2015 for the award of the Degree of Bachelor of Engineering (Information Technology)

Prof. R. B. Murumkar **Dr. A. M. Bagade**

Internal Guide Head of Department

Department of Information Technology Department of Information Technology

Date:

Place:

ACKNOWLEDGEMENT

On the very outset of this project, we would like to extend our sincere and heartfelt obligation towards all the personages who have helped us in this endeavor. Without their active guidance, help and cooperation, we would have not made headway in the project.

We express our sincere gratitude to Dr. Prahlad Kulkarni, Principal of Pune Institute of Computer Technology.

We pay our deep sense of gratitude to Dr. Anant Bagade, IT HOD, who has encouraged us to the highest peak and has provided us with this opportunity.

We are ineffably indebted to our mentor Prof. Ravi Murumkar Sir for continuous evaluation, encouragement and supervision given throughout the project which shaped the present work.

Lastly, we acknowledge with a deep sense of reverence, our gratitude towards our parents and members of our family, who have always supported us morally as well as economically.

Ruchika Pande

Aashish Prasad

Vedant Puranik

Sakshi Tantak

Abstract

Indian Premier League is the most anticipated cricket tournament that is organized every year by the BCCI. Here, different information such as the performance of players and the probability of a team winning a match is of utmost importance and can also be used as metrics for deciding the team's strategy and the players who will be playing against a particular team. Hence we built a match prediction system that predicts which team, out of the two currently playing, has a higher probability of winning the match by using the Decision Tree Classifier, the Random Forest Classifier, and the XGBClassfier.

For this, two datasets were used: one which had a ball-by-ball data for every match played in the last 11 years of the Indian Premier League and the other which consists of a summary of the statistics of the match. Data was pre-processed where all the null values were either removed or replaced by their mean (wherever necessary). All the outliers were manually removed instead of using any particular method because of the nature of the dataset. Feature selection was initially done by evaluating and understanding the necessity of prevalence of the features, and further done again more accurately while tuning the model, which led to selection of the following features essential for building and training the models: team1, team2, win_by_runs, player_of_match, win_by_wickets. Multiple models such as Random Forest Classifier, the Decision Tree Classifier and the XGBClassfier were trained and tested wherein XGBClassifier attained the highest Test distribution accuracy of 96.04%. Visualization of the two datasets was done in a combined fashion by using the open-source libraries called 'matplotlib', 'seaborn' and 'plotly' which analyzed different aspects of the teams and the matches such as player (batsman/bowler) performance over the years, percentage of match wins, run distribution over the years, etc.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Motivation

Sports analytics and data visualization has provided a great platform for player selectors, managers and also the players to increase on-field performance. Decision making and analysis, the next piece of the framework, is the process of applying statistical tools and algorithms to data to gain insight into what is likely to happen in the future. Sports analytics is being applied in IPL. Each movement of the ball, the player strike rate, run rate, everything is captured using special camera systems and other recording mechanisms. This data is run through various statistical algorithms, tools and visualization techniques to provide deeper insight and pave way for recommendations to the player or team. With the ease of obtaining and storing data, advanced analytics and machine learning techniques can be applied to engineer a predictive model for various team sports like cricket. This motivated us to come up with our own prediction model for IPL and also to provide insight into trends with the help of data visualization.

1.2 Purpose

Analytics bridges the gap for team selectors, coaches, and managers. Analytics gives us a clearer idea about player consistency, fast scoring and finishing ability. To manage the risk in a better way and to get the probable winners, analytics play a crucial role in the field and out of the field. Data visualization is one of the major outcomes in sports analytics. The visual form of data is more easily understandable over numbers and text. This project explores the data visualization techniques and a predictive model that supports the decision makers for identifying inherent players for their team and in carrying out other tasks. The major purpose of the project is hence to get an insight into IPL data and to get better visualization of player and team statistics.

1.3 Need

Due to the enormous measure of cash associated with IPL, it imposes high pressure on team players and additionally on team owners to search for victories. To earn more revenue, a team needs to win more successes. Purchasing great players for the team and shaping the most ideal team inside a given spending plan can make an extraordinary chance to increase the winnability of a team. For this, basically, it is critical to find the right set of metrics that would prompt assembling a team with the highest chance of winning. Hence this project serves as a very primitive but not an inapplicable model for prediction and visualization of IPL data.

CHAPTER 2

LITERATURE SURVEY

1.

Title : “Prediction of the outcome of a Twenty-20 Cricket Match.”
Author : Arjun Singhvi, Ashish V Shenoy, Shruthi Racha, Srinivas Tunuguntla

Methodology :

Modelling player batting and bowling ratings:

Cricket players are based on the aggregated statistics of players: total runs/total wickets, batting/bowling average, strike rate/economy rate etc. All of these methods suffer from one big flaw, which is that any player can have inflated statistics by playing against “weaker” teams, and therefore achieve a higher rating which may not reflect the true standing of the player. In this approach we present a new rating method which automatically takes care of the “strong” and the “weak” opponents, and is therefore more reflective of the true rating of the player in relation to his peers. The paper considers the pairwise interactions between batsmen and bowlers, converts the pairwise interactions into a “score” and uses these to rank the players. A score for an interaction between batsman i and bowler j is modelled as:

s !" = A + a ! − b ! + ε

where a ! is the batting rating of player i and b ! is the bowling rating of player j. The intercept A represents the expected score between average players. ε is a zero-mean random error. The error term is included because two competing players are not necessarily expected to repeat the same score in two different interactions.

Limitation and Future Scope:

The main limitation in carrying out this project is the limited dataset. The next logical step in the direction to improve the accuracy of the prediction problem at hand would be to test out the approaches and various methodologies proposed in this paper using a larger and more representative dataset. Also this study encourages to extend the candidate classifier set considered to a more exhaustive list and compare the performances among them.

2.

Title : "Dynamic Winner Prediction in Twenty20 Cricket: Based on Relative Team Strengths."
Author : Sasank Viswanadha, Kaustubh Sivalenka, Madan Gopal Jhawar

Methodology :

In this paper, a model is proposed to predict the winner at the end of each over in the second innings of an IPL cricket match. The methodology not only incorporates the dynamically updating game context as the game progresses, but also includes the relative strength between the two teams playing the match. Estimating the relative strength between two teams involves modeling the individual participating players' potentials. To model a player, it uses his career as well as recent performance statistics. Using the various dynamic features, it evaluates several supervised learning algorithms to predict the winner of the match. Finally, using the Random Forest Classifier (RFC), it has achieved an accuracy of 65.79% - 84.15% over the course of second innings, with an overall accuracy of 75.68%.

Conclusion and Future Scope:

The problem of dynamic winner prediction in a Twenty20 cricket match has been successfully addressed in this paper. A combination of features which capture the state of the match have furnished promising results. Relative strength Team B /Team A has been shown as an important feature that is successful in quantifying and comparing the strengths of the playing teams. In order to further make the prediction model adapt at addressing the entire match scenario, they intend to extend their approach in order to account for the innings 1 dynamics as well. The primary challenge that stands in the way of this is to estimate the score that the team batting first is expected to score.

3.

Title : “Live Cricket Score and Winning Prediction”
Author : Rameshwari Lokhande,P.M.Chawan

Methodology:

This paper will execute a live cricket score and winning forecast. In this model there are two sections. In the initial segment it will anticipate adding up the score of the group which is bat first. To anticipate the aggregate score it will consider different variables; hurl, number of wickets fallen, setting, home diversion advantage, day/night impact and positioning of the group. In this framework first it will watch who will win the hurl and which group bat first. At that point it will be dissected that will add up to the score of batting initially grouped from the past record. The second part model will indicate who will win the match by utilizing previously mentioned factors.

Conclusion:

The main purpose of this paper is to make a model for predicting the final score of the first innings and estimating the outcome of the match in the second innings for the limited overs cricket match. Factors like the toss, the ODI ranking of the teams and the home team advantage will be considered in the predictions. Two separate models, one for the first innings and other for the second innings using the Linear Regression classifier and Naive Bayes classifier respectively on the past matches have been proposed. Reinforcement algorithms will be used instead of linear regression.

CHAPTER 3

DESIGN DETAILS AND IMPLEMENTATION

3.1 Dataset Description

Dataset	Description
1. deliveries.csv	179078 rows and 21 columns (ball-by-ball dataset)
2. matches.csv	756 rows and 18 columns (match-by-match dataset)

1. deliveries.csv

Column	Description
match_id	specifies the id of the match (starts from 1)
inning	specifies inning number in which ball was delivered
batting_team	specifies the name of the batting team
bowling_team	specifies the name of the bowling team
over	specifies the number of the current over (out of 20).
ball	specifies the number of ball in the current over (out of 6 or greater)
batsman	specifies the name of the batsman (striker)
non_striker	specifies the name of the batsman (non-striker)
bowler	specifies the name of the current bowler delivering the ball

is_super_over	specifies if over being played is super over or not
wide_runs	specifies if any wide runs were given on the current ball
bye_runs	specifies if any bye runs were given on the current ball
legbye_runs	specifies if any leg-bye runs were given on the current ball
noball_runs	specifies if any no ball runs were given on the current ball
penalty_runs	specifies if any penalty runs were given on the current ball
batsman_runs	specifies if the batsman scored any runs on the current ball
extra_runs	specifies if any extra runs were made/given on the current ball
total_runs	specifies the total batsman runs and extra runs made on the current ball
player_dismissed	specifies if current player was dismissed on the ball or not
dismissal_kind	specifies the dismissal kind (if any)
fielder	specifies the fielder involved in the dismissal if any.

Table 1: Dataset Description: deliveries.csv

2. **matches.csv**

Column	**Description**
id	specifies the id of the match (starts from 1)
season	specifies the year in which season happened
city	specifies the city in which the match was played
date	specifies the date on which the match was played
team1	specifies name of team 1
team2	specifies name of team 2
toss_winner	specifies the name of the team who won the toss
toss_decision	specifies whether the winning team chose to field or bat
result	specifies if the result of the match was normal or under different conditions
dl_applied	specifies whether Duckworth-Lewis rule was applied or not
winner	specifies the name of the winning team of the match
win_by_runs	specifies the number of runs by which the match was won by the winning team
win_by_wickets	specifies the number of wickets by which the match was won by the winning team
player_of_match	specifies the name of the man of the match

venue	specifies the name of the venue where the match was played
umpire1	specifies the name of the umpire (1)
umpire2	specifies the name of the umpire (2)
umpire3	specifies the name of the umpire (3)

Table 2: Dataset Description: matches.csv

Links for datasets: https://www.kaggle.com/nowke9/ipldata

3.2 Data preprocessing

Data preprocessing was done extensively on both the data sets i.e. deliveries.csv and matches.csv

1. deliveries.csv

Initially it was observed that the columns titled: player_dismissed, dismissal_kind and fielder consisted of a large number of null values as it was very natural that a wicket would not fall on every ball and maximum 10 balls per match would be those on which wickets were taken. Hence in order to remove all the unnecessary space occupied by these three columns a new column was introduced called 'is_player_dismissed'. This column specified whether a player was dismissed on the particular ball or not: 1 means player is dismissed and 0 indicates that a player is not dismissed.
As all the other information (in the three columns) became redundant, they were removed from the dataset and stored in a separate data frame if in case they would be required in the future for visualization or analysis purposes. The corresponding rows (8834) were also removed from the dataset that accounted for the wickets.

Furthermore, the outliers were removed manually and no particular method like Interquartile Range (IQR) or Z-Score was used as the outliers were not in terms of extreme values (greater or lesser) but showcased a different nature which could not be handled by a particular formula and needed manual treatment. If the value of the batsman_runs made were greater than 6 then they were brought down to 6 and correspondingly the total_runs column's value was adjusted as per the new change made. Similarly it was also made sure that the data correctly represented the scenario in case of wide balls. As per law 22 of laws of cricket, A delivery is a wide if it is not sufficiently within reach for the batter to be able to hit it with the bat by means of a normal cricket stroke from where the batter is standing, and also would not have been sufficiently within reach for the batter to be able to hit it with the bat by means of a normal cricket stroke if the batter were standing in a normal guard position. When a wide is bowled then no runs are added to the batsman's runs but only to the team total. Hence all those places in the dataset where more than 0 runs were allocated to the batsman on a wide ball were brought down to zero.

Finally, as all teams must have unique names, all duplicate names present in the dataset i.e. both Rising Pune Supergiant and Rising Pune Supergiants were given a single and unique name

2.matches.csv

The Machine Learning model was required to be built on “matches.csv”, therefore the preprocessing involved various areas that eventually resulted into converting the entire dataset into a numerical dataset that could fit a Machine Learning model.
The pre-processing of this dataset was carried out in the following manner:

1. Renaming team names for ease to work with in the flow of the project:
    - The following columns of “matches.csv” had names of the teams that participated in the seasons of IPL under analysis here, namely “team1”, and “team2” and “winner”.
    - Thus a dictionary was prepared mapping the team names to short abbreviations, and the team names in the entire dataset were replaced.

2. Handling Null Values:
    - The null values were seen to occur in the following columns, namely “winner”, “city” and “umpire3”.
        - “winner” : When there is no clear winner to a match, it is clearly a Draw, and thus null values in this column were replaced with “Draw”.
        - “city” : A default city, here “Dubai”, was filled into the null values in this column.
        - “umpire3” : It was observed that the fraction of non-null value of this column was very low, 0.033 approximately, and the correlation coefficient between this column and the target or dependant variable, i.e, “winner”, was found to be negative, and thus it was dropped for the better, assuming it would not substantially affect the accuracy scores of the Machine Learning Model.

3. Label Encoding:
    - The columns, namely, “date”, “team1”, “team2”, “toss_winner”, “toss_decision”, “result”, “winner”, “player_of_match”, “venue”, “umpire1”, “umpire2”, “umpire3” were of non-numerical data-types and had to be encoded into numerical form to fit a Machine Learning model on the dataset.

4. Removal of not-so-necessary columns:
    - It was assumed that the columns “id” and “date” would not hugely influence the model and thus were dropped.
    - The column “umpire3” had a very low fraction of non-null values and the correlation coefficient with “winner” was negative. Thus this column was also dropped with the same, above stated, assumption.

5. Scaling the columns:
    - The column “season” had values in the range (2008, 2019) which represented the season which the particular match had been played in. To reduce the value of these, 2007 was subtracted from the values in the “season” and thus the values in this column were reduced to lie in the range (1, 11).

6. Splitting Training and Testing Data:
    - The dataset was split into initially, the independent variables, X and independent variable y.

- Further, X and y were split into training and test distributions in a ratio of 7:3 to avoid overfitting or underfitting of the model.

3.3 Data Model Selection and Model Building

The next step after preprocessing is to select an appropriate Machine Learning Model and build it to fit on the training data. The Model selection for this dataset was done on the nature of the target variable, which clearly had multiple classes, and so this problem had to be handled using classification, or regression at first hand. Since the target variable classes were discrete and not continuous, the classification model was eventually preferred.

Three different classifiers were chosen for training and testing against the data

1. Decision Tree Classifier.
2. Random Forest Classifier.
3. XGBoost Classifier

1. About Decision Tree Classifiers

- Decision tree classifiers are utilized as a well known classification technique in different pattern recognition issues, for example, image classification and character recognition
- In particular, no distribution assumption is needed by decision tree classifiers regarding the input data. This particular feature gives to the Decision Tree Classifiers a higher adaptability to deal with different datasets, whether numeric or categorical, even with missing data. Also, decision tree classifiers are basically nonparametric.
- The basic idea behind any decision tree algorithm is as follows: Select the best attribute using Attribute Selection Measures(ASM) to split the records, make that attribute a decision node and breaks the dataset into smaller subsets, starts tree building by repeating this process recursively for each child until one of the condition will match:
  - All the tuples belong to the same attribute value.
  - There are no more remaining attributes.
  - There are no more instances.

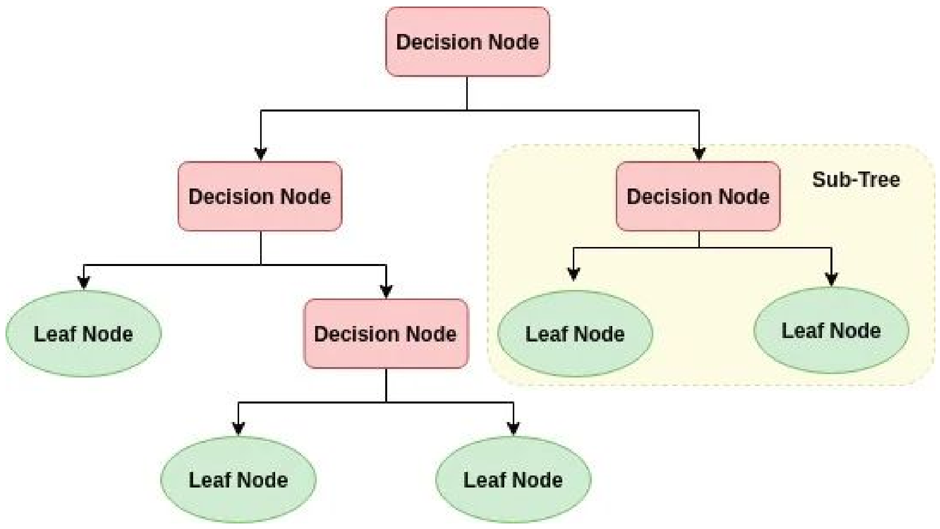

Figure 2: Decision Tree Classifier Model Figure

2. About Random Forest Classifier

- Random forests are supervised learning algorithms. They can be used both for classification and regression. They are also the most flexible and easy to use algorithms.
- Random forests work by creating decision trees on randomly selected data samples, make predictions from each tree, and select the best combinations of features and parameters by means of voting. It gives the feature importances to see which features are contributing towards maximum accuracy scores.
- It works in four steps: Select random samples from a given dataset, construct a decision tree for each sample and get a prediction result from each decision tree, perform a vote for each predicted result and then select the prediction result with the most votes as the final prediction.

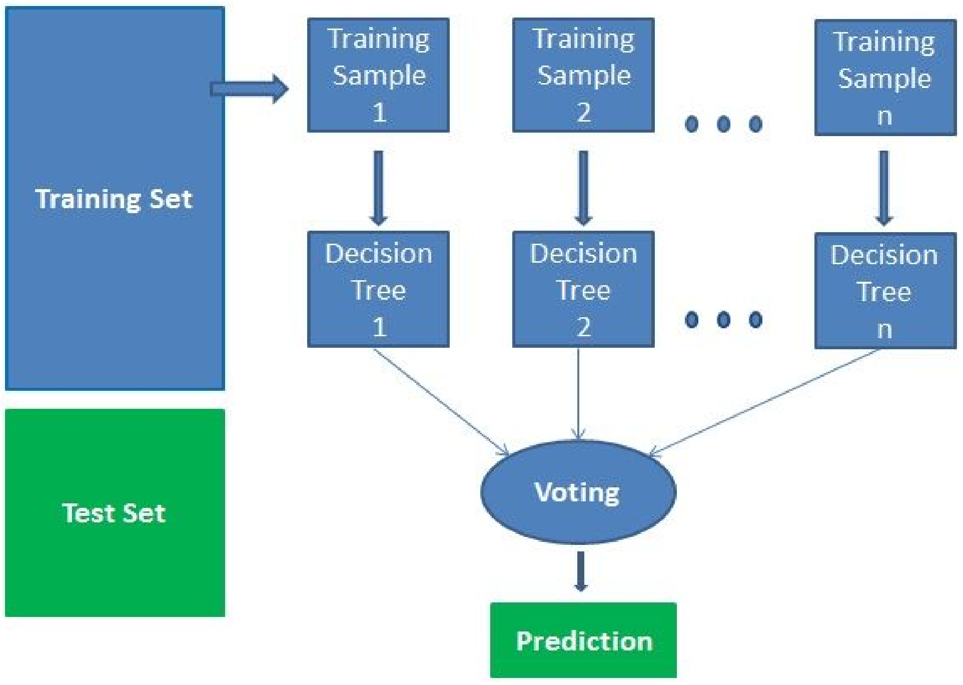

Figure 1: Random Forest Classifier Model Figure

3. **About XGBClassifier :**

- The XGBoost API has XGBClassifier in sklearn for python, which is basically a gradient boosted machine.
- XGB is an ensemble of weak models that, when put together, give robust and accurate results. The weak models can be decision trees, which can be randomized in the same way as random forests.
- This Classifier is computationally expensive for large datasets, and relatively slow. Despite this, it was chosen to work with because of the small size and low dimensionality of the dataset.

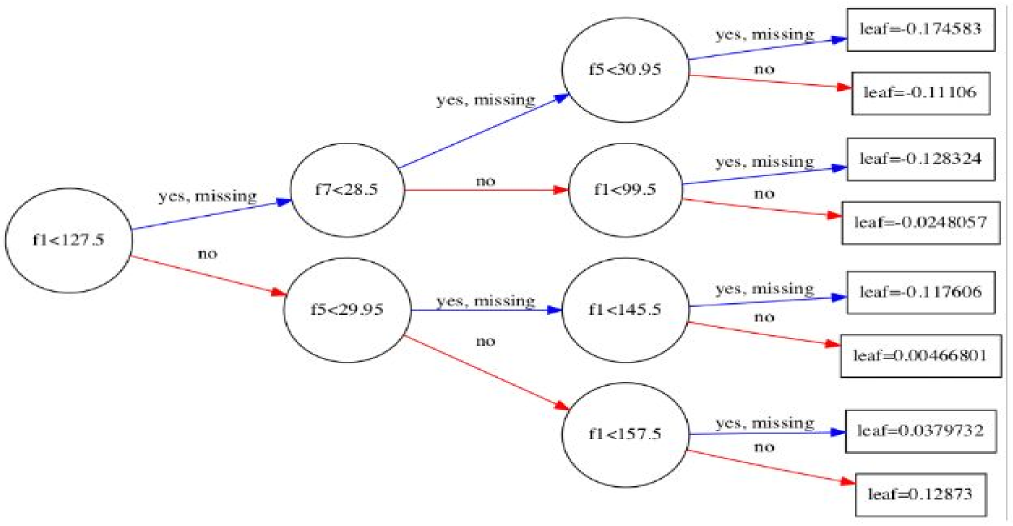

Figure 3: XGBoost Classifier Model Figure

3.4 Prediction:

The predictions from the 3 models were found to be as follows:

a. Decision Trees Classifier with default hyperparameter values:
   - Train distribution Accuracy = 100%
   - Test distribution Accuracy = 77.53%

   Thus it was observed that the model was highly overfitting.

b. Random Forest Classifier with default hyperparameter values:
   - Train distribution Accuracy = 99.81%
   - Test distribution Accuracy = 56.82%

   Thus it was observed that this model was no better than the Decision Trees Classifier, and in fact had a higher difference between the train and test accuracies than in Decision Trees Classifier.

c. XGBClassifier with default hyperparameter values:
   - Train distribution Accuracy = 100%
   - Test distribution Accuracy = 91.19%

   It was concluded that although this model was overfitting, it was still performing better than the other two, and thus it was tuned for optimum values of hyperparameters.

d. Hyperparameter tuning for XGBClassifier manually:
   - Train distribution Accuracy = 100%
   - Test distribution Accuracy = 95.59%

   This was also found to be overfitting to an extent. So for finding better values for hyperparameters, Grid Searching was used.

e. Hyperparameter tuning for XGBClassifier using Grid Search:
   - Train distribution Accuracy = 100%
   - Test distribution Accuracy = 96.04%

   This is evidently still a bit overfitting, but since the model is performing much better than the earlier models, it was accepted. Finding the balance between a good accuracy and overfitting is indeed a challenging task while building most Machine Learning models, but a good fitting model should not be discarded reluctantly without analysing, just because it is slightly overfitting. Another reason why the tuning was stopped at this particular stage was because when testing on a wider range of values in Grid Search, the time taken is a lot, and also note that XGBoost algorithms are difficult to optimize after a certain point. Thus this was accepted as the final model for predictions.

3.5 Visualization

For visualization of data, three open-source python-based libraries viz. matplotlib, seaborn and plotly were used. Matplotlib and seaborn were used for primitive data visualization whereas plotly was used for interactive visualization wherein the user can play with the visuals generated in the sense that he can zoom in/out, focus on one particular area of the plot,etc.

A total of 13 charts/graphs were plotted using the above mentioned libraries.

1. A countplot (bar graph) using seaborn was plotted. This countplot indicated the number of matches that were played per season from 2008-2019.

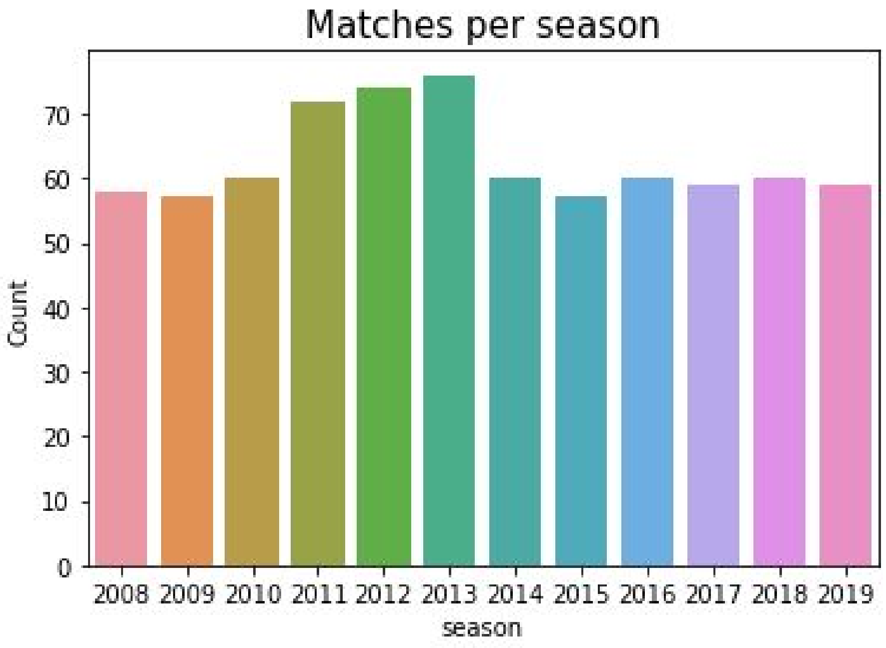

Figure 4: Number of matches per season

2. A barplot using seaborn was plotted. This bar plot indicates the number of times every team has won a match over all the different seasons of IPL that were played.

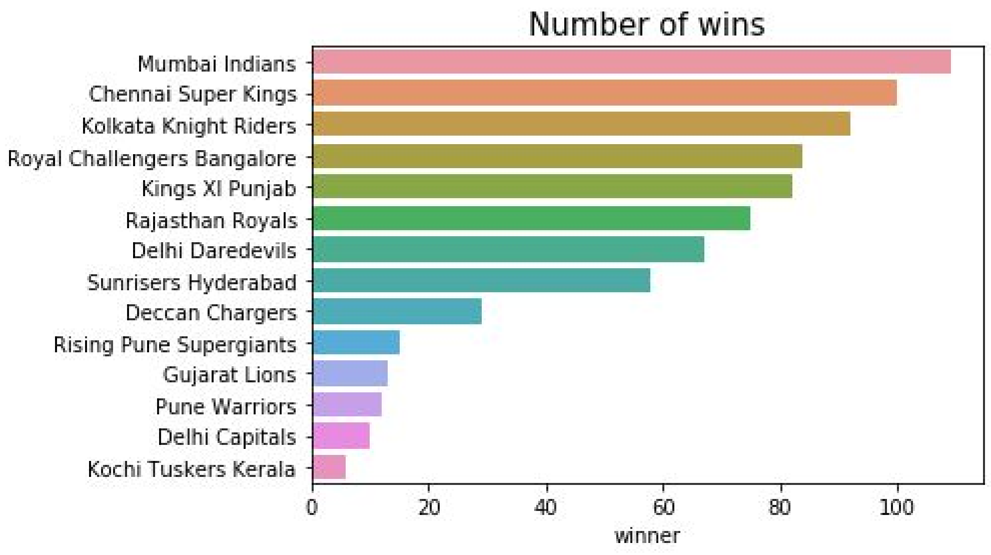

Figure 5: Number of wins per team

3. A countplot using seaborn was plotted. This countplot indicated the number of times each venue available in the dataset was utilized for an IPL Match.

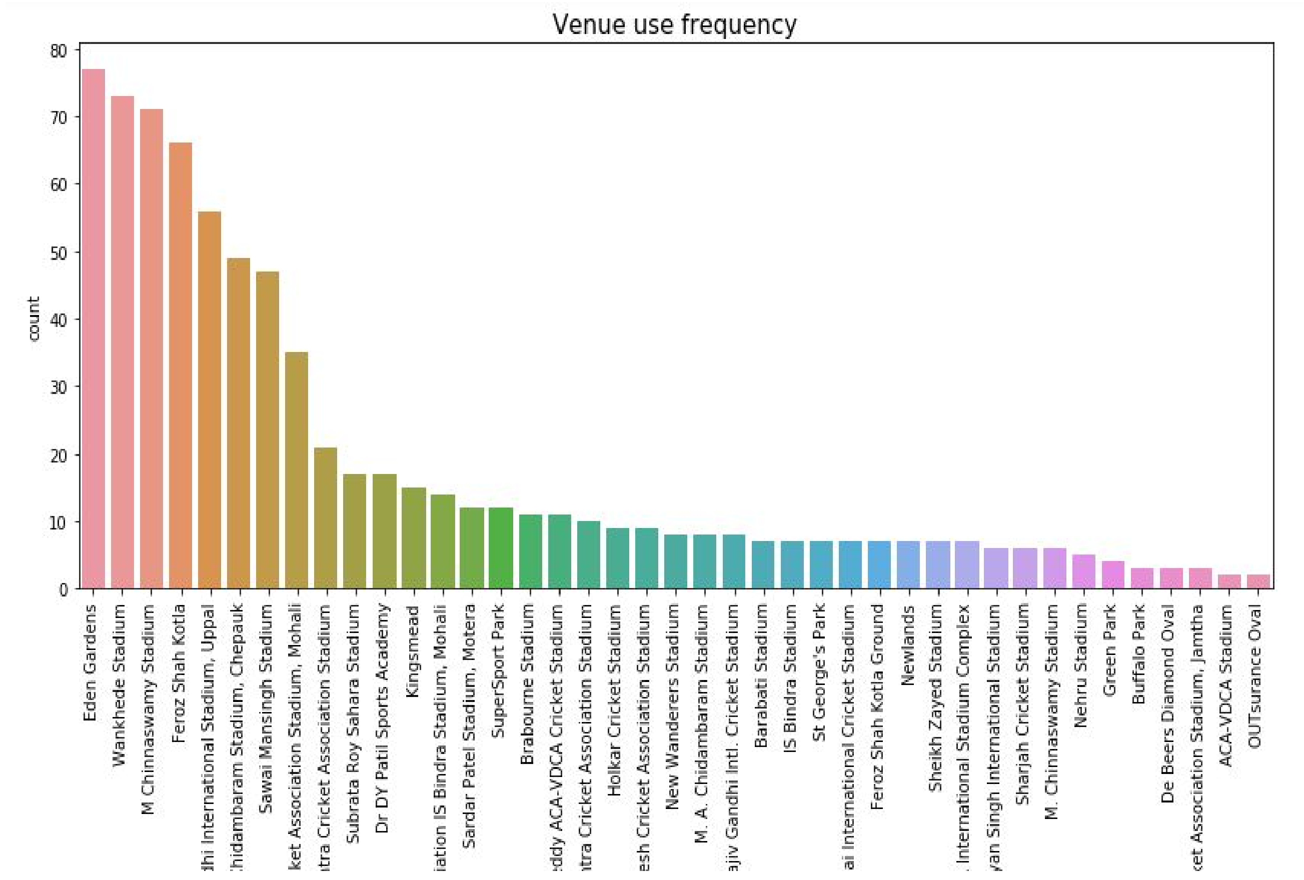

Figure 6: Frequency of venue use

4. A countplot using seaborn was plotted. This countplot indicated the number of times teams chose to bat or field after winning the toss i.e. the preference given to batting or fielding after winning a toss over the seasons.

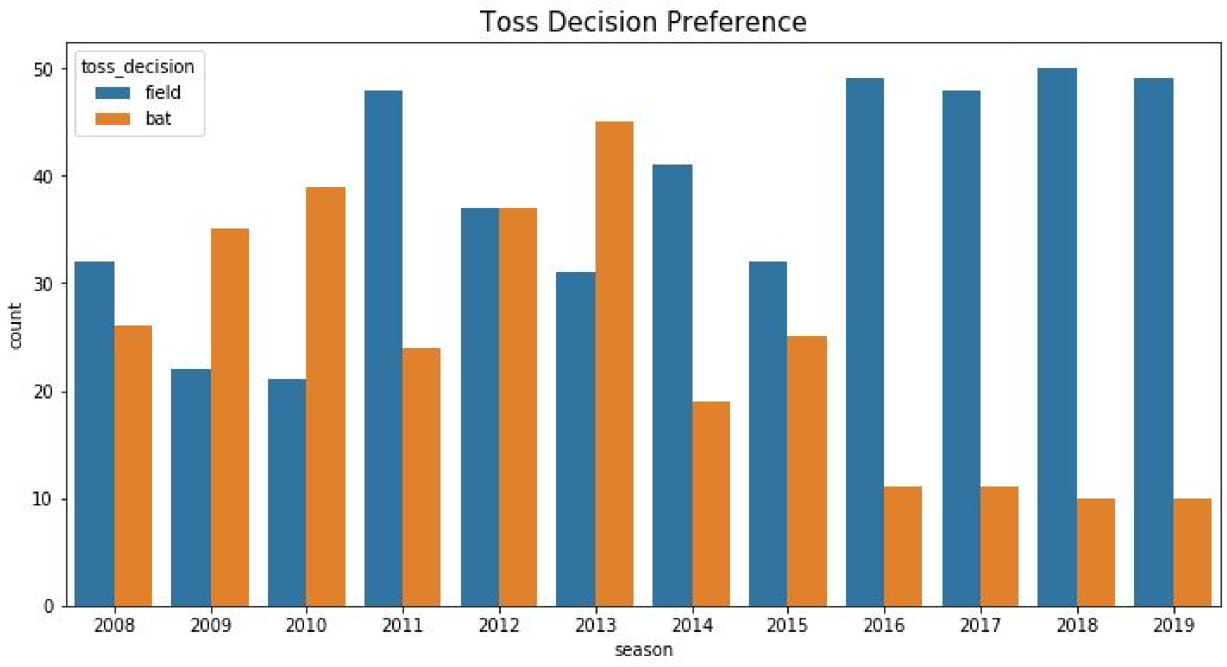

Figure 7: Toss Decision Preference

5. A histogram was plotted using Plotly. This was the same plot as the first one but it provided interactive visualization where the user can interactively view the plot.

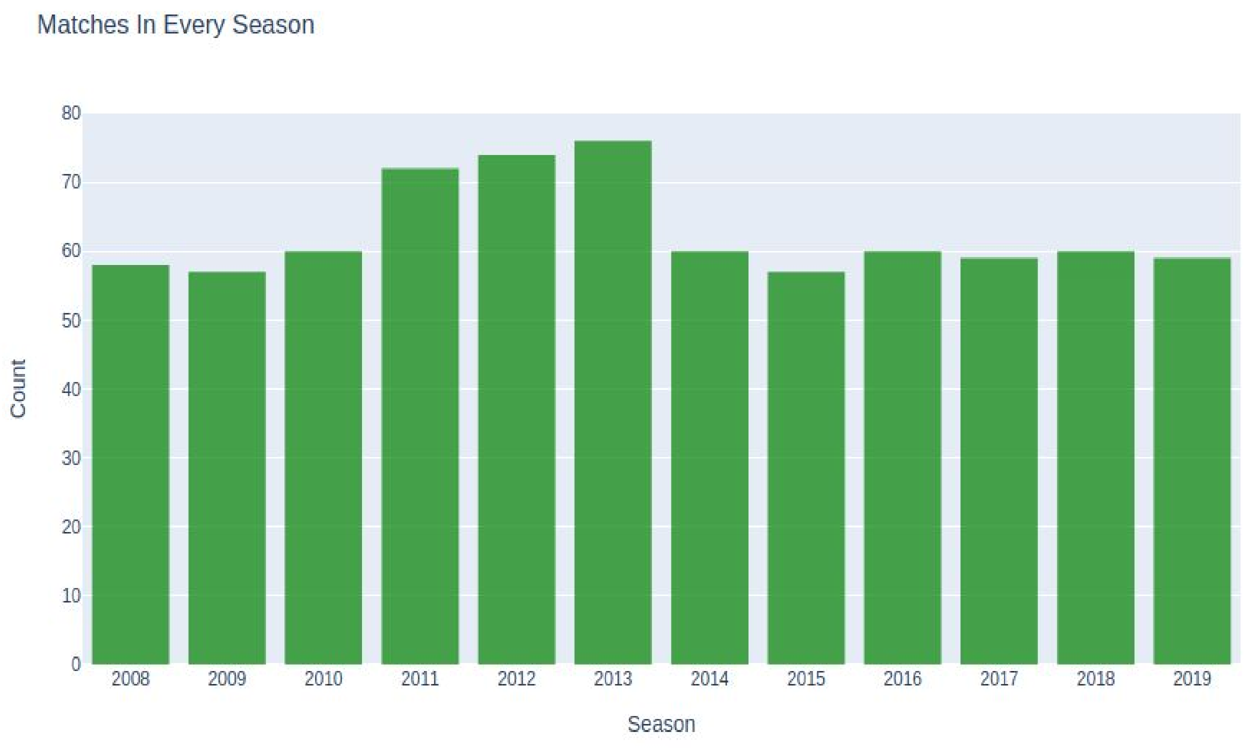

Figure 8: Number of matches per season (plotly)

6. A stack plot was plotted using Plotly. This stack plot indicated the run distribution over the years in terms of 6’s, 4’s and remaining runs.

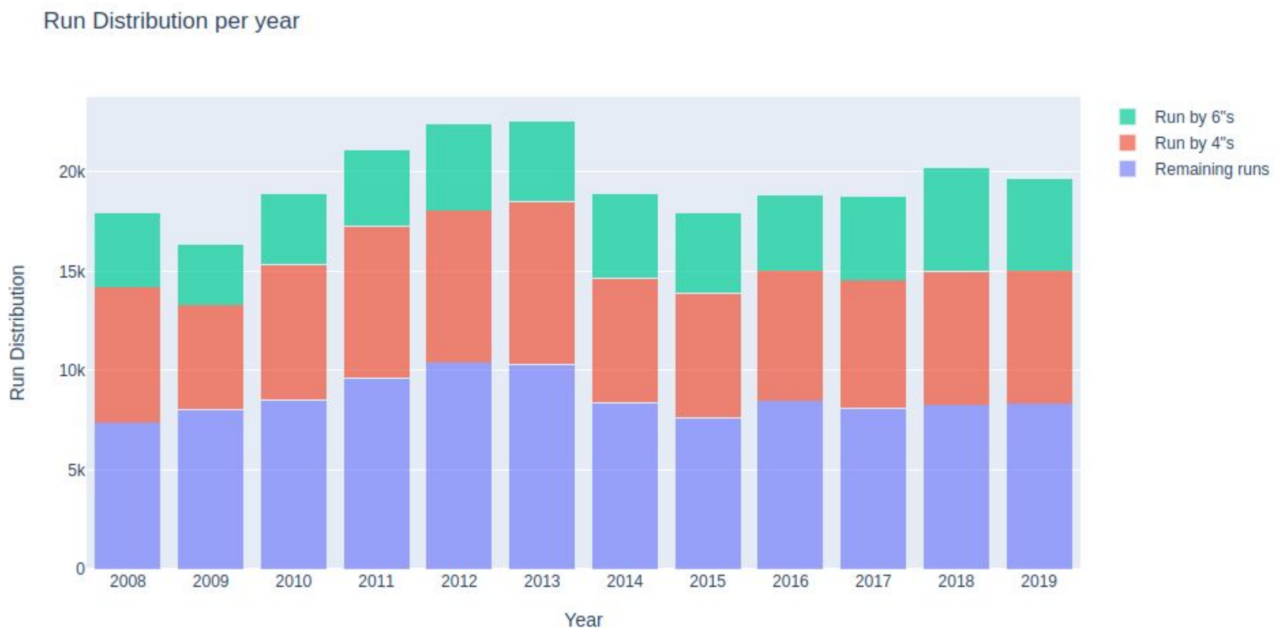

Figure 9: Run distribution per year

7. A combined line chart was plotted using Plotly. This chart showcases the average number of runs scored by each team per season when they chose to field first vs. when they chose to bat first.

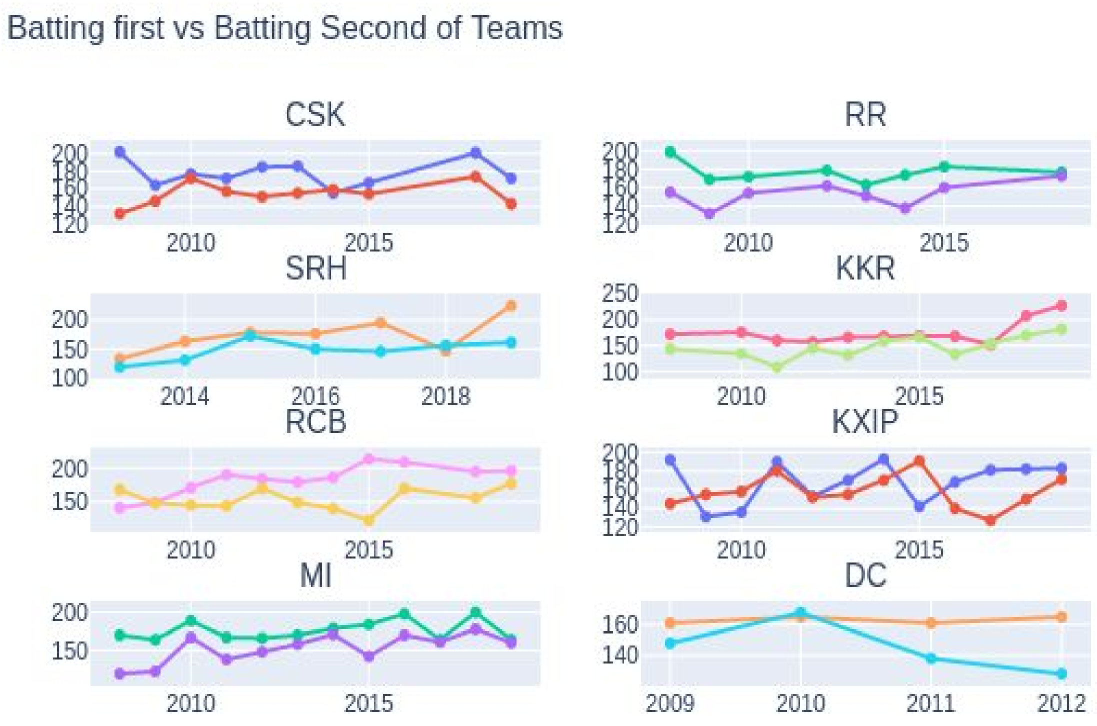

Figure 10: Average runs per team by season

8. A line chart was plotted using Plotly. This line chart indicated the average runs scored by every team in every over over all the seasons played from 2008 to 2019.

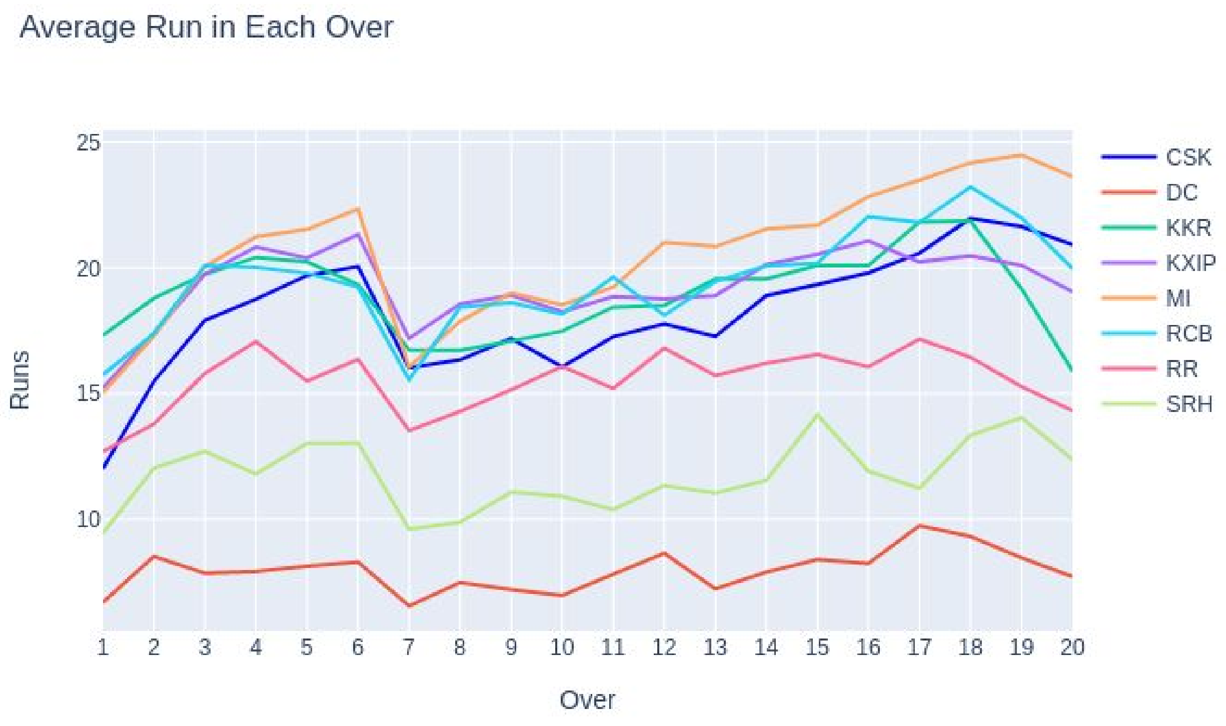

Figure 11: Average runs of team per over

9. A function was defined in python which plots the performance of the batsman whose name is provided to it as a parameter. This function plots a line chart that indicates the performance of the player over all the seasons in terms of number of runs he made in every season.

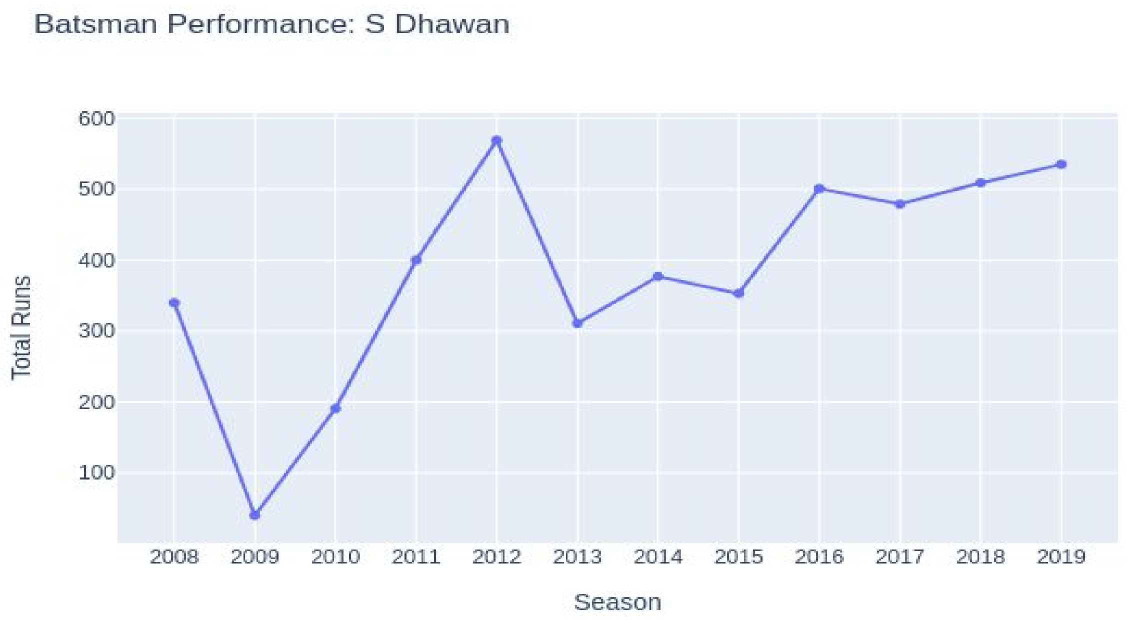

Figure 12: Batsman Performance

10. A function was defined in python which plots the performance of the bowler whose name is provided to it as a parameter. This function plots a line chart that indicates the performance of the player over all the seasons in terms of number of wickets he took in every season.

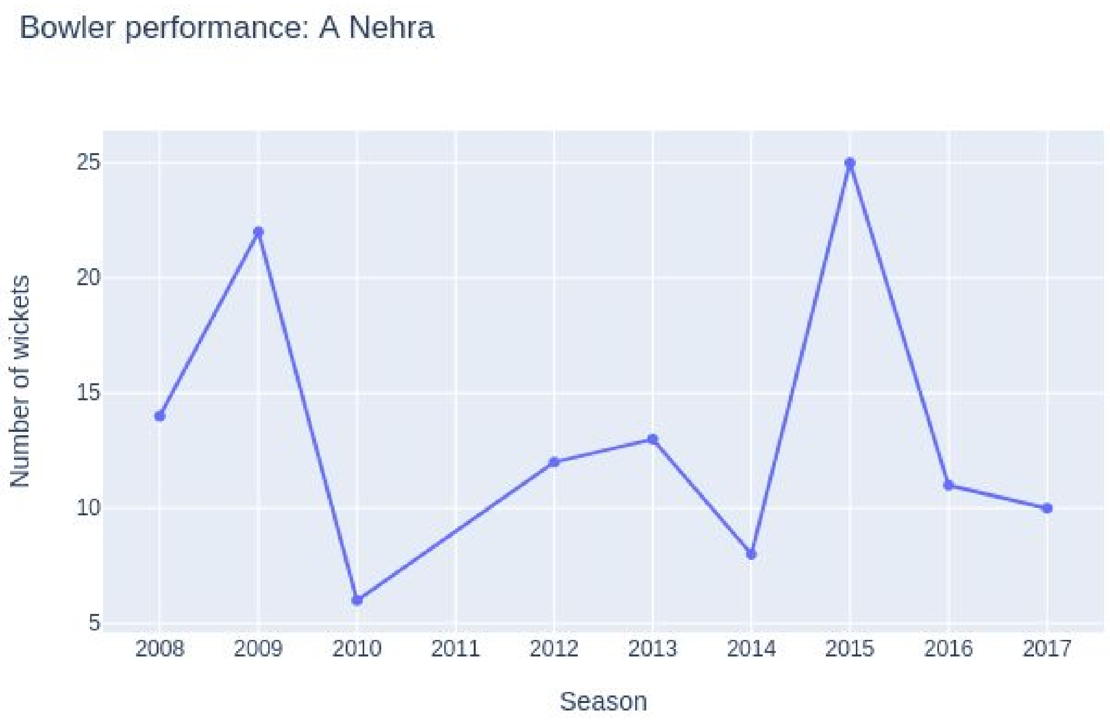

Figure 13: Bowler Performance

11. A pie chart was plotted in Plotly. This pie chart indicates the percentage of wins for every team over the seasons.

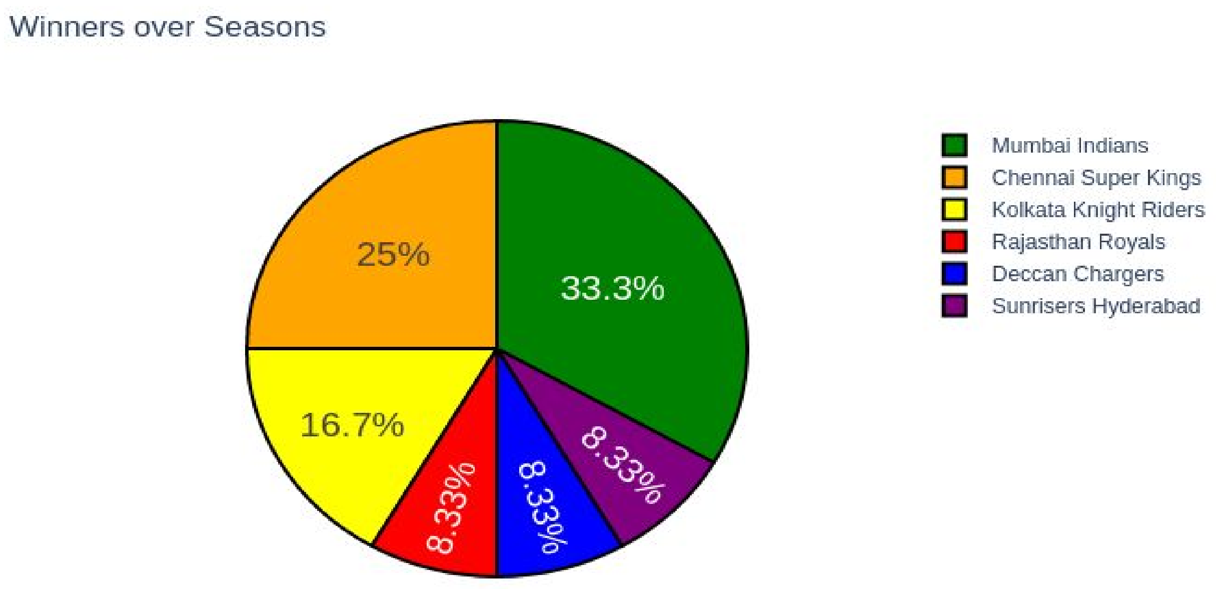

Figure 14: Winner over seasons

12. A scatter plot was plotted using Plotly. This scatter plot indicated the frequency of runs (1-6) that the batsman made over the seasons in which he was a part of.

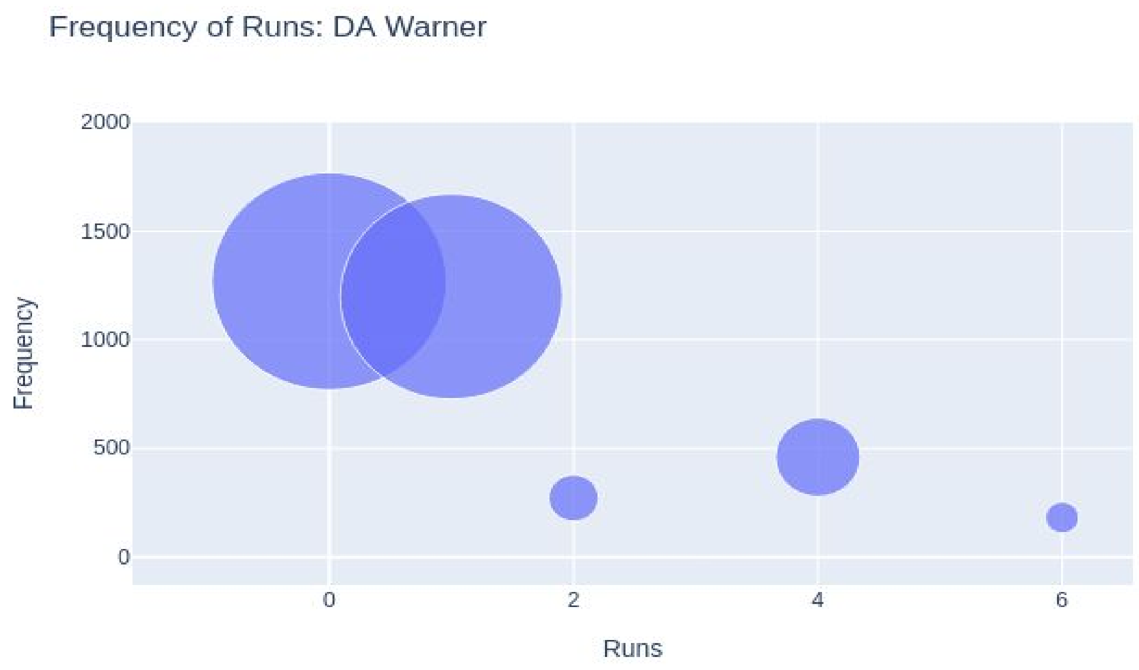

Figure 15: Frequency of runs by player

13. A function was defined in python which shows the runs made by a team whose name is provided to it as a parameter. This function plots a swarmplot in seaborn indicating the trend in which the team scored.

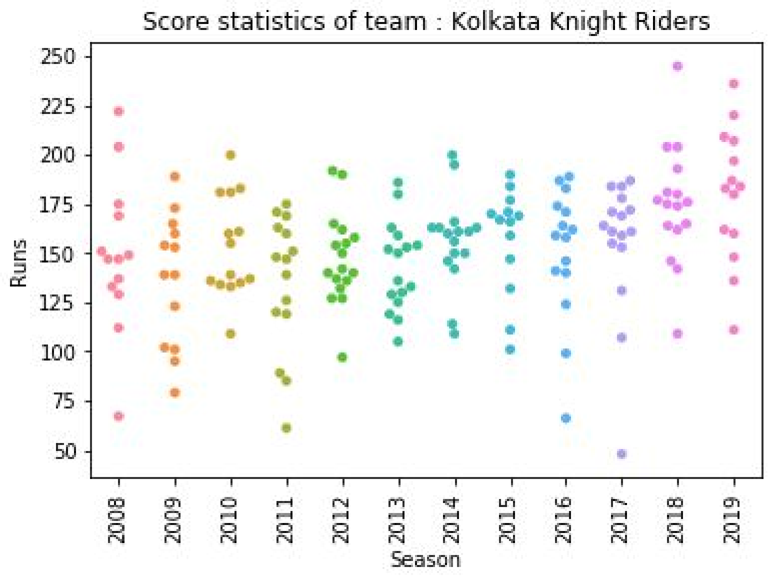

Figure 16: Store statistics per team

CHAPTER 4

OPTIMIZATION AND EVALUATION

As mentioned earlier, 3 models were built and accordingly, the one performing best on default hyperparameter values was chosen to be optimized.

Parameter	**Default Value**	**Manual Tuning**	**Grid Search Tuning**
learning_rate	0.300000012	0.36	0.4
max_depth	6	10	10
n_jobs	0	-1	-1
n_estimators	100	20	20
min_child_weight	1	1	1
Features considered	all - {id, date, umpire3}	{team1, team2, win_by_runs, player_of_match, win_by_wickets}	{team1, team2, win_by_runs, player_of_match, win_by_wickets}
Train Accuracy	100%	100%	100%
Test Accuracy	91.19%	95.59%	96.04%
Train merror	0	0	0
Test merror	0.08811	0.04405	0.03965

Table 3: Hyperparameter values comparison.

Initially, all the features were selected to train the XGBClassifier with default hyperparameter values. With that, the test distribution accuracy obtained was better than the other models, but had scope of optimization, and thus we aimed at reducing the difference between the train and test accuracies in order to reduce overfitting, if any.

So for that, the tuning was carried out manually in an iterative fashion, such that the model's performance was evaluated by picking and dropping a few particular features. In this process, a number of features were found to be unnecessary and dropped after evaluating the feature_importance graph.

Further, we still thought that we could reduce overfitting by searching for the best hyperparameter values in an organised fashion, so the grid search approach was taken to search for the best values of the important hyperparameters indicated in the table above, with cv = 3.

The model was built and fit again using the best values obtained and the accuracy not only increased, but the test merror value also decreased to 0.03965, which was still an achievement keeping in mind the merror values in the previous models and approaches. Thus the process of optimization was halted at this stage as it was realised that the XGBoost models are difficult and very time-consuming to optimize after a certain point.

CHAPTER 5

RESULTS

1.Results from the built model

Model	Train Accuracy	Test Accuracy
Decision Trees Classifier	100	77.53%
Random Forest Classifier	99.81%	56.82%
XGBClassifier	100%	96.04%

Table 4: Prediction Results Comparison

2. Results from visualization

The following results have been deduced from the visualization that was done on the aforementioned datasets:

1. The maximum number of matches were played in the year 2013 (70+), whereas the least number of matches (less than 60) were played in the years 2009 and 2015.
2. Highest number of wins (100+) were secured by Mumbai Indians and the least number of wins (less than 10) were by Kochi Tuskers Kerala over the seasons.
3. Eden Gardens, Kolkata was the stadium where the highest number of IPL matches were played.
4. Trends in the toss decision preference over the seasons were obtained.
5. Runs distribution over the years was found wherein we could distinguish the amount of runs scored by 6's, 4's and remaining.
6. Trends in the average runs scored by each team over seasons depending upon their toss win decision preference was obtained.
7. Average runs of each team per over over the seasons were obtained.
8. A graph showcasing any batsman's performance in terms of runs scored over the seasons can be obtained with the help of a user-defined python function.
9. A graph showcasing any bowler's performance in terms of wickets taken over the seasons can be obtained with the help of a user-defined python function.
10. Percentage of wins for every team over the season has been obtained in terms of a pie chart.
11. A plot showcasing the frequency of runs scored by any player can be obtained with the help of a user-defined python function.
12. Performance of any team over seasons has been obtained in terms of runs scored with the help of a user-defined python function.

CHAPTER 6

LIBRARIES AND SOFTWARE USED

Libraries used:

1. numpy
2. pandas
3. matplotlib.pyplot
4. sklearn.preprocessing.LabelEncoder
5. sklearn.ensemble.RandomForestClassifier
6. sklearn.tree
7. sklearn.model_selection.train_test_split
8. sklearn.metrics.accuracy_score
9. sklearn.model_selection.GridSearchCV
10. xgboost
11. seaborn
12. plotly

Software used:

1. Python 3.7.4
2. Jupyter Notebook 6.0.1
3. Kaggle Notebook
4. Browser: Mozilla Firefox Quantum 49.0

CHAPTER 7

CONCLUSION AND FUTURE WORKS

In our project, historical data has been collected from real IPL cricket matches and useful features have been extracted after pre-processing of data. Deciding an ideal set of attributes encourages team owners to search for players with these credits to shape a team by which they can enhance the winnability of a team. Ball by ball information for all past IPL matches were gathered and collected for the analysis and the problem was modeled as a classification problem. Data visualization was carried out to highlight 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.

As our model well predicts the IPL outcomes in the current scenario that is based on the previous records, it can be further extended in a changing environment when many new talents join the team, their records are made available. Further it can be tested by analyzing future IPL match results. Accordingly, new features can be identified and prediction can be made more accurate.

CHAPTER 8

REFERENCES

List all the material used from various sources for making this project proposals

[1] Journal article – Tejinder Singh,Vishal Singla ,"Score and winning prediction in cricket through data mining" , 2015 International Conference on Soft Computing Techniques and Implementations (ICSCTI)

[2] Journal article – Rameshwari Lokhande, P.M.Chawan, "Live Cricket Score and Winning Prediction" International Journal of Trend in Research and Development, Volume 5(1), ISSN: 2394-9333

[3] Journal article – Y.Freeund, R.Sneddon, "A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting", 1995.

[4] Books- J.D. Kelleher, Brian Mac Namee, Aoife D'Arcy . "FUNDAMENTALS OF MACHINE LEARNING FOR PREDICTIVE DATA ANALYTICS".Algorithms, Worked Examples, and Case Studies.The MIT Press.