IS 733: Data Mining

A Report on

IPL Dataset Analysis

By

Sulabh Sharma

Nihaar Ketan Shah

Shubham Manoj Heda

Under the guidance of:

James Foulds

Information Systems Department

University of Maryland, Baltimore County

Contents

1. Abstract ... 3
2. Background and Related Work ... 3
3. Introduction ... 3
4. Exploratory Data Analysis ... 5
5. Data Preprocessing ... 7
6. Support Vector Machine ... 9
7. Random forest ... 10
8. Decision tree Classifier ... 11
9. Logistic Regression ... 11
10. Metrics Used ... 12
11. Experimental Results ... 12
12. Conclusion ... 13
13. Future scope ... 14
14. References ... 14

1. Abstract

The Indian Premier League is a professional Twenty20 cricket league in India contested during march or april of every year by eight teams representing eight different cities in India. The league was founded in 2008 by the Board of Control for Cricket in India (BCCI). IPL is the most attended cricket league in the world and was ranked sixth in 2014 among all the tournaments by calculating average attendance. There have been 12 seasons of the IPL and Mumbai Indians are the current IPL title holders who won the 2019 season. Among all the thirteen teams that have played in IPL, Mumbai Indians is the most successful team with 4 IPL titles followed by Chennai Super Kings with 3 IPL titles followed by Kolkata Knight Riders with 2 IPL titles. The project goal is to implement Supervised machine learning algorithms to identify the winners of Indian Premier league and to find the accurate results based on the given features.

2. Background and Related Work

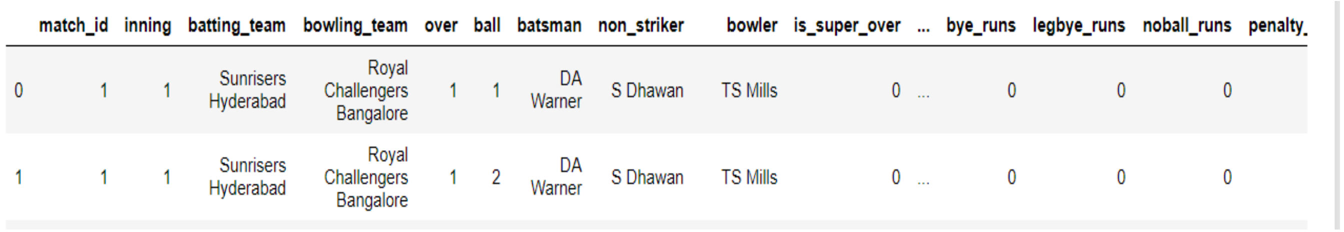
Of the many things that humans want in life, predicting the future is right at the top. Watching your favourite team win the match is blissful and satisfying. Combination of these two aspects together is quite intriguing. Sports is not about logic. It is about feelings. A six, a wicket taken or an out of the blue catch can change the emotion of the game. We as humans connect with every aspect of it. With that in mind, we plan to predict the winner of the next season of the IPL. The amount of money bet on each IPL match in India is between Rs. 6 billion and Rs. 8 billion. [1] By applying statistical methodology, we can predict the outcome of the tournament. Even though betting is illegal in India, a lot of money is spent on it. Though not for money, with the help of this project, we can predict the outcome of the match based on logic rather than gut feeling. Or for money, maybe?

3. Introduction

IPL is the heartbeat of Indian cricket. It is one of the most sought after sporting events in the year. In this project, we analyze the previous years IPL data from 2008-2019 and predict the winner of the upcoming season. The data set consists of two files viz., deliveries and matches. Deliveries file consists of ball by ball data of every ball bowled in the IPL. The matches file consists of match by match data of every match in the IPL. A glance of the IPL dataset is as follows:

Deliveries:

A glimpse of the deliveries dataset is as follows:



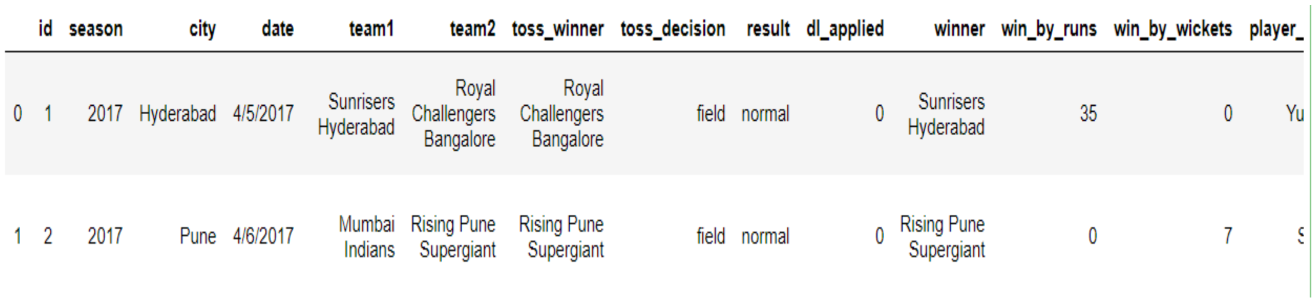
	match_id	inning	batting_team	bowling_team	over	ball	batsman	non_striker	bowler	is_super_over	...	bye_runs	legbye_runs	noball_runs	penalty_
0	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	1	DA Warner	S Dhawan	TS Mills	0	...	0	0	0	
1	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	2	DA Warner	S Dhawan	TS Mills	0	...	0	0	0	

Attributes of the deliveries dataset:

Match_id: unique ID of this dataset, **inning**: match inning, **batting_team**: batting team name, **bowling_team**: bowling team name, **over**: the current over, **ball**: the current ball of the over, **batsman**: the name of batsman on strike, **non_striker**: the name of batsman on non-striker's end, **bowler**: the name of bowler, **is_super_over**: whether a super over (0 or 1), **bye_runs**: runs given as bye, **legbye_runs**: runs given as legbyes, **noball_runs**: runs given as a no-ball, **penalty_runs:** penalty runs given, **batsman_runs**: runs scored by the batsman, **extra_runs**: extra runs, **total_runs**: total runs scored on the ball, **player_dismissed**:the name of the player dismissed (if out), **dismissal_kind**: how the player was dismissed, **fielder**: name of the fielder if involved.

Matches:

A glimpse of the matches dataset is as follows:



	id	season	city	date	team1	team2	toss_winner	toss_decision	result	dl_applied	winner	win_by_runs	win_by_wickets	player_
0	1	2017	Hyderabad	4/5/2017	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	field	normal	0	Sunrisers Hyderabad	35	0	Yu
1	2	2017	Pune	4/6/2017	Mumbai Indians	Rising Pune Supergiant	Rising Pune Supergiant	field	normal	0	Rising Pune Supergiant	0	7	S

Attributes of the matches dataset:

Id: unique match ID, **season**: match year, **city**: city where the match was played, **date**: match date, **team1**: team batting first, **team2**: team batting second, **toss_winner**: toss winner, **toss_decision**: bat or field, **result**: match result, **dl_applied**: DL rule applied (0 or 1), **winner**: match winner, **win_by_runs**: no of runs by which match was won, win_by_wickets: no of wickets by which match was won, **player_of _match**: man of the match, **umpire1**: umpire 1, **umpire2:** umpire 2, **umpire3:** umpire 3

Environment:
Languages: Python
Libraries: Pandas, matplotlib, seaborn, sklearn and numpy
IDE: Jupyter Notebook

Structure of the report:

This report will follow step by step procedure to achieve the goal of predicting the winner of the IPL. The steps taken will be as follows:

Exploratory Data Analysis: The section will visualize the data and look for evident patterns and hidden meanings in the data. This step will assist in better understanding of the dataset.

Data Preprocessing: In this section we will perform data integration, feature selection, data cleaning to yield a better dataset to train the model on. This step is quintessential to achieve good accuracy.

Data mining: After splitting the data into test and train sets, we apply different algorithms to train the model and predict the outcome. In this project we apply a random forest algorithm, logistic regression algorithm, support vector machine and decision tree classifier to achieve the goal.

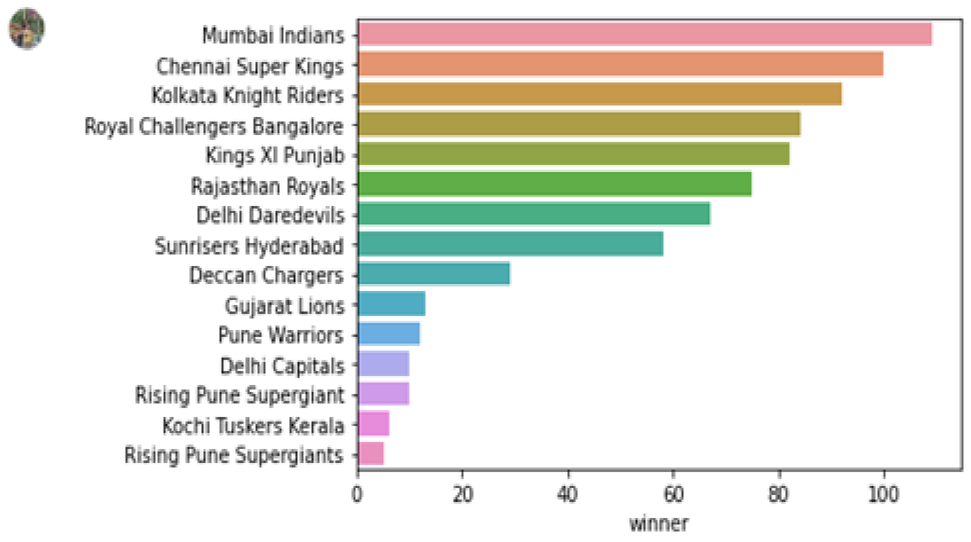
4. Exploratory Data Analysis

Exploratory data analysis is used to find conspicuous patterns, spot anomalies, provides context for further research and helps in checking assumptions regarding the data set. The exploratory data analysis of the IPL dataset is as follows:

Data Visualization:

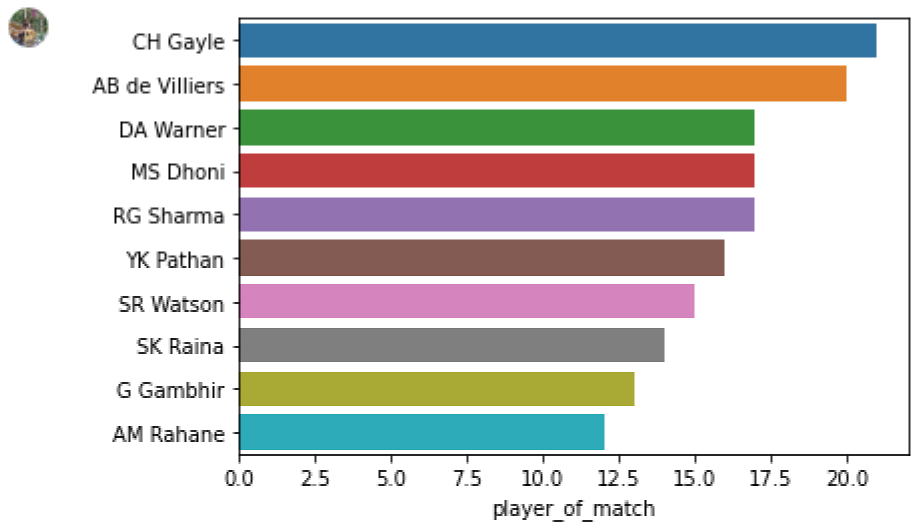
Data visualization is the graphical representation of information and data. By using visual elements like Pie charts and graphs data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data. Data visualization is an essential way to depict the performance of the data and the task performed on it. Python libraries such as matplotlib and seaborn are used with different types of visual elements. We will visualize which is the best batsman based on the player of the match, which team is the best team overall and did toss played an important role while winning the matches.

1. Winner attribute plot graph: Depending on the count of the number of wins, the graph shows which team has performed well through all the seasons.



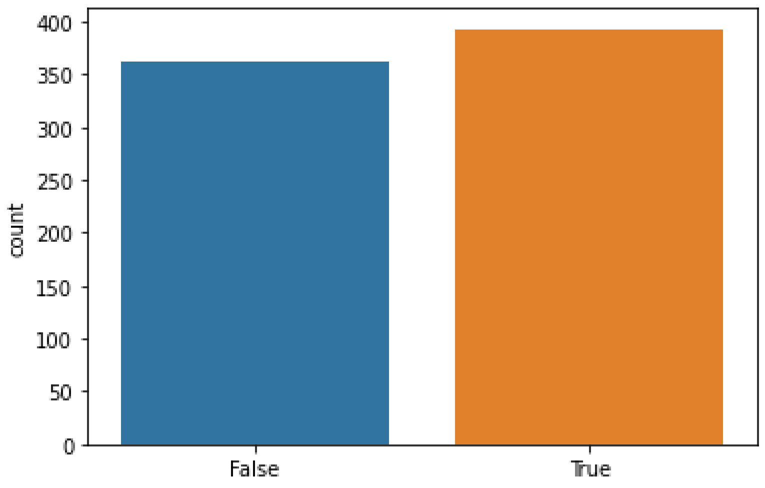
From the above graph it is observed Mumbai Indians won the maximum number of games considering every season in the visualization.

2. Player_of_match attribute: Plot graph shows the top ten count of matches played by each player.



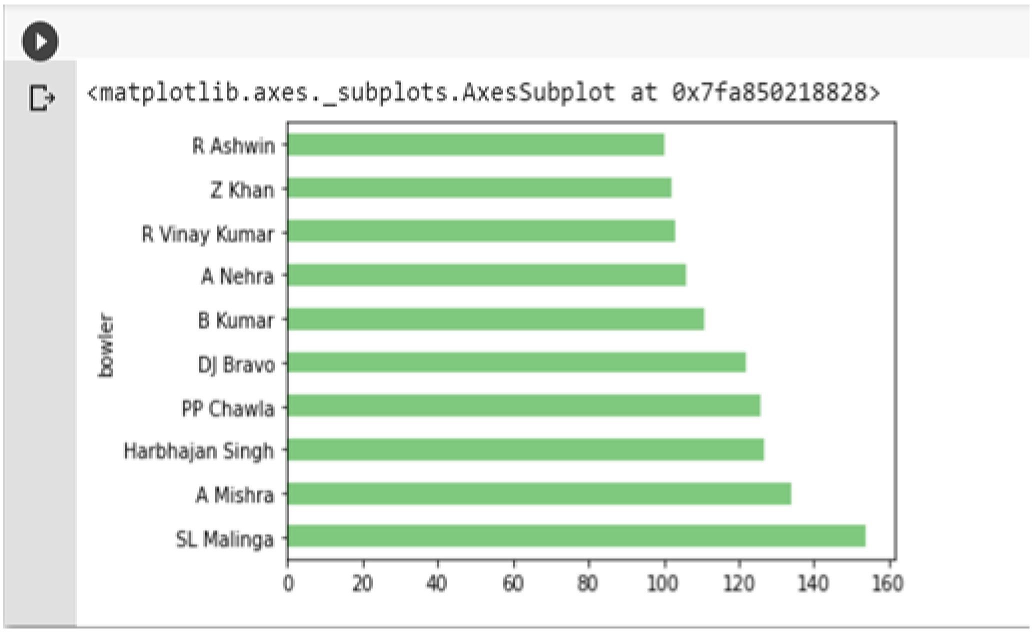
From the above graph it is observed that the best batsman after all seasons of IPL is Chris Gayle with an average of more than 20 matches in each season since the last 10 years.

3. 'Winner' and 'Toss winner': Plot graph of winner of match and the toss winner to decide if toss winning has helped in winning the matches.



From the above graph it is observed that winning the toss does have a lot of impact on the final result of the match.

4. Best Bowler: This graph shows the most successful bowler in all the seasons.



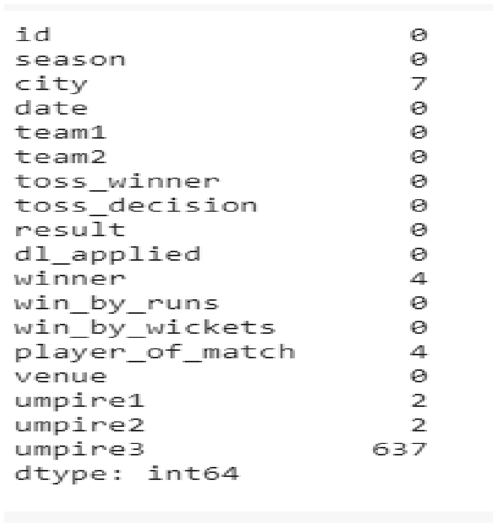
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa850218828>
```

Above graph with bowlers on y-axis and wickets on x-axis shows the top ten bowlers of all time with SL Malinga taking the maximum number of wickets till date.

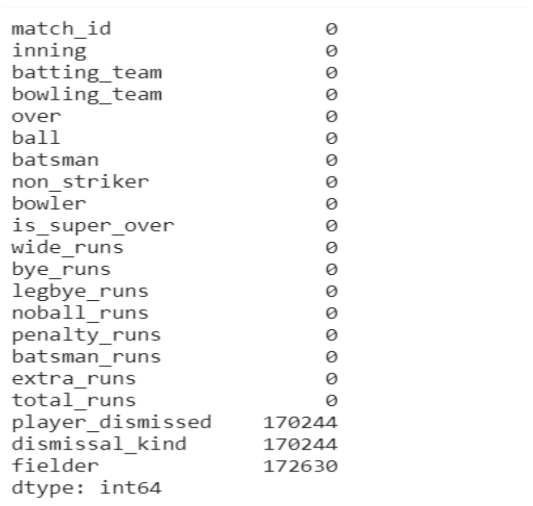
5. Data Preprocessing

Data preprocessing is probably the most important step of the process. The phrase ‘garbage in’ and ‘garbage out’ is apt for data mining. To gain insightful results, we need to provide a clean dataset to the model. The better the result, higher are the chances of getting great results.We will perform the following data preprocessing steps:

1. Check for missing values and fill them: In our project, we are checking for any missing data. Using python we are using the function isna() which checks for any missing data in our data file. In the dataset, seven ‘city’ values are missing which are filled by ‘Dubai’ and ‘umpire1’ by ‘Aleem Dar’. Attributes which seem not to contribute to the final goal are dropped.

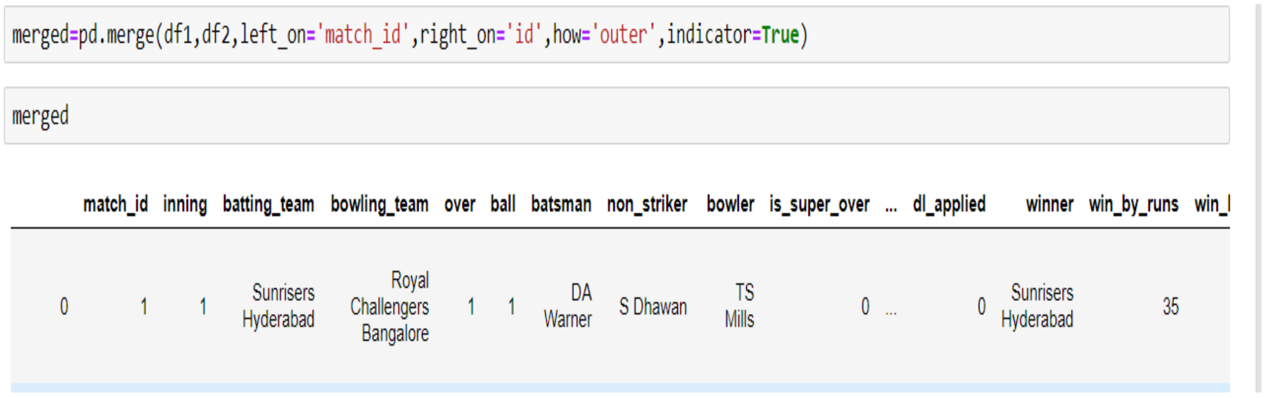


```
id                  0
season              0
city                7
date                0
team1               0
team2               0
toss_winner         0
toss_decision       0
result              0
dl_applied          0
winner              4
win_by_runs         0
win_by_wickets      0
player_of_match     4
venue               0
umpire1             2
umpire2             2
umpire3           637
dtype: int64
```



```
match_id                0
inning                  0
batting_team            0
bowling_team            0
over                    0
ball                    0
batsman                 0
non_striker             0
bowler                  0
is_super_over           0
wide_runs               0
bye_runs                0
legbye_runs             0
noball_runs             0
penalty_runs            0
batsman_runs            0
extra_runs              0
total_runs              0
player_dismissed   170244
dismissal_kind     170244
fielder            172630
dtype: int64
```

2. Merge datasets: To unearth patterns and meaning from the data, it is essential to merge the two datasets. We merged the deliveries and the matches datasets by merging the match_id column in the delivervies dataset and id in the matches dataset.

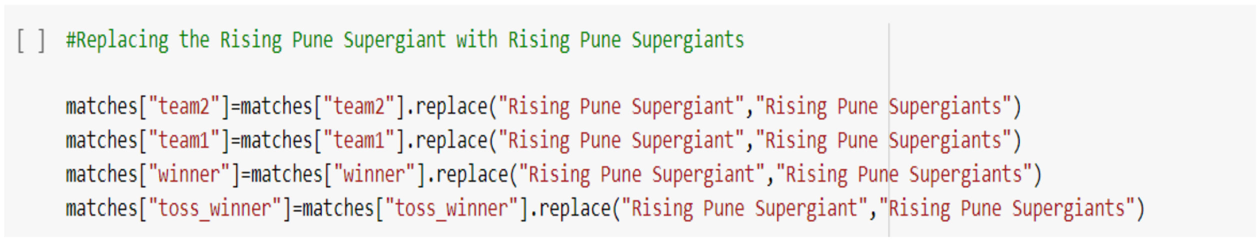


```
merged=pd.merge(df1,df2,left_on='match_id',right_on='id',how='outer',indicator=True)
```

```
merged
```

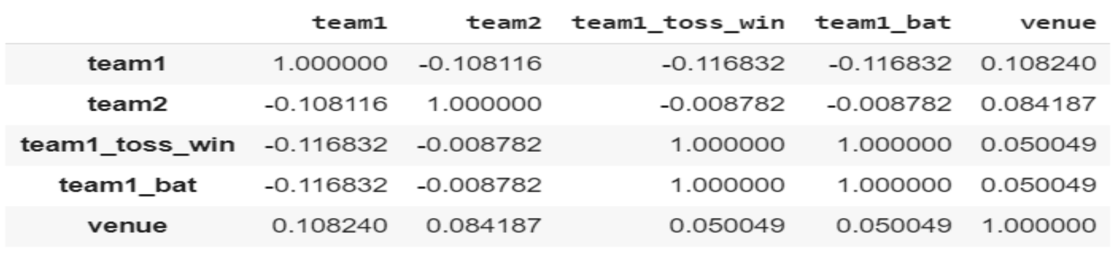
	match_id	inning	batting_team	bowling_team	over	ball	batsman	non_striker	bowler	is_super_over	...	dl_applied	winner	win_by_runs	win_l
0	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	1	DA Warner	S Dhawan	TS Mills	0	...	0	Sunrisers Hyderabad	35	

3. Replace values: For better performance of the models, venues of different stadiums in the same city are replaced by the city name as that is their home ground. 'Maharashtra Cricket Association Stadium' and 'Wankhede Stadium' is replaced by 'Mumbai'. Also, in this dataset there is an anomaly wherein the same team name is represented differently. 'Rising Pune Supergiant' and 'Rising Pune Supergiants' are the same team however because of a spelling error, the model will be considered to be two different teams. To avoid that, we replaced 'Rising Pune Supergiant' with 'Rising Pune Supergiants' to ascertain consistency in all the attributes of the dataset.



```
[ ]  #Replacing the Rising Pune Supergiant with Rising Pune Supergiants

     matches["team2"]=matches["team2"].replace("Rising Pune Supergiant","Rising Pune Supergiants")
     matches["team1"]=matches["team1"].replace("Rising Pune Supergiant","Rising Pune Supergiants")
     matches["winner"]=matches["winner"].replace("Rising Pune Supergiant","Rising Pune Supergiants")
     matches["toss_winner"]=matches["toss_winner"].replace("Rising Pune Supergiant","Rising Pune Supergiants")
```

4. Feature selection: The merged dataset has thirty nine attributes in all. Selection of the right attributes is one of the most important aspects of this project. We narrow down the selection of attributes from thirty nine to six. We then create a correlation matrix and find out the relationship between these attributes. From the correlation matrix, we realize that whenever team 1 wins the toss, they opt to bat. Hence, we drop the team1_toss_win attribute for keeping it increases redundancy. The features selected are team1, team2, team1_bat, team1_win and venue.



	team1	team2	team1_toss_win	team1_bat	venue
team1	1.000000	-0.108116	-0.116832	-0.116832	0.108240
team2	-0.108116	1.000000	-0.008782	-0.008782	0.084187
team1_toss_win	-0.116832	-0.008782	1.000000	1.000000	0.050049
team1_bat	-0.116832	-0.008782	1.000000	1.000000	0.050049
venue	0.108240	0.084187	0.050049	0.050049	1.000000

5. Handling categorical values: In order to perform different algorithms on the dataset, team names are assigned initials and each initial is assigned a number since all columns cannot be used. These values are then encoded and used in prediction.

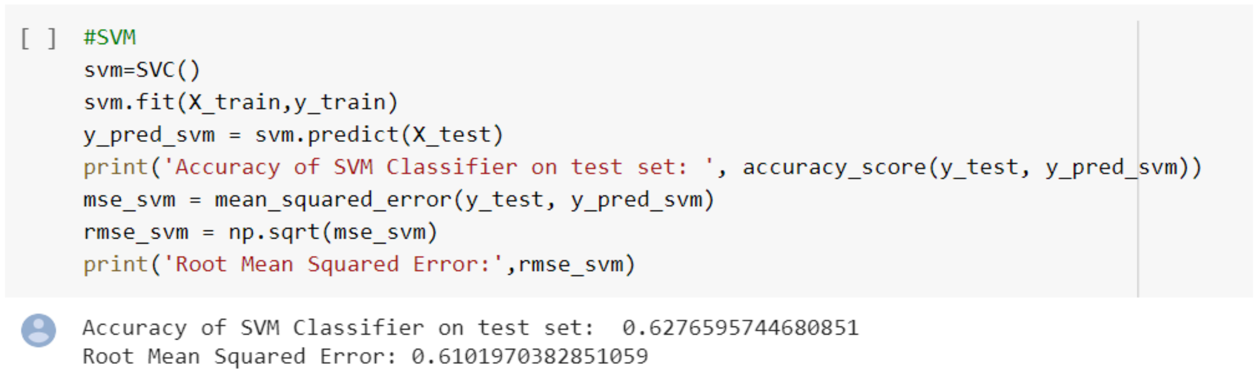
6. Splitting data: In order to create a model that generalizes the new model well, we split the dataset into 75% train data and 25% test data. The train data is used to train the data and the test data is used to test the trained data. With the test data, we predict the output which is then compared with observed values.

6. Support Vector Machine

Support Vector Machine is a supervised learning technique in Machine Learning used for classification, regression and outliers detection. SVM have the main objective of finding hyperplanes in an N-dimensional space that distinctly classifies the data points. To divide all the data points there are many possible hyperplanes that can be used. But our objective is to find a plane that has a maximum margin that is maximum distance between data points of both classes. Maximizing the margin will lead to classification of data points with more confidence in future. Hyperplanes are decision boundaries used to classify data points on both sides of the margin. The data points that lie on the hyperplane are called support vectors.

The root mean square error (RMSE) is a standard deviation of the residuals. Residuals are a measure of how far from the regression line data points are, it tells you how concentrated the data is around the line of best fit. In our case the RMSE score is about 61.01% Model performance is estimated in terms of its accuracy to predict the occurrence of an event on unseen data. A more accurate model is seen as a more valuable model. The RMSE score is used to evaluate accuracy.

In this project we took 75% of train data and 25% of test data and got the following results that are given below. We performed SVM between the observed values and the predicted values and got the accuracy of **62.76%. Root Mean Squared Error** is the standard deviation of the error and for this dataset we got the value of it as **61.01%**.



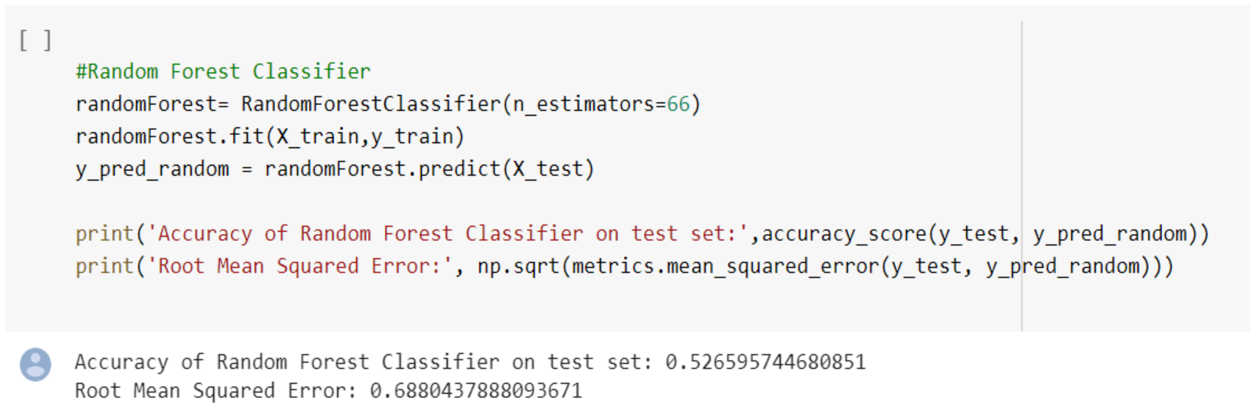
```
[ ]  #SVM
     svm=SVC()
     svm.fit(X_train,y_train)
     y_pred_svm = svm.predict(X_test)
     print('Accuracy of SVM Classifier on test set: ', accuracy_score(y_test, y_pred_svm))
     mse_svm = mean_squared_error(y_test, y_pred_svm)
     rmse_svm = np.sqrt(mse_svm)
     print('Root Mean Squared Error:',rmse_svm)
```

```
Accuracy of SVM Classifier on test set:  0.6276595744680851
Root Mean Squared Error: 0.6101970382851059
```

7. Random forest

Random forest is an ensemble learning method for regression and classification. Random forest operates by including multiple decision trees. A decision tree consists of a set of questions to be answered to predict the class label. Random forest not only includes one decision tree but a set of decision trees. If class is categorical the final output will be most class values predicted by all decision trees and if the class is continuous value the final output will be the mean of the values of all the decision trees. In our case we are predicting the match winner which is a continuous value. We have applied a random forest regression model by first training the model with 75% training data and predicted the purchase amount for 25% test data.

Similarly, like Support Vector Machine we have implemented different metrics like accuracy and RMSE. We performed Random Forest between the observed values and the predicted values and got the accuracy of **52.65%. Root Mean Squared Error** is the standard deviation of the error and for this dataset we got the value of it as **68.80%**.



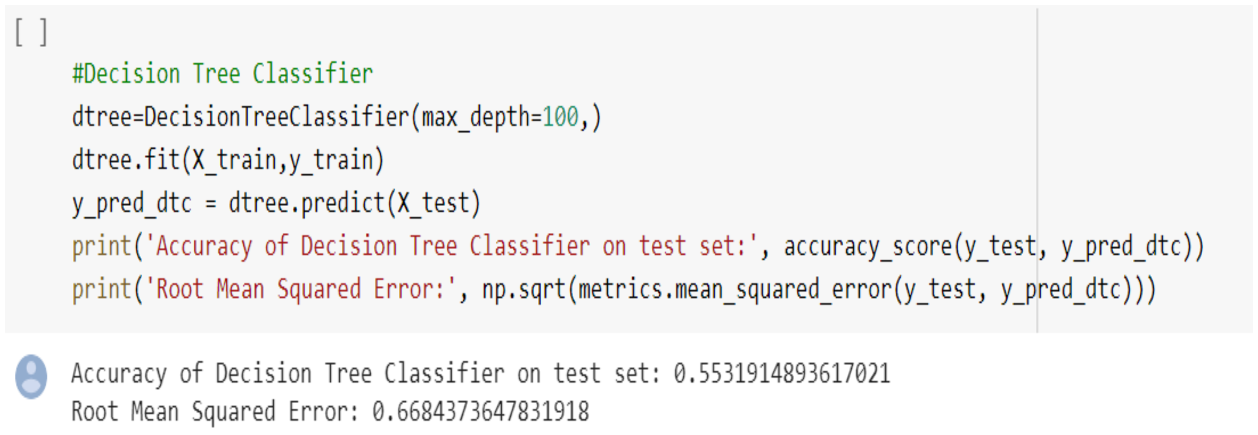
```
[ ]
     #Random Forest Classifier
     randomForest= RandomForestClassifier(n_estimators=66)
     randomForest.fit(X_train,y_train)
     y_pred_random = randomForest.predict(X_test)

     print('Accuracy of Random Forest Classifier on test set:',accuracy_score(y_test, y_pred_random))
     print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred_random)))
```

```
Accuracy of Random Forest Classifier on test set: 0.526595744680851
Root Mean Squared Error: 0.6880437888093671
```

8. Decision tree Classifier

Decision tree Classifier Observes features of an object and trains a model in the structure of a tree to predict data in the future to produce meaningful continuous output. For the decision tree we have used the best parameter max_depth = 100 and criterion = 'gini'. Gini Impurity is a measure of the probability that a new instance of a random variable will be labeled incorrectly if that new instance is randomly labeled according to the class mark distribution from the data set.

Similarly, like SVM and Random forest classifier we have implemented different metrics like accuracy and RMSE in the Decision tree classifier. We performed Decision Tree Classifier between the observed values and the predicted values and got the accuracy of **55.31%. Root Mean Squared Error** is the standard deviation of the error and for this dataset we got the value of it as **66.84%**.



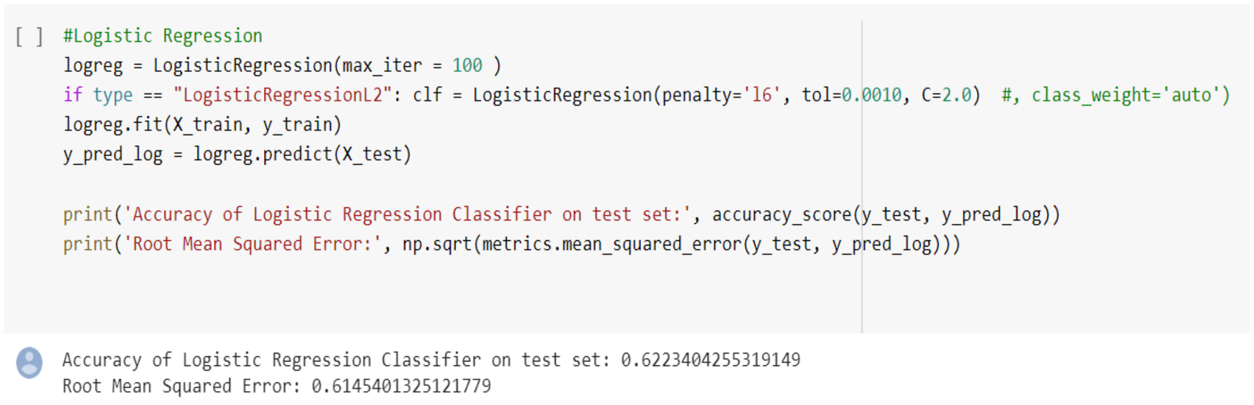
```
[ ]
    #Decision Tree Classifier
    dtree=DecisionTreeClassifier(max_depth=100,)
    dtree.fit(X_train,y_train)
    y_pred_dtc = dtree.predict(X_test)
    print('Accuracy of Decision Tree Classifier on test set:', accuracy_score(y_test, y_pred_dtc))
    print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred_dtc)))
```

```
Accuracy of Decision Tree Classifier on test set: 0.5531914893617021
Root Mean Squared Error: 0.6684373647831918
```

9. Logistic Regression

Logistic Regression is the appropriate regression analysis that is to be conducted when the dependent variable is binary. Like all regression analyses, the logistic regression is a predictive analysis. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.

Similarly, like other classifiers that we have used so far, we have implemented different metrics like accuracy and RMSE in the Logistic Regression. We performed Logistic Regression between the observed values and the predicted values and got the accuracy of **62.23%. Root Mean Squared Error** is the standard deviation of the error and for this dataset we got the value of it as **61.45%**.



```
[ ] #Logistic Regression
    logreg = LogisticRegression(max_iter = 100 )
    if type == "LogisticRegressionL2": clf = LogisticRegression(penalty='l6', tol=0.0010, C=2.0)  #, class_weight='auto')
    logreg.fit(X_train, y_train)
    y_pred_log = logreg.predict(X_test)

    print('Accuracy of Logistic Regression Classifier on test set:', accuracy_score(y_test, y_pred_log))
    print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred_log)))
```

```
Accuracy of Logistic Regression Classifier on test set: 0.6223404255319149
Root Mean Squared Error: 0.6145401325121779
```

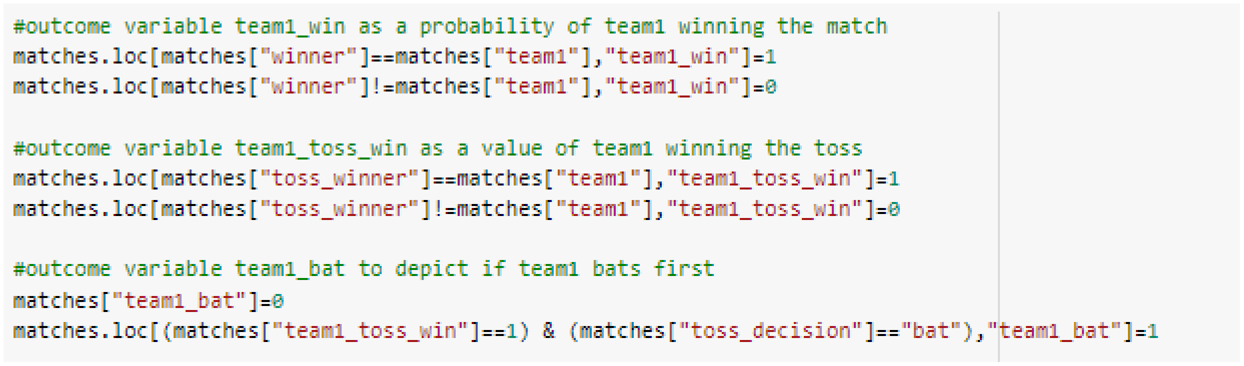
10. Metrics Used

The metrics that we have used is Root Mean Square Error and Accuracy.

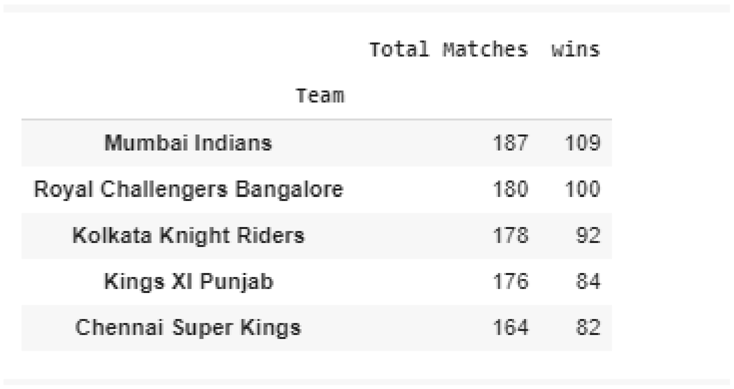
The root means square error (RMSE) is a standard deviation of the residuals. Residuals are a measure of how far from the regression line data points are, it tells you how concentrated the data is around the line of best fit. The model performance is estimated in terms of its accuracy to predict the occurrence of an event on unseen data. A more accurate model is seen as a more valuable model. The RMSE score is used to evaluate accuracy.

11. Experimental Results

Using our model we Predicted winners of IPL 2020 , based on winning percentage which we calculated after getting the total number of wins by individual teams in total IPL seasons ever played till year 2019 and then calculating individual wins based on different variables as mentioned in the below image.

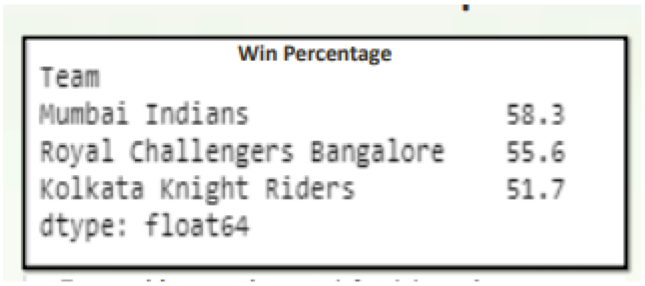


```
#outcome variable team1_win as a probability of team1 winning the match
matches.loc[matches["winner"]==matches["team1"],"team1_win"]=1
matches.loc[matches["winner"]!=matches["team1"],"team1_win"]=0

#outcome variable team1_toss_win as a value of team1 winning the toss
matches.loc[matches["toss_winner"]==matches["team1"],"team1_toss_win"]=1
matches.loc[matches["toss_winner"]!=matches["team1"],"team1_toss_win"]=0

#outcome variable team1_bat to depict if team1 bats first
matches["team1_bat"]=0
matches.loc[(matches["team1_toss_win"]==1) & (matches["toss_decision"]=="bat"),"team1_bat"]=1
```

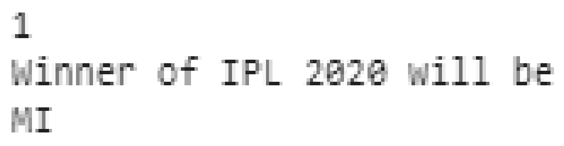


Team	Total Matches	wins
Mumbai Indians	187	109
Royal Challengers Bangalore	180	100
Kolkata Knight Riders	178	92
Kings XI Punjab	176	84
Chennai Super Kings	164	82

We calculated the winner of IPL season 2020 which is Mumbai Indians as they have won the maximum number of matches and their win percentage is the maximum . In the below images we can see the teams with the highest win percentage, which plays important role in predicting the winner of IPL 2020.



```
Win Percentage
Team
Mumbai Indians                 58.3
Royal Challengers Bangalore    55.6
Kolkata Knight Riders          51.7
dtype: float64
```



```
1
Winner of IPL 2020 will be
MI
```

12. Conclusion

Selection of the best team for a cricket match plays a critical job for the group's triumph. The principal objective of this paper is to break down the IPL cricket information and anticipate the players' presentation and foresee IPL champion for year 2020. This project has intended on analyzing the results of the IPL match during the year 2008-2019 by applying the data mining algorithms on both the balanced as well as imbalanced dataset. The outcome values are higher than the imbalanced dataset and errors are lesser. Here, four classification algorithms are used and compared to find the best accurate algorithm. This knowledge will be used in future to predict the winning teams for the next series IPL matches when data like players, weather and ICC rankings will be taken into consideration. Hence using this prediction, the best IPL team can be predicted.

- The model which is used to analyze the results of matches was built successfully with a support vector machine having the highest accuracy rate of 62.76% for the balanced dataset using the classifiers (i.e) after oversampling the imbalanced IPL dataset .
- The Random Forest model can be used to predict class efficiently as it not only depends on the outcome of the individual decision tree but considers the outcomes of multiple decision trees. Comparing the root mean square error of the four algorithms Random Forest is found to have less root mean square error.
- Team 1, Team 2, Venue, Team1_bat, Team1_win and team1_toss_win were observed to be the highest contributors to predict IPL 2020 winners.

13. Future scope

- Once teams are finalized, batting and bowling forms of players in international cricket and the IPL format can be considered in prediction of the tournament.
- Factors such as the weather condition, pitch condition and past performances can be linked to the outcome of the result.
- Home game crowd advantage, team balance, the umpire and experience in certain situations can assist in improving the accuracy of the model.

14. References

1. https://www.rediff.com/cricket/report/bookies-have-rs47000-crores-riding-on-ipl-6/20130516.htm
2. https://www.kaggle.com/nowke9/ipldata#deliveries.csv
3. https://www.statisticssolutions.com/what-is-logistic-regression/