Predicting the winner of IPL matches using Machine Learning

Parth Agrawal, Tanishq Nandan, Pragalbh Dev Singh, and Abhigyan Utsav

BITS Pilani, Goa Campus

November 18, 2020

Abstract

Indian Premier League, or IPL is one of the most popular T20 leagues in the world. Millions of people try to estimate the outcome of each IPL match, form the best 11 for the day, performance against a particular team, results on a particular field, etc. before the match starts.Apps like Dream 11 use a predefined method of rating to evaluate player performances. In this research, we explore the importance of the various factors which are considered while making these predictions like history of the team, strength of players, etc. In this paper, we create a dataset containing the pre-match information and outcomes of all the IPL matches played till the end of the 2019 season, along with some custom features added by us. We then train a variety of models, ranging from logistic regression to Neural Networks on that dataset and compare their results based on f1 score. We also look at the different ways to deal with less amounts of data, and to modify the dataset based on the assumptions used by each model.

Keywords : Sports Analysis, Cricket Analysis, IPL Analysis, Machine Learning, SVM, Neural Network, Data Mining

1 Introduction

Cricket is a sport with huge popularity and following in India. The impact of cricket in India has been immense. Out of the three formats of cricket, we chose T20 as it has maximum viewer engagement. T20 is a growing area within cricket which is receiving attention from all over the globe. As the T20 format is only 17 years old, the coaches and the teams should be engaged in learning the dynamics of the format for better decision making. A major decision to be made was to select the tournament. Since, we wanted to predict match outcomes, we needed some tournament held at least annually with all match data readily available. This led us to the Indian Premier League, since it is currently the most-famous T20 league, and an extremely popular tournament with huge betting capital. Since it is auction based, the results here could also be used to predict which combination of 11 players will be the best.

2 Related Works

Previous works in this domain mostly focus on individual players and not on the entire matches' outcome. Iyer and Sharda[1] classified cricket players into 3 categories -performer,moderate and average using neural networks on the past performance of the players. Using the classification as a basis they recommended if a player should be selected for 2007 WC or not. Jhanwar and Pudi[2] predicted the outcome of the cricket match by using the past record of each player in the team.Parker Burns and Natarajan[3] built a machine learning model to predict the valuation of players in IPL auction.They used features like previous bidding price,experience,strike rate etc.Rabindra Lamsal and Ayesha Choudhary[4] used python machine learning libraries to predict the outcome of an IPL match in one season by using some key features.The key features were selected from the data of the past seasons of the IPL. Abhishek Naik, Shivanee Pawar, Minakshee Naik, Sahil Mulani[6] used K-means clustering and logistic regression to dynamically predict the outcome of a cricket match.Their model used the team level statistics before the match began and after that it used the scoreboard of both teams to predict the outcome.Singhvi, Arjun, Ashish Shenoy, Shruthi Racha and Srinivas Tunuguntla[5] took data from all T20 matches irrespective of the tournament and used Random Forest,Naive Bayes, Decision Trees, Linear SVM, Non-Linear SVM.They found that Support Vector Machines gave the best accuracy amongst all the machine learning algorithms.E. Geddam Jaishankar Harshit,Rajkumar S[6] in their paper compared the accuracy of Support Vector Machine,Logistic Regression, Decision Tree and Bayes Classifier in predicting the outcome of cricket matches,for this they took dataset of 5000 ODI matches and used 70% of that for training the model. They found that Bayes Classifier had the best accuracy(72%). In all the research papers that we read the accuracy of predicting the outcome of cricket matches was always in the ballpark of 60% to 70%.

3 Dataset

3.1 Data Collection

We collected the data for previous years' IPL tournaments by scraping data from espncricinfo and cricbuzz websites. The packages used for this purpose were Scrapy and Beautiful Soup . We chose these sites as their page designs were relatively simpler and they contained the data for all seasons in a systematic manner, so we could iterate over the URLs to collect the data. We used cricbuzz to get information about matches like squad lists, venue, winner etc and then espncricinfo to get the scoreboards of those matches.

We did face quite a few problems such as inconsistent team and player names(V Kohli in scoreboard and Virat Kohli in squad list, Delhi Daredevils vs Capitals, and some more), different page designs in some places(class attribute values changed for certain matches), and instances like abandoned matches. We had to take care of these cases manually(making dictionaries to convert names, changing the site attributes wherever it was different, and deleting the abandoned match instances). We scraped and checked the data for each year manually, before finally compiling and storing it in 2 csv files, containing the match wise data and scoreboard of every match respectively.

3.2 Features Generated

After scraping all this data, the next step was to construct more features from the existing relations and create a final database on which we can train and test our models. We constructed 13 features for each team, 7 based on the team data and 6 based on the scoreboard and each player's individual stats.

3.2.1 Team Level Stats

Every team has a brand value associated with it. Win percentage prediction of a match of Chennai Super Kings vs Mumbai Indians might not be too inclined towards one team while in a match of a new team against any of these(say Chennai Super Kings vs Kochi Tuskers), it should be expected to lean in the favor of the experienced team. Some players are constant in a team, especially the captains who don't change each year. Based on these intuitions, we incorporated the following variables:

$$WinPercentage = MatchesWon/MatchesPlayed \tag{1}$$

$$WonWhenTossWonPercentage = \frac{MatchesWonAndTossWon}{MatchesTossWon} \tag{2}$$

$$WonWhenTossLostPercentage = \frac{MatchesWonAndTossLost}{MatchesTossLost} \tag{3}$$

$$WonBattingFirstPercentage = \frac{MatchesWonBattingFirst}{MatchesBattedFirst} \tag{4}$$

$$WonChasingPercentage = \frac{MatchesWonChasing}{MatchesBattedSecond} \tag{5}$$

$$TeamBalance = \sum e^{-(batCon-bowlCon)^2)} \ over\ all\ players \tag{6}$$

The last feature measures how balanced the team is. Thus, it will penalize teams with too many batsmen and too few bowlers, or vice versa. As the prediction for the outcome of a match is to be made, and not for the purpose of a fantasy team, this feature becomes important. We also added a Form index, which takes values of 0,5 or 10 depending on whether the team won its last 0,1 or 2 matches. So, a team would ideally want to go into a match with a winning streak than a losing one

3.2.2 Player Level Stats

Since players change every year, just the team stats aren't enough to represent a team entirely. We have included separate bowling and batting attributes for each player whose aggregate gives the bowling and batting strength of the team. This can help us in storing and retrieving a player's stats even if he joins a different team next year. All the attributes are aggregated over the IPL career of the player till the season we are predicting.

- Consistency: These attributes capture the player's experience/ how consistent his performance has been throughout his IPL career despite being in different teams.

BatConsistency = 0.51*StrikeRate + 0.25* HardHit *100 + 0.155 * fifties + 0.056* centuries - 0.049* zeroes *50

The weights were determined using the analytical hierarchy process.[8]
The analytic hierarchy process decomposes the decision making process in following steps: 1. Define the problem and the knowledge sought. 2. Structure the decision hierarchy with the goal at the top level, objectives/attributes at the intermediate levels and alternatives at the lowest level. 3. Construct a set of pairwise comparison matrices where each element in an upper level is compared to the elements in the level immediately below it. These comparisons are made using a scale of numbers which indicates how many times more important is one element over another. This scale is tabulated in table below. 4. The priorities obtained from the comparisons are used to weigh the priorities in the level immediately below. This is done for every element. The overall or global priority for every element in the level below is obtained by adding its weighted values. This process is continued until priorities of the alternatives in the lowest level obtained. The weights are calculated using the following mathematical operations.

Level of Importance	Meaning	Description
1	Equal Importance	Two activities contribute equally to the objective.
2	Weak or Slight	
3	Moderate Importance	Experience and judgement slightly favor one activity over another.
4	Moderate Plus	
5	Strong Importance	Experience and judgement strongly favor one activity over another
6	Strong Plus	
7	Very strong or demonstrated importance	An activity is favored very strongly over another; its dominance demonstrated in practice
8	Very, very strong	
9	Extreme importance	The evidence favoring one activity over another

		is of the highest possible order of affirmation
Reciprocals of the above	If activity *i* has one of the above non-zero numbers assigned to it when compared with activity *j*, then *j* has the reciprocal value when compared with *i*	A reasonable assumption
1.1 – 1.9	If the activities are very close	May be difficult to assign the best value but when compared with other contrasting activities the size of the small numbers would not be too noticeable, yet they can still indicate the relative importance of the activities.

Here is the calculation for weights used in calculating consistency We took the order based on our cricket knowledge in IPL as StrikeRate, then hardHit, then 50s, then 100s, then zeros
'hardHit' and '0s' were multiplied with 100 and 50 respectively to bring the parameters to the same range Next, we created a matrix of priorities refering the table shown above. Then we calculated priority using the formula : $P_j = \left(\prod_1^N p_{ij}\right)^{1/N}$ Pj is the priority of attribute j, N is the number of attributes and pij is the level of importance of attribute j over attribute i. Next, we normalize each attribute's priority using the following formula: $W_j = \frac{P_j}{\sum_1^N P_i}$ Final weights were :

	strike rate	HardHit	Fifties	centuries	zeros
strike rate	1	4	4	5	6
HardHit	0.25	1	3	4	5
Fifties	0.25	0.333333	1	4	5
centuries	0.2	0.25	0.25	1	1
zeros	0.166667	0.2	0.2	1	1
weights	0.508	0.247781	0.155132	0.056632	0.049872

BowlConsistency = 0.2* OversBowled + 0.3* 50 / (BWA + 1) + 0.18* 50 / (SR + 1) + 0.05*4Haul + 0.02*5Haul*5 + 0.1*Maidens*6 + 0.05*20/(Econ+1)

The weights were again determined using an analytical hierarchy process. Order considered for the same was OversBowled, then BWA, then SR, then 4Haul, then 5Haul, then Maidens, then Econ.

- PCA : Higher values indicate better performance for the batsman. The PCA Index for batsmen assigns uniform weights to average and total number of runs scored through the IPL career, fours, sixes, centuries

or half centuries and strike rate. Since the range of weights is quite small, the relative impact of the indices is not too prominent here. The absolute values of the metrics are widely different i.e. a player could have scored many boundaries whereas the number of centuries and half centuries would be relatively very small.
PCA (batsmen) = 0.398 * PBA + 0.325 *SR+ 0.417 * Sixes + 0.406 * Fours + 0.432 * 50's+0.458 *RunsScored

Higher values indicate better performance for the batsman.

PCA (bowlers) = 0.591 * bowlAvg + 0.383 *Economyrate + 0.566 * BowlSR-0.428 * Wickets Taken

Smaller the PCA value, better the bowler.

- HardHit :

$$HardHit = \frac{NoOfBoundaries}{NoOfBallsFaced} \tag{7}$$

IPL is a T20 format game so a player constantly scoring more boundaries is more valuable for the team.

- Bowling Index : It captures the 3 main attributes of a bowler : Average + SR + Economy

3.3 Preprocessing

1. Assigning Relevance : More relevance was given to recent data. For this, we calculated the stats after a season as :
Post(Year)Stats = 0.6*(Year)Stats + 0.4*Post(Year-1)Stats
e.g. : Post2011Win = 0.6*2011Win + 0.4 * Post2010Win
2. The winner was set as 1 if Team1 won the match, and 0 if Team2 won the match.
3. Irrelevant columns like Umpire names, Referee Names were removed from the dataset.
4. The venue names were converted to Home/Away. 1 if it is Team1's home ground, 0 if Team2's home ground and 2 if neither.

So, in summary, the features generated by us were :

Column Name	Feature
Team1_A	win(%)
Team1_B	won when toss won(%)
Team1_C	won when toss lost(%)
Team1_D	won when batting(%)
Team1_E	won when chasing(%)
Team1_F	hard hitting
Team1_G	batting consistency
Team1_H	batting pca
Team1_I	bowling index
Team1_J	bowling consistency
Team1_K	bowling pca
(Same for Team2)	

Table 1

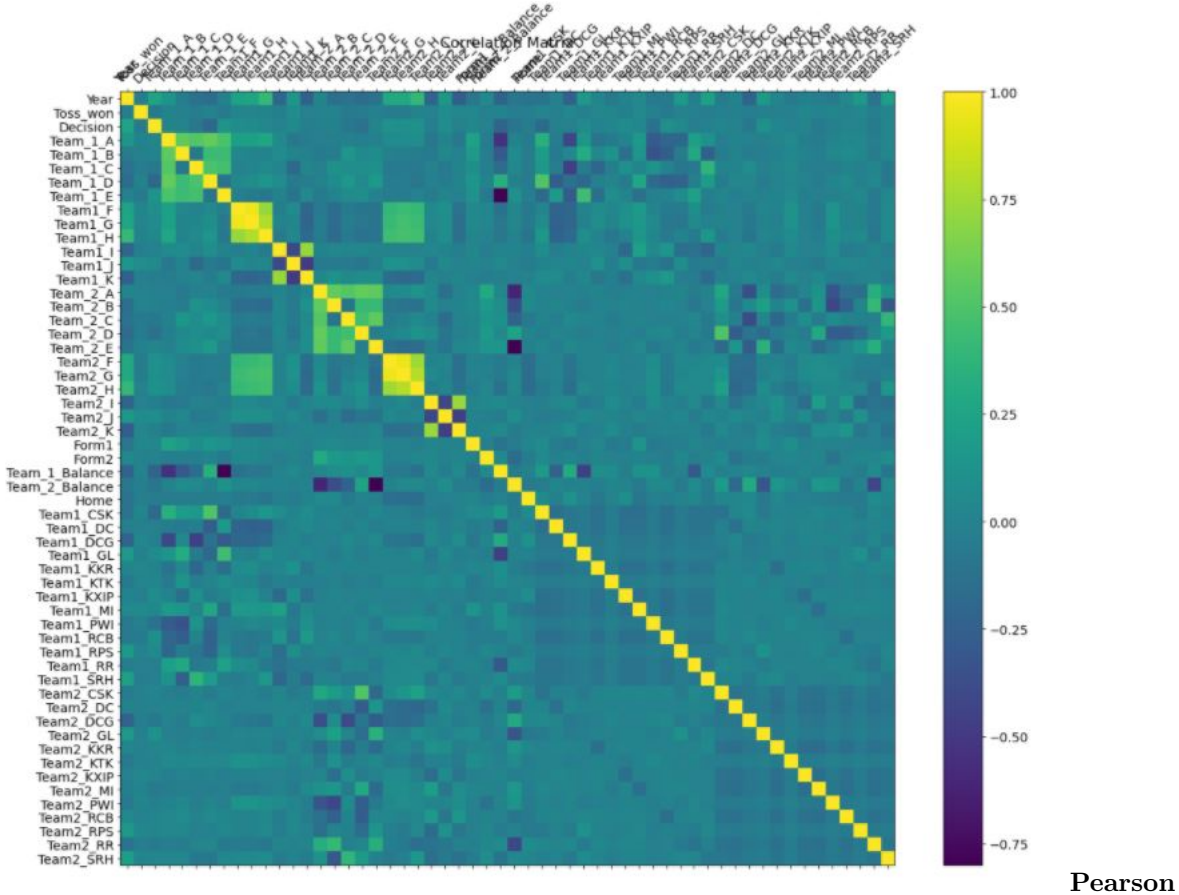
Some Columns in final database and which feature they represent

Year
Team1
Team2
Toss_won
Decision
Venue
Time
Ump1
Ump2
Ump3
Ref
Squad1
Squad2
Team1_A
Team1_B
Team1_C
Team1_D
Team1_E
Team1_F
Team1_G

Team1_H
Team1_I
Team1_J
Team1_K
Team2_A
Team2_B
Team2_C
Team2_D
Team2_E
Team2_F
Team2_G
Team2_H
Team2_I
Team2_J
Team2_K
Form1
Form2
Team_1_Balance
Team_2_Balance
Winner

Table 2
All columns in final database

Correlation Matrix :



Pearson Correlation Matrix

4 Methodology

We tried various machine learning models to predict the outcome of the match. Some of the models used which gave decent accuracy are:

4.1 Gaussian Naive Bayes Classifier

As we did not have much data, we started predicting using classifiers with simple structures. Despite its over-simplified assumptions, it worked quite well for our dataset.

Data was scaled using Standard Scaler. A correlation matrix was printed and showed high correlation between Team1F and Team1G features, and between Team2F and Team2G features respectively, so we dropped the TeamG columns, to satisfy the assumption of Naive Bayes, i.e., the independence of all features. Oversampling was not used. No Grid-Search was used as there was no hyper-parameter to tune. The Gaussian Naive Bayes classifier (from sklearn) gave 62.7% f1-score accuracy-score and 73.1% f1_score.

	precision	recall	f1-score	support
0	0.47	0.33	0.39	21
1	0.68	0.79	0.73	38
accuracy			0.63	59
macro avg	0.57	0.56	0.56	59
weighted avg	0.61	0.63	0.61	59

Fig 2
Classification report for Gaussian Naive bayes model

4.2 Random Forests

Since we didn't have a very large dataset, overfitting was a major concern. Random forests have been known to limit overfitting without substantially increasing the error. The parameter grid we used for cross validation had a large no of trees present([200,500,700]) to increase available options, while also having low values of max_depth(4-8), so that complex models would be penalized. Both these help in reducing overfitting. RF Classifiers can also handle both missing and categorical values. There weren't a lot of missing values in our dataset, but we did have a lot of categorical columns such as Venue,Time and Umpire.

As for the dataset, we manually normalized the new features we added, so that any column without a 0/1 value got reduced to a 0-1 range. Then, we used the dummy_variables() pandas function to convert all categorical columns into a list of one hot encoded columns. This is because Random Forest Classifiers only work with integer values, not with strings. (We also removed the squad lists, since they would add unnecessary columns to the data and all their qualities have been reflected in the team stats already)

Finally, we just used sklearn's cross validation for hyperparameter tuning, to get the optimal parameters for our model.

```
{'criterion': 'entropy',
 'max_depth': 6,
 'max_features': 'log2',
 'n_estimators': 200}
```

Fig 3
Chosen Parameters for Random Forest model

Then we generated predictions on the test(2019) data from our model and checked it's accuracy when compared to the actual data. We didn't use the same feature set as previous models as Random Forest is an ensemble method, and would benefit from having more decision points, giving us better accuracy than with the previous set.

	precision	recall	f1-score	support
0	0.00	0.00	0.00	21
1	0.63	0.95	0.76	38
accuracy			0.61	59
macro avg	0.32	0.47	0.38	59
weighted avg	0.41	0.61	0.49	59

Fig 4
Classification report for Random Forest model

4.3 Support Vector Machines

We explored SVMs as SVM is a relatively less complex model which can classify the data using kernel trick and works well with less data.

First of all, Oversampling using SMOTE of the train dataset was done to improve the ratio of the majority class/minority class to 1. After that, the train and test datasets were normalized using StandardScaler().Following this, we used GridsearchCV with 5 folds with SVM Classifier and rbf kernel. As a result of the Cross Validation, the following best parameters were found :

```
SVC(C=1, break_ties=False, cache_size=200, class_weight=None, coef0=0.0, decision_function_shape='ovr'
degree=3, gamma=0.1, kernel='rbf',max_iter=-1, probability=False, random_state=None, shrinking=True,
tol=0.001, verbose=False)
```

We used f1-score as the result-metric.

	precision	recall	f1-score	support
0	0.00	0.00	0.00	21
1	0.64	1.00	0.78	38
accuracy			0.64	59
macro avg	0.32	0.50	0.39	59
weighted avg	0.41	0.64	0.50	59

Fig 5
Classification report for Support vector machine model

4.4 Neural Network

Neural Networks are generally better and more complex classifiers when it comes to a large dataset where other ML algorithms saturate. We did not have a huge dataset, but since we had a good number of features, we used oversampling using SMOTE to increase the number of datapoints and to improve the ratio of majority to minority class data points. Since the accuracy level of other models were not satisfactory we decided to implement a multi layered perceptron, since there could be two reasons for the low accuracy i.e. either the model is too simple to model that data or the features we used were not enough to capture major aspects of the game. We have discussed the second point further in the future work section.

Since a tensor implementation using Keras or pytorch would've been overkill, we used Multi Layered Perceptron(MLP) classifier from sklearn. The main problem here was to select the optimum structure so that the model does not overfit.First grid search to find optimum Structure or number of layers and get a rough idea of number of units in each layer.

```
params={'alpha': 10.0 **(-np.arange(1, 10)),
        'learning_rate_init':[0.001,0.005,0.008,0.0005],
        'activation':['logistic', 'tanh', 'relu'],
        'random_state':[0,1,2,3,4,5,6,7,8,9],
        'hidden_layer_sizes':[(10,10,5),(10,10,10,5),(10,10,10,10,5),(10,10,10,10,10,5),
                              (50,20,10),(30,30,30),(100,100,100),(50,50,50,50)]}
```

Second grid Search was performed after shorlisting the number of layers to 3 and number of units around 10 for a layer. Through this grid search we would get our final structure from the narrowed down space and meanwhile an idea for the range of other hyperparamteres. The grid for second search was:

```
params2={'learning_rate':['invscaling','adaptive'],
         'alpha': [5,4,3,2,1,0.1,0.001,0.01,0.005,0.0001],
         'learning_rate_init':[0.1,0.05,0.001,0.005,0.008,0.0005,0.01],
         'random_state':[0,1,2,3,4,5,6,7,8,9],
         'hidden_layer_sizes':[(4,4,4),(6,6,6),(10,10,10),(8,8,8),(15,15,15),
                               (10,6,4),(5,4,4),(10,4,6),(12,6,4)],
         'activation':['relu','tanh','logistic']}
```

Having narrowed down the structure to a considerable extent we performed a grid search for other hyper-parameters like learning rate initial value, regularization parameter, momentum(for sgd) etc.
Grid for final search was :

```
params3={'learning_rate':['adaptive'],
         'alpha': [1.2,1,0.5,0.2,0.3,0.1,0.001,0.002,0.01,0.005,0.0001],
         'learning_rate_init':[0.001,0.005,0.008,0.0005,0.01],
         'random_state':[0,13,41,52,89,148,185,210],
         'hidden_layer_sizes':[(10,6,4)],
         'tol':[0.005,0.008,0.003,0.0025,0.001,0.0001,0.0003,0.0004],
         'activation':['relu']}
```

We reached a final set of hyperparameters which was :

```
MLPClassifier(learning_rate_init=0.0015,random_state=185,
              solver='adam',activation='relu',warm_start=True,
             learning_rate='adaptive',max_iter=3000,
              hidden_layer_sizes=(10,6,4),alpha=0.5,tol=0.0025)
```

From results of the previous run we reached at the following hyperparameters The best case was obviously over-fitting due to less data and a complex model for the problem but we saved the grid search results in an excel file and then mapped the test and train accuracy values for all the grid points to a value that is high if the difference in train and test accuracy are low and high if the test accuracy is high. doing that we reached our final set of hyper-parameters. The excel file can is generated by the Neural net code file but we have also shared the generated score file in :'/data/score7.csv'

The final classification report for the test set is :

	precision	recall	f1-score	support
0	0.75	0.29	0.41	21
1	0.71	0.95	0.81	38
accuracy			0.71	59
macro avg	0.73	0.62	0.61	59
weighted avg	0.72	0.71	0.67	59

Fig 6
Classification Report For neural network model

5 Results

We trained on the IPL matches from 2008 to 2018 and predicted the results for the 2019 season (We did not include the 2020 season in our test set because it is being played in UAE and has just ended). As our test data was imbalanced (38 ones and 21 zeroes), we decided to choose F1-score instead of accuracy score as the metric for most models.[9] The performance of our models was as follows :

Model Used	Metric Used	Score
Gaussian Naive Bayes	Accuracy Score f1-Score	62.7% 73.1%
Random Forest	Accuracy f1-score	61.01% 75.79%
SVM	f1-Score	78.3%
Neural Network	Accuracy score F1 score	71% 80%

Table 3
Summary of performance of all the models.

The 2019 season Points Table looked like this as per our predictions.

	Teams	Wins_SVM
0	CSK	8
1	KKR	7
2	MI	9
3	RR	7
4	DC	8
5	RCB	6
6	SRH	7
7	KXIP	7

Fig 7
Predicted Points Table 2019

The actual IPL 2019 Points Table after the league stage [10] was as shown below :

	Team	Pld	Won	Lost	Tied	N/R
Q	MI	14	9	5	0	0
Q	CSK	14	9	5	0	0
Q	DC	14	9	5	0	0
Q	SRH	14	6	8	0	0
5	KKR	14	6	8	0	0
6	KXIP	14	6	8	0	0
7	RR	14	5	8	0	1
8	RCB	14	5	8	0	1

Fig 8

Actual Points Table of IPL 2019 After League Stage

As can be seen, the win predictions vary usually by only 1 match per team, and the difference is at most 2 (for Rajasthan Royals).

6 Conclusions

Our research allowed us to understand the factors which matter when an IPL match is played between 2 teams. It offers a novel way to divide a team into 2 different levels - Team level and Player level. The various player level stats are important deciding factors for the win. Also, the team level stats tell us about the general history of the team. We found out that the experience of a team in a specific situation helps them face a similar situation again.
So, via this work, we are able to quantify different aspects of batting and bowling relevant for predicting the output of the game. Further, this research also provides good insight into dealing with less amounts of data, and tuning the various machine learning models accordingly.

Since our test set was imbalanced, we used f1 score as a metric instead of accuracy. After training all our models, we found Neural Networks to have the highest f1-score of 80% closely followed by SVM with 78.3%. Then, we had Random Forest at 75.79%, and finally Naive Bayes at 73.1%. Neural networks have been known to perform well at learning the weights and biases of the different features. Our research showed that with proper hyper-parameter tuning, neural networks can function well even with lesser amounts of data. At the same time, the good performance of SVM can be explained by its convex loss function which makes it easier to reach the minima in less data.

7 Future Work

In future, we are planning to include the following :

1. Pressure : A pressure attribute which would assign a real value to each category of match based on its importance like playoffs, qualifiers, finals in increasing order. Every team's performance changes under pressure. This parameter is used to model that.

2. Captaincy Index : We all know that a captain has extra responsibilities than other players, like making changes in bowling and batting order. We are planning to include the captaincy index as another attribute. So, extra importance will be given to the captain. We tried implementing it this time only, but were unable to extract the captain from the format of the scraped data.

3. Fielding Score : On the team level, we could not include the fielding attributes because that couldn't be done from the data we scraped, so we are planning to scrape the fielding data of players separately and derive a new fielding index from that for each player.

4. Transfer Learning using T20 data : To overcome the lack of data, we plan to train our model on T20 data first and use those values as the priors for training the model on IPL data. Since the format is similar, the attributes would have similar effect on the match result, and hence these priors will give reasonable and more specific initial values for our parameters than some random value. This will also give us an advantage while using more complex models like Multi Layered Perceptron since having a good starting value would account to some extent for the less data.

5. Performance against a Particular Team : The rivalry aspect of the teams can be captured by this attribute. Certain teams have much better track records against some particular teams. So, win percentage against that team can be used as a feature.

6. Regressor : We had another idea of implementing a Decision tree regressor to estimate a range of predicted score for both the teams for the match and then use the outputs of the classifiers and the proximity of scores to get a final estimate of winning team.

8 References

1. Subramanian Iyer and Ramesh Sharda,"Prediction of athletes performance using neural networks: An application in cricket team selection,"April 2009,Expert Systems with Application 36(3):5510-5522, https://doi.org/10.1016/j.eswa.2008.06.088

2. Jhanwar and Pudi,"Predicting the Outcome of ODI Cricket Matches: A Team Composition Based Approach",August 2016,Conference: Machine Learning and Data Mining for Sports Analytics, ECML-PKDD'16, https://www.researchgate.net/publication/309457872

3. Parker,Burns and Natarajan,"Player valuations in the Indian Premier League",January 2008,DOI: 10.1017/CBO9780511 https://www.researchgate.net/publication/265012640

4. Rabindra Lamsal and Ayesha Chowdhary,"Predicting Outcome of Indian Premier League (IPL) Matches Using Machine Learning", https://arxiv.org/abs/1809.09813

5. Singhvi, Arjun, A. Shenoy, Shruthi Racha and Srinivas Tunuguntla. "Prediction of the outcome of a Twenty-20 Cricket Match." (2015), http://pages.cs.wisc.edu/~shruthir/Documents/MachineLearning_Final_Report.pdf

6. Abhishek Naik, Shivanee Pawar, Minakshee Naik, Sahil Mulani, "Winning Prediction Analysis in One-Day- International (ODI) Cricket Using Machine Learning Techniques", International Journal of Emerging Technology and Computer Science, Volume: 3 Issue: 2 (April2018), https://1library.net/document/yrw769vz

7. Geddam Jaishankar Harshit, Rajkumar S, "A Review Paper on Cricket Predictions Using Various Machine Learning Algorithms and Comparisons among Them", International Journal for Research in Applied Science Engineering Technology (IJRASET), IJRASET17099 (April 2018), https://www.ijraset.com/fileserve.php?FID=17099

8. Decision making with the analytic hierarchy process http://www.rafikulislam.com/uploads/resourses/197245512559a37aadea6d.pdf

9. Accuracy vs f1 score : Medium Article by Purva Huilgol https://medium.com/analytics-vidhya/accuracy-vs-f1-score-6258237beca2#

10. IPL 2019 Points Table : https://www.iplt20.com/points-table/2019