IPL Match Prediction

Haramrit Singh Khurana (G01229319) , Neel Narvankar (G01210145)

December 9, 2020

Abstract

Cricket is one of the famous outdoor sports. The T20 format is most watched and loved by the people, where no one can guess who will win the match until the last ball of the last over. In India, The Indian Premier League (IPL) started in 2008 and now it is the most popular T20 league in the world, Since its introduction in 2003, T20 has become the star attraction within cricket for its high voltage action and exciting phases of play, As a result of its increasing demand, teams are spending money towards a variety of avenues to gain a distinct competitive edge. As part of its popularity in this report, we try to predict the outcome of the IPL matches. IPL has a lot of statistical data, there is data about batting records, bowling records, individual player records, the scorecard of different matches played, etc. This data can be put to proper use to predict the results of games. In this project, we aim to answer one primary question, what is the best classification algorithm to predict the results of an IPL match. The data mining algorithms that are used are K-Nearest Neighbor, SVM, Naive Bayes, Decision Tree, Logistic Regression, Random Forest, and Neural Networks. This report also discusses some of the important features that were selected and also the ones which were derived from the domain knowledge, an example of such features derived are average runs and average wickets of each team. Given the 60 matches played overall in one particular season, our project was able to predict 61% of those matches using the Sequential Model (neural network).

1. Introduction

Cricket is a sport played in many countries all around the world. It is a fairly straightforward game played between two teams both consisting of 11 players. The result can either be a win, a loss, or a tie. In addition to these natural factors, there can be unnatural elements such as match-fixing affecting the outcome of the game as well. Considering all these unpredictable scenarios of this unpredictable game, there is a huge interest among the spectators to do some prediction either at the start of the game or during the game. Many spectators also play betting games to win money. So, keeping in mind all these popular and financial aspects of the game, this report aims at studying the problem of predicting the game results before the game has started based on the statistics and data available from the data set. The study uses the Indian Premier League data set of all 13 seasons played till now i.e. from 2008 to 2020. The prediction can be done taking into consideration a player's performance as well as based on the team performance. However, keeping these factors aside, it is still possible to make a prediction based on past team and player statistics. The report discusses a methodology that we followed for the game result prediction which consists of first the attribute selection algorithms which trim down the list of attributes to only important ones namely toss_decision, avgTeam1, avgTeam2, avgWckTeam1, avgWckTeam2, team1_toss_win, team1_bat and then the data mining algorithms which can be applied to those attributes. The game prediction problem that we are studying does not take into consideration the player's performance but it does take into consideration the team's past performance at a high-level extent along with the other factors like toss winner, toss decision, average runs scored in the previous season, etc. The attribute selection techniques consist of the **backward selection** method. The data mining algorithms used are K-Nearest Neighbor, SVM, Naive Bayes, Decision Tree, Logistic Regression, Random Forest, and Neural Networks. This project makes use of Python for implementing the model since it allows us to use inbuilt modules like ***NumPy*** (for mathematical operations on data), ***pandas*** (for data manipulation and transformation) and ***sklearn*** (for allowing direct use of different data mining algorithms to train, test and validate our model). It also allows us to use Keras which is a very sophisticated module for designing and implementing neural networks.

2. Problem Statement

IPL involves a lot of money from sponsors and stakeholders and it is the biggest league of the most popular sport in India. Our main motive with this project is to find the best subset of features and the best classification algorithm which helps us predict the outcome of a match between two teams as accurately as possible. Such analysis can also help in strategic planning for teams to understand their rivals better, and to work on their own weaknesses to improve their chances of victory.

2.1 Notations

IPL - Indian Premier League, CSK-Chennai Super Kings, SVM- Support Vector Machine

3. Literature Review

[1] ANALYZING IPL MATCH RESULTS USING DATA MINING ALGORITHMS

The paper speaks about how data mining tools predict the future trends thus giving an opportunity to predict the outcome of an IPL match. The proposed system in this paper focuses on analyzing the IPL match results by applying classification algorithms. The dataset used in this paper includes parameters such as which **two teams are playing the match, winner of the toss, the number of runs the team won by and the number of wickets the team won by** and using these attributes the outcome 'winner' is predicted. Undersampling of the dataset is done to handle the imbalanced dataset the majority and minority class had a difference of about 150 samples , The algorithms used in this paper include Decision tree, Naive Bayes and Linear Regression. The paper also provides a comparison of the accuracies before and after sampling of Data.

[2] Prediction of Indian Premier League-IPL 2020 using Data Mining Algorithms

The paper speaks about how cricket has a lot of statistical data and how this data can be used to **predict the first inning score** of a cricket match **on the basis of current run-rate which can be calculated as the amount of runs scored per the number of overs bowled**. The paper includes factors like **number of wickets fallen, venue of the match, toss** and predicts the score in each of the innings and finally the winner of the match using the Random Forest algorithm. The analysis in this paper is done using Rapid miner, an information software system platform developed by the corporate of an equivalent name that gives an integrated setting for information preparation, machine learning, deep learning, text mining and predictive mining.Analysis are also done using **Decision Tree, Naive Bayes, SVM and Random Forest**. In this paper Prediction of IPL2020 are done **on the basis of survey**. The survey was done using google forms and the data set formed by this **includes about 107 rows and 12 attributes**.

[3] DATA MINING ON CRICKET DATA SET FOR PREDICTING THE RESULTS

The report is about analysis done on IPL dataset using Weka software,the dataset used has **matches from 2008 to 2016** the dataset used in this analysis **had 21 attributes some of which include Match_number, First Inning score, Overs_Played_In_Second_Innings, Second_Innings_Run_Rate** etc.The methodology followed by the report is as follows; Data generation, Data Cleaning, Attribute Selection, Data mining, analysis of results **The Attribute selection was done using Wrapper method and Ranker method**, The wrapper method uses a subset evaluator algorithm like CFSSubsetEval and WrapperSubsetEval in combination with a searching technique such as Best First search or Greedy search to evaluate the subset of attributes.The prediction part is done using the data mining algorithms Decision Tree (J48), Random Forest, Naïve Bayes and K-Nearest Neighbor.

[4] Analysis on Attributes Deciding Cricket Winning

The paper is about **how various attributes can affect the outcome of a game**, various factors like home crowd advantage,performances in the past, experience in the match, performance at the specific venue, performance against the specific team and the current form of the team and the player. The paper takes into consideration that the result of **a cricket match depends on more in-game and pregame attributes**. Pre-game attributes **like Pitch, Team Strength, Weather, Venue** etc. and **in-game attributes like run rate, total run, strike rate, wickets in hand** etc. The paper also explains the importance of attributes such as Home ground advantage, Weather, Pitch, etc. The paper addressed the factors or attributes that Cricket match depends on.

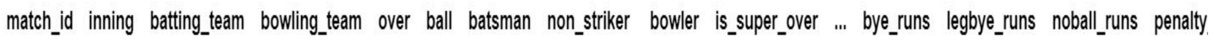
4. Methods and Techniques

a. Feature Selection and Pre-processing

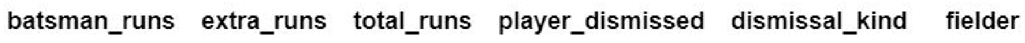
The dataset obtained from Kaggle has 4 files of interest:

1. Matches IPL 2020.csv
2. Deliveries IPL 2020.csv
3. Deliveries IPL 2008-2019.csv
4. Matches IPL 2008-2019.csv

Feature selection needs to be done on each of these files carefully so as to obtain the best results in predicting the outcome of a given match. In this section we shall discuss which features were selected and why were they selected, also the preprocessing done on the data. The later sections will describe processing and transforming the data based on the features that were selected and the methods used for prediction. Let us begin with the Deliveries IPL 2008-2019.csv file, the features/attributes of this file are as shown in the image below.

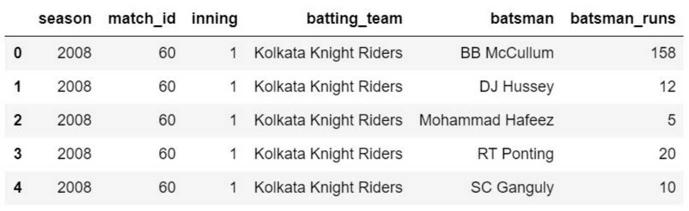


match_id inning batting_team bowling_team over ball batsman non_striker bowler is_super_over ... bye_runs legbye_runs noball_runs penalty



batsman_runs extra_runs total_runs player_dismissed dismissal_kind fielder

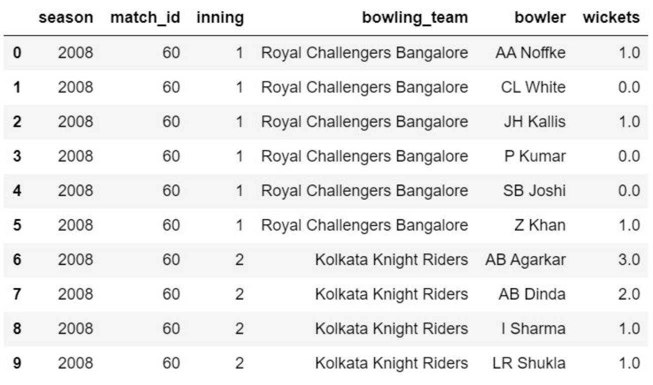
From the attributes shown above, the first step was to obtain the number of runs scored by each batsman in each match in that particular season that helped in adding a derived attribute of the average number of runs scored by each team which helped in predicting the results more accurately. This was achieved by doing a sequence of **group_by and merge** operations. The first step in doing this was grouping together the data on match_id, inning, batting_team, and batsman, then taking the sum of this new groupby data on batsman_runs. While doing this, we must keep in mind the wide runs that are added to the total which must not be counted towards the batsman's total, the next step was handling these wide deliveries. The final data frame after doing this looks something like the data frame below:



	season	match_id	inning	batting_team	batsman	batsman_runs
0	2008	60	1	Kolkata Knight Riders	BB McCullum	158
1	2008	60	1	Kolkata Knight Riders	DJ Hussey	12
2	2008	60	1	Kolkata Knight Riders	Mohammad Hafeez	5
3	2008	60	1	Kolkata Knight Riders	RT Ponting	20
4	2008	60	1	Kolkata Knight Riders	SC Ganguly	10

Fig:Shows the data frame of number of runs scored by a batsman in each match

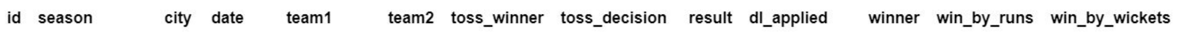
We do a similar preprocessing for the bowlers and find out the number of wickets taken by each bowler in each match, this helped in deriving our second derived attribute the average number of wickets taken by each team. The preprocessing on the data frame for the number of wickets taken by bowlers in each match is shown in the image below:



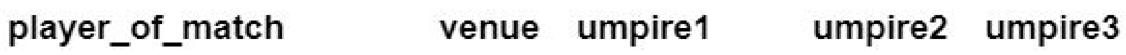
	season	match_id	inning	bowling_team	bowler	wickets
0	2008	60	1	Royal Challengers Bangalore	AA Noffke	1.0
1	2008	60	1	Royal Challengers Bangalore	CL White	0.0
2	2008	60	1	Royal Challengers Bangalore	JH Kallis	1.0
3	2008	60	1	Royal Challengers Bangalore	P Kumar	0.0
4	2008	60	1	Royal Challengers Bangalore	SB Joshi	0.0
5	2008	60	1	Royal Challengers Bangalore	Z Khan	1.0
6	2008	60	2	Kolkata Knight Riders	AB Agarkar	3.0
7	2008	60	2	Kolkata Knight Riders	AB Dinda	2.0
8	2008	60	2	Kolkata Knight Riders	I Sharma	1.0
9	2008	60	2	Kolkata Knight Riders	LR Shukla	1.0

Fig:Shows the data frame of number of wickets taken by a bowler in each match

This was the preprocessing that was done on Deliveries IPL 2008-2019.csv, now we will look at pre-processing and attribute selection that was done based on our knowledge of the domain on Matches IPL 2020.csv. The attributes that are in this file are as follows:



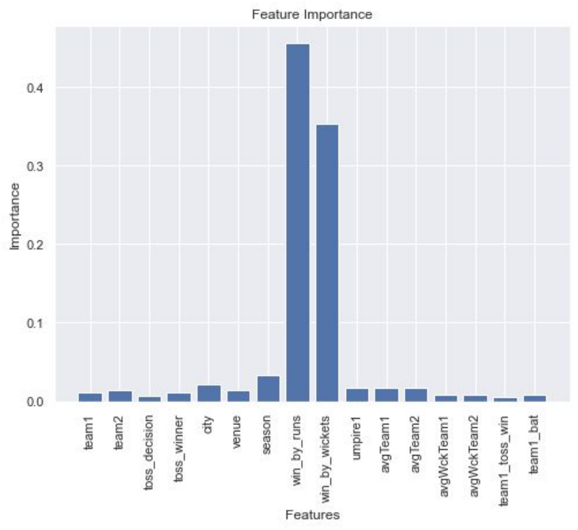
id season city date team1 team2 toss_winner toss_decision result dl_applied winner win_by_runs win_by_wickets



player_of_match venue umpire1 umpire2 umpire3

This was the set of features we had after pruning all the unnecessary attributes:

['team1', 'team2', 'toss_decision', 'toss_winner', 'city', 'venue', 'season', 'win_by_runs', 'win_by_wickets', 'umpire1', 'avgTeam1', 'avgTeam2', 'avgWckTeam1', 'avgWckTeam2', 'team1_toss_win', 'team1_bat']



This is the feature importance graph plotted for all these features. It is clear that **win_by_runs** and **win_by_wickets** are the 2 most important features which almost single handedly influence the outcome of a match. However, these 2 features cannot be considered as this data can only be fetched after the match has ended. Hence they cannot be used to "predict" the outcome of a match, and thus need to be discarded from the feature set. From the domain knowledge of the game and by experimentation, we also had to eliminate city, venue, season, and umpire1 since they have little to no effect on the game as such. toss_winner and toss_decision were also converted to binary attributes team1_toss_win (indicating whether team 1 won the toss) and team1_bat(indicating whether team1 batted first). We also decided to introduce two new attributes which are the average runs scored and the average wickets taken by each team. This was done by using the two data frames which we obtained before, we then sum the data frame on the batsman's runs and then group by the season and the team and mean this value to get the average number of runs scored by each team in that particular season, we then store each of these averages in the dictionary, so the dictionary now contains the average runs scored by each team in that particular season. We then append these values to the Matches IPL 2020 so that each team's batting average is now added to the data frame, similarly, we do this to the average number of wickets taken by each team in the IPL, the average runs, and wickets columns This led us to have the following final set of features:

toss_decision avgTeam1 avgTeam2 avgWckTeam1 avgWckTeam2 team1_toss_win team1_bat

There was a minor imbalance between the number of training data points in which team1 won, and in which team2 won. This imbalance was fixed by <u>upsampling the minority</u> class to match the number of majority class data points.The next step was to <u>encode</u> the names of each team with a numerical value. The next step was converting <u>categorical to numerical data</u> which was done with the help of 'cat.codes()' which replaces all the similar categories with the same numerical value, the final process was <u>data normalization</u> which was done with the help of 'MinMaxScaler()' from 'sklearn'.

b. <u>Data Cleaning:</u>

Since the data contains a lot of teams and a lot of seasons there is a naming error done in some of the seasons, wherein the names of some teams are written in abbreviated form or in some cases a letter at the end of the team is missed for e.g, we needed to replace 'Rising Pune Supergiant' with 'Rising Pune Supergiants' because there was an 's' missing at the end. Some of the seasons had "CSK" instead of "Chennai Super Kings"; this was the same for almost all the teams.

c. <u>Methods used for prediction</u>

Once we were ready with the most important features that could influence the outcome of a cricket match, we moved on to train different machine learning models on this dataset, and test and find out the one that yields the highest accuracy. We tested the following models for our project:

1. K-Nearest Neighbor
2. SVM
3. Naive Bayes
4. Decision Tree
5. Logistic Regression
6. Random Forest
7. Neural Networks

We used the sklearn module to implement and test most models. Our general approach was to use GridSearchCV to find out the most optimal hyperparameters for training each model and then test each model on the testing dataset for the year 2020. GridSearchCV tries every possible combination of the model hyperparameters to find the most optimal set of hyperparameters, and then applies cross validation on the trained model thereafter to find the mean accuracy obtained by using those parameters. Following is a brief overview of how some models work and why they were used. Experimental results of these models and the optimal hyperparameters are explained in the later section

K-Nearest Neighbor (k-NN):
In k-NN binary classification, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class that is most common among its k nearest neighbors where k is a positive integer (fairly small for practical purposes). We chose to implement this model due to the simplicity of the algorithm. Parameters used: n_neighbors, weights, distance

SVM:
In SVM binary classifiers, the primary task is to find the optimal hyperplane (or the decision boundary) which separates the data points of the two classes as accurately as possible. We chose SVM because of its ability to classify linearly on a non-linear feature space. Parameters used: C, kernel

Naive Bayes Theorem
It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors.We choose Naive Bayes classifier because of its ability to assume that the presence of a particular feature in a class is unrelated to the presence of any other feature. Parameters used: var_smoothing.

Decision Tree
Decision Tree is a type of algorithm that includes conditional 'control' statements to classify data, decision trees are used for forecasting future outcomes and assigning probabilities to those outcomes which is useful in the case of predicting the outcome of a match. We use decision trees because they break down complex data into more manageable parts. Parameters used: criterion, max_depth, min_sample_leaves.

Logistic Regression
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 Logistic Regression is used in making estimates, and forecasts, hence it proves to be very useful in case of making predictions of a match. Parameters: Penalty and c.

5. Discussion and Results

5.1 Datasets

The dataset we're working on is collected from Kaggle. This dataset contains ball by ball details of the domestic tournament held in India i.e. the Indian Premier League played from 2008 to 2019. It is a 20-20 format of the tournament. It means that each team bats or bowls for a maximum of 20 overs each and the result of the game is decided at the end of the total 40 overs played. The dataset we used is an imbalanced dataset. The data set includes a good spread of categorical and numerical data.

This dataset contains two CSV files:
1. Deliveries IPL 2008-2019.csv which contains about 21 features of ball by ball data of the game and about 180,000 data entries.
2. Matches IPL 2008-2019.csv which contains about 18 features ranging from the teams between the matches are played, the city where the match is played, and the outcome of the match.

5.2 Evaluation Metrics

We have evaluated our models based on the accuracy with which they predict the outcome. So, the evaluation metric chosen was the train and test accuracy to evaluate how good a model is. We also decided to test the F1 score for each model.

5.3 Experimental Results

As shown in **Fig1**, For k-nearest neighbors, we manually ran the algorithm to find the zero in on the best range of k values to help us reduce overhead during GridSearch.

Fig2 shows a quick analysis of the teams who won the game given they won the toss, since we decided to keep the feature 'team1toss_win', we thought it would be helpful to visualize how many teams won the game given they have won the toss.

Fig3 & Fig4 shows the model loss and the model accuracy respectively for each epoch from 0 to 50.

We used GridSearch along with k-fold cross validation to find the most optimal hyperparameters for our models. The following estimators turned out to give the best results:

1. **K-nearest neighbours:**
```
KNeighborsClassifier(metric='euclidean', n_neighbors=10, weights='distance')
```

2. **SVM:**
```
SVC(C=10, gamma='auto', kernel='linear')
```

3. **Naive Bayes classifier:**
```
GaussianNB()
```

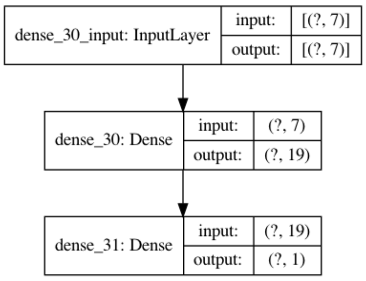
4. **Decision Tree classifier:**
```
DecisionTreeClassifier(max_depth=23, min_samples_leaf=2)
```

5. **Logistic Regression:**
```
LogisticRegression(C=100, penalty='l2')
```

6. **Random Forest Classifier:**
```
RandomForestClassifier(max_depth=80, max_features=2, min_samples_leaf=3,
min_samples_split=10)
```

7. Neural Network (built with Keras):

As for the neural network with 2 hidden layers, we performed systematic experimentation to arrive at a fairly optimal model giving us decent results. The model had an input layer followed by two hidden layers where the first layer had an activation function of 'relu' and the second had the 'sigmoid' activation for binary classification. Loss function used to train the neural network was binary cross-entropy loss and the validation split in the train-test data was 90-10. The figure below shows this;



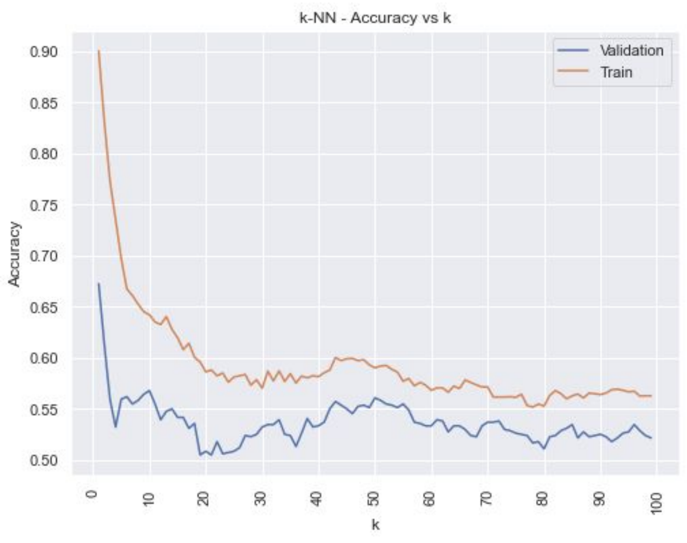


Fig1: k-NN - accuracy vs k

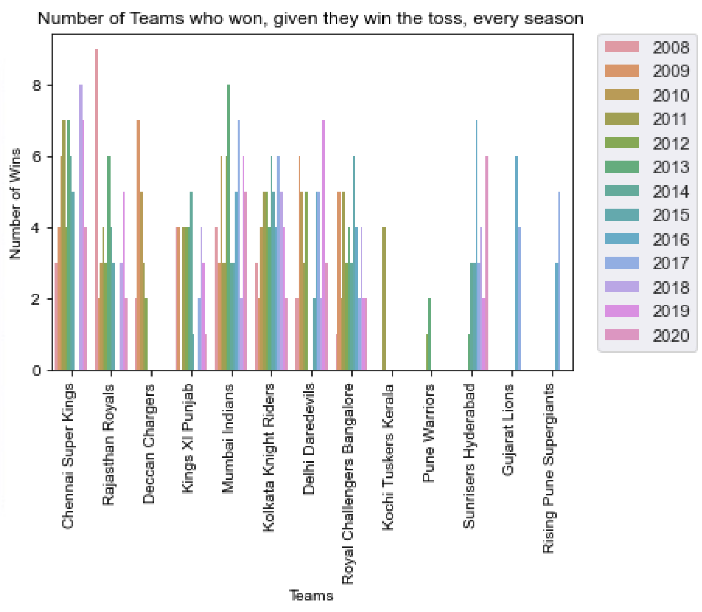
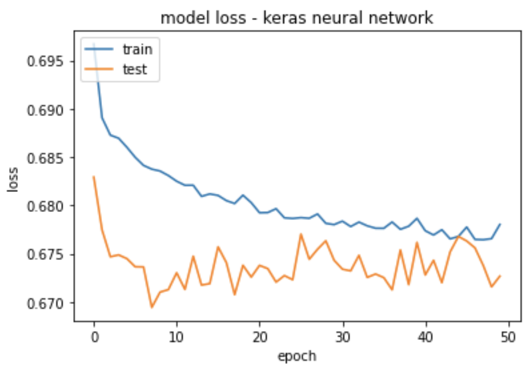


Fig2: winning probability based on toss win



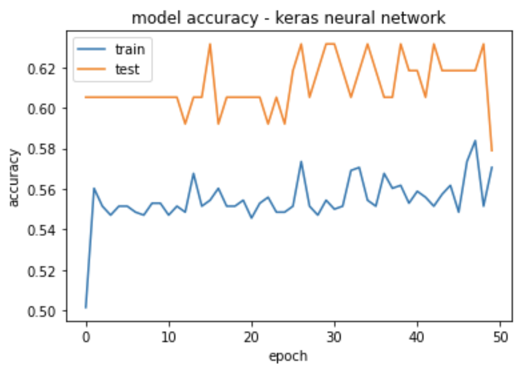


Fig3: neural network - model loss

Fig4: neural network - model accuracy

All the graphs have been explained in the section above

Results:

Name of Model	Train accuracy	Test accuracy	F1 score
KNN	57.41	53.33	68.28
SVM	55.69	51.67	62.95
Naive Bayes	51.72	46.67	64.15
Decision Tree	56.21	55	63.61
Logistic Regression	56.61	53.33	56.5
Random Forest	53.7	45	63.7
Neural Network	55.47	61.1	64.46

Future Work:

- Whether a team is playing home or away is also an important factor when it comes to predicting the outcome of a match. However, we could not choose to include this feature since we were planning to predict matches for the 2020 season and the matches took place in Dubai i.e. outside India, due to COVID-19. So no team had any home advantage as such, and thus although we could have chosen to include the home city as a feature for training, it would have just given us misleading and inaccurate results. This feature can be used in the future when things come back to normal and matches are played in India where these teams can have an added advantage of playing on their home ground.
- One more thing which can be included in future is to predict the outcome of the matches depending on the performance of the players on individual level like, the total number of runs scored by the player in the previous season or the total number of wickets taken by a bowler in the previous season, etc.

Conclusion:

This project was made to predict the outcome of an IPL match by analyzing the matches played between the seasons 2008 and 2019.The models were evaluated on the basis of accuracy. The prediction was done with various models namely, K-Nearest Neighbor, SVM, Naive Bayes, Decision Tree, Logistic Regression, Random Forest, and Neural Networks; out of which the highest accuracy of 61.1% was obtained using Neural Networks. Predicting the outcome of a match depends on a lot of factors, this project takes into consideration the best attributes that can help in predicting the outcome.

References:

[1] ANALYZING IPL MATCH RESULTS USING DATA MINING ALGORITHMS, Shimona.S, Nivetha.S, Yuvarani. P, International Journal of Scientific & Engineering Research Volume 9, Issue 3, March-2018

[2] Prediction of Indian Premier League-IPL 2020 using Data Mining Algorithms, Sachi Priyanka, International Journal for Research in Applied Science & Engineering Technology (IJRASET)

[3] Data Mining on Cricket Data Set for Predicting the Results, Sushant Murdeshwar, Project Report, Rochester Institute of Technology, New York

[4] Analysis on Attributes Deciding Cricket Winning Swetha ,Saravanan.KN, International Research Journal of Engineering and Technology (IRJET)

[5] https://www.kaggle.com/aritrachakraborti/ipl-2020-ball-by-ball-data

[6] https://scikit-learn.org/0.21/documentation.html

[7] https://keras.io/api/