DATA MINING AND DATA WAREHOUSING

IPL Match Winner Prediction Analysis

Team Number — 3

Sanket Kar	200911264
Ravi Sharma	200911268

Abstract

Forecast of result of an interaction between teams utilizing Machine Learning calculations is a vital perspective in cricket. Records of the past performances of players and other related information are frequently examined to make models that predicts the triumphant group. This model is made utilizing the Machine Learning calculations like Logistic Regression, Decision Tree and Random Forest and their outcomes are much of the time looked at upheld the assessment measures as exactness, accuracy, review. For expectation we've utilized some of the highlights like Team1, Team2, match winner, toss winner, toss decision, whether result was normal or not, DL applied or not, win by runs, win by wickets for forecast with higher exactness. For this interaction we took dataset from Kaggle.com, imported essential python libraries then we envisioned information. We prepared ML model and anticipated results. For expectation we concentrated on different grouping calculations to get best exactness.

Introduction

Cricket is quite possibly of the most seen sport in whole world. Numerous regular variables influencing the cricket match, colossal media scope, and a gigantic wagering fan market have serious areas of strength for given to show and prepare the sport from different viewpoints. Albeit the complicated and various guidelines decide the game, the power of individual and their exhibitions on a specific day, and numerous different boundaries assume a significant part in impacting the best consequence of a match. Due to the rapid advancement of technology, the sizeable market for wagering, and the popularity of cricket, everyday people are now using machine learning predictions to guess the outcomes of cricket matches. Information science and machine learning simplify life in every manner. With the aid of machine learning and game outcome predictions made before the game, the players and trainers will be able to identify their weak areas. To predict who will win the cricket tournament, we make use of a few machine learning concepts and calculations. Additionally, computational experiences are closely related to (and frequently overlap with) machine learning, which is expanding and focuses on forecasting using invention.

Predicting a match's outcomes has become increasingly important with the advancement of both growth and sports. One of the most well-known and popular team sports in the world is cricket. We are using ML concepts like controlled learning to predict the match leaders to predict the outcomes of a T20 (twenty) cricket match. We prepare the models using profession measurements as well as group competitions like batting and bowling demonstrations. Nonetheless, the capricious standards overseeing the game, the limit of players and various boundaries have a vital impact in affecting the consequences of the match. Subsequently we are utilizing managed learning calculations to foresee the final product of the game and it will help the mentors of the group to comprehend and investigate how really the group is veering off-track.

Literature survey

[1] prepared the selected highlights on machine learning models after analysing a variety of factors to predict the winner even before the match began. To do this, they

prepared datasets of different sizes using a variety of Machine Learning calculations, including Arbitrary Timberland, SVM, Guileless Bayes, Strategic Relapse, and Choice Tree. This structure will help the cricket team's captain preferentially evaluate the team's unity and cricket analysis.

By using a directed learning method, [2] have created a model for predicting the outcome of a One Day International (ODI) game. They involved both the late execution of each player and the dynamic player's observations. They have therefore decided that KNN is their preferred formula for arrangement.

[3] This research uses live match anticipation, where match prediction is done while the game is still in progress. In this way, highlights—such as the wickets lost, the match scene, the positioning of the group, the pitch report, home field advantage, etc.—rather than prematch factors like past player success, past group insight, etc., are taken into account. To predict the outcomes of cricket matches, this paper analyses various elements of cricket that have been gathered from various examination papers.

[4] Used machine learning concepts like regulated learning to predict the outcome of One Day (ODI) cricket contests. In order to increase the model's accuracy, this will also highlight the model's critics and areas for improvement. Therefore, in these papers, various machine learning computations were used, and a correlation was created to identify the best-prepared model with the most accurate calculations. We had taken into account the prior coordinate measurements as well as group plausibility in terms of execution, such as batting, bowling, and other activities.

A prediction model [5] with a variety of components is essential because the model's accuracy will increase with the quality of the highlights. Significant funding has been provided, making it essential to build a more accurate model. In order to create a grouping model that can assist various owners and selectors by providing the best precise model, predefined techniques had to be broken down and made. There are various highlights that are still to come that will be considered as giving the forecast model more accuracy, such as historical data from real events, player information, and resistance data.

[6]The participants' behaviour in terms of their strengths and shortcomings is used to forecast the result of the match. Mentors, group skippers, selectors, and leaders will benefit from this by taking into account the different factual proportions of the participants. To forecast matches against specific competitors, they have used guided ML techniques like SVM with various components like straight, nonlinear, and RBF. Kimplies has been used to identify the top three to four players who most closely resemble a specific person. All players in the top request, the middle request, and the lesser request have taken the final test.

[7] Developed the expectation structure for a T20 encounter in a specific IPL match.

The first inning's score can currently be predicted using the current run rate, which can be assessed using the basic recipe; though, it excludes a number of factors like the home field, setting, throw, and other factors that are excessively significant during forecast because they can alter the result of the game. Additionally, there is no anticipation model to forecast how the game will turn out in the second inning.

To create the ampleness of the model, [8] created an assumption model with a variety of characteristics. The model uses two methodologies. The first technique predicts the first inning's result by taking into account the runs per over, the number of wickets lost, the batting order, and the settings. The subsequent process anticipates the outcome of the game in the forecoming innings while taking into account vague characteristics from the prior method. Innocent Bayes Classifier is used in the second inning and Direct Relapse Classifier in the first. The 50 over game was divided into 5 overs for each of the two methods, and in this way, every game played between 2002 and 2014 was documented.

The goal of this evaluation, according to [9], was to grasp how routine cricket matches are. The English Twenty Over Region Cricket Cup served as the ultimate challenge. The primary characteristics close-by planned features provided knowledge about 500 groups and their players. The model was developed with two cycles in mind, such as first and primarily considering group highlights alone and then later considering both group and player highlights. The functionality was tested during seasons 2009-2014 after being set up using prior season data for similar situations. The ideal model had the ability to directly predict the outcome of a coordinate combined with the concept of confounding different levelled highlights.

Indian ODI cricket coordinates are the focus of [10] study, which also makes use of Market Container tools and ascribes for extracting various affiliation rules. To put the ideas into practice, they concentrated on various elements such as throw result, groups, home benefit, and so on. They measured assistance and certainty measures to foresee instances which would lead to group losing using ten years' worth of historical data from matches.

[11] This algorithm is used to predict how a match with important information will turn out. Different elements were included in the process of extracting the highlights, but forecasting incorporated the most crucial elements. They also developed a player positioning system based on presentation metrics and a group structure for openings that characterises the most crucial areas contributing to game dominance. They grouped all of the players who were consistent with their exhibition using K-means, and they used KNN (K-closest neighbour) to find players who were consistent with a specific player. Using straight, polynomial, and RBF, an SVM model was created. (Outspread Premise capability). They therefore preferred SVM with RBF bit for anticipation.

[12] It takes into account the various elements that affect how an Indian Chief Association contest turns out. The visiting team, the home team, the throw victor, the throw selection, the arena, and consequently the weight of the individual teams, are the seven factors that primarily affect how an IPL encounter turns out. With a precision

of 72.66% and an F1 score of 0.72, the planning machine learning model for match result of a bartering-based 2020 configuration chief association is therefore extremely acceptable at this point.

[13] They used all of the 2019 season's matches to evaluate their model and 692 matches from a variety of seasons (2008–2019). In ongoing, we can commence at the top of the first inning. Despite the fact that our review was limited to IPLT 20 matches, the same technique could be used to predict outcomes in test cricket and ODI matches, among other types of cricket matches.

The authors put forth the fluffy bunching theory in [14]. The IPL batting measurements results were divided into various groups, and the information mining strategy of bunching produced expert and compelling exact results. The use of MATLAB allowed us to complete this task. The Indian Head Association's batting measurements, which contain fluffy information, are arranged into appropriate groups using the grouping concept.

A review [15] is for the viability of using data from Twitter to gauge the outcomes of the interaction. It is suggested that work be done to determine whether machine learning techniques are effective when used to interpret data collected from virtual entertainment companies and other real-world events. Backing Vector Machine, Credulous Bayes Classifier, and the Straight Relapse are the techniques used in their research. The SVM approach stands up well.

[16] asserts that management must always select the greatest employees to guarantee the best outcomes. The study provides the best approach for selecting the finest team utilising data mining techniques as opposed to the time-consuming conventional method. The best team must be selected when the time and date for a particular tournament are determined, as doing so increases the possibility that the team will win.

[17] proposes the notion of developing a prediction system that uses historical data to determine whether upcoming matches will be won or lost. They will employ Nearest Neighboring, Linear Regression, and Clustering techniques to demonstrate how effectively the algorithm mathematically anticipated the model's outcomes.

[18] model that forecasts the outcome of each ball in a match. The probability is determined using the par score concept developed by Duckworth & Lewis, and it makes it clear who will win the match.

[19] categorised aspects that affect match outcomes, such as home field advantage, the impact of day and night on the toss, and the significance of batting first, using artificial intelligence methods, particularly bayes classifiers in machine learning. CricAI is the name of the computer programme that is the result of this effort. The programme calculates the likelihood of winning based on input variables, such as having the home

field advantage at the start of the game. The CricA1 is useful when cricket teams are competing. In order to increase the likelihood of success in the real world, it is utilised to alter a few parameters.

[20] The models use data analytics methods that are connected to machine learning. The outcome of a game that has been impacted by rain may be difficult to predict.It's crucial to pay attention to your batting, bowling, fielding, squad selection, result forecast, and target revision in a rain-affected game. The mathematical model developed based on knowledge of the outcomes of previous matches can be used to solve the match prediction problem. SVM is employed to build the Predictor models. This work is being proposed using Deep Mayo Predictor.

[21] Together with parameter-based filtering and the Bayesian Prediction Technique, statistical analysis and data mining make a strong team. The database contains information about the players in the current IPL. With the aid of this work, the necessary data can be mined for the prediction algorithm to use in order to produce a statistical analysis of each player.

An algorithm is developed to predict cricket scores and player performance using the ODI dataset [22]. used supervised methods like SVM and Nave Bayes. For exact categorization, KNN and MLP clustering algorithms are employed.

[23] assembled this article using information from the Indian Premier League's official website. They reviewed the data and selected a few of the most crucial features because it contained numerous features. They performed certain model applications and data preprocessing using the Scikit-learn machine learning programme. Recursive, univariate, and low variance traits were disregarded. These feature selection techniques allowed them to isolate 5 out of 15 features as being of utmost significance. Aspects include the home team, visiting team, location, toss winner, toss decision, and winner. Data from the IPL's first ten seasons, followed by data from the eleventh, were used to train the Random Forests and Multiple Linear Regression models. Their model properly classified 41 out of 60 matches. As a result, they are just 68.33 percent accurate, which isn't very good. This model has limitations because it only uses two machine learning models and five features.

[24] used machine learning techniques to predict each player's talents based on the field, pitch type, rival team, and many other variables. Using the Random Forest Algorithm, the model provided accuracy for batters, bowlers, and all-rounders of 76%, 67%, and 96%, respectively. They were able to forecast game outcomes and choose the top players thanks to this approach.

Based on the 2008–2019 datasets, [25] employed data mining techniques to predict player appraisal in the IPL. Data mining methods are used to analyse a player's performance and determine his base price using player statistics. Using these assumptions, they predicted how players would be chosen for the IPL.

Methodology

Data Pre-processing:

Preparation of raw data to be used for a Data mining algorithm is the system of data medication. It's the most important part of creating a Data mining literacy model.

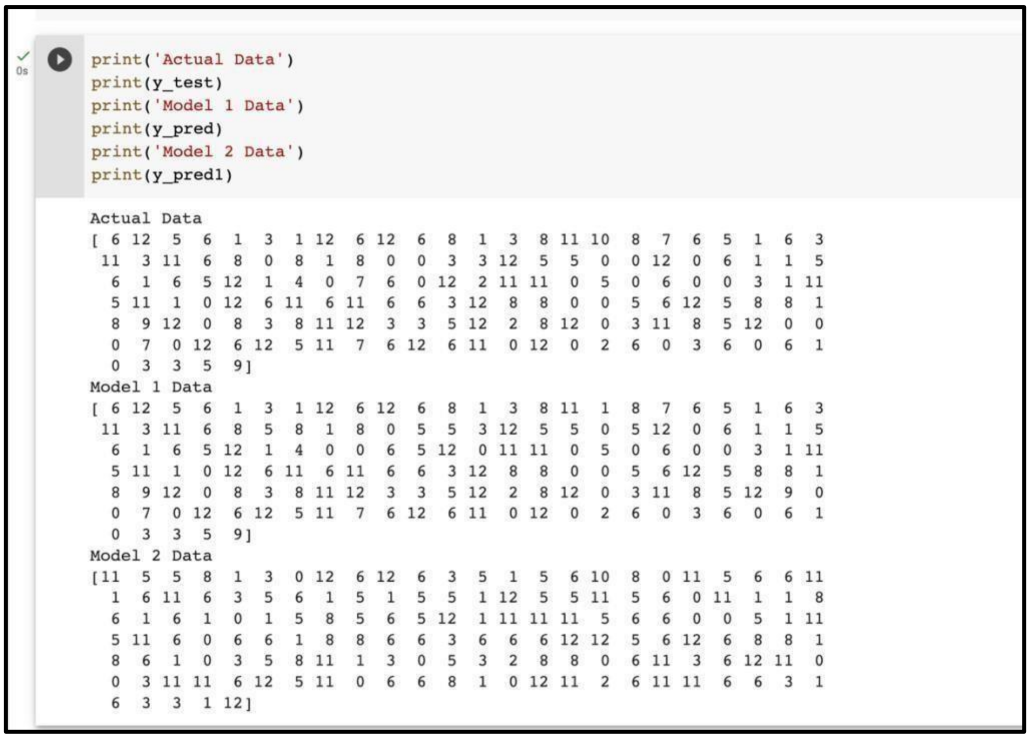
When embarking on a Data mining design, we do not always have access to acceptable, well- prepared data. It’s also needed to clean and prepare the data before working in any data affiliated exertion. That’s why, we employee a pre-processing stage fashion.

In our case, the data we acquired had ample amounts of attributes describing data on every match that had occurred in the Indian Premier League between 2008 and 2019. However, it was crucial to pick only those fields with sufficient correlation to aid our predictive objective.

We began by finding null values presence in the dataset. We found most ‘umpire 3’ values to be null, and accordingly decided to drop the column due to it’s relevance with the predictive analysis. Over the years, names of IPL teams have also altered and this was not reflected over the dataset. This alteration had also been accommodated to not lose vital data correspondence between older and newer data across the timeline.

Some of the categorical data that has effect in the result is converted into numerical form using panda library function, pd.get_dummies(). It performs one-hot encoding. In one-hot encoding, each unique value in a categorical variable is transformed into a new categorical variable, and a binary indicator variable is created for each unique value. This process is also known as creating dummy variables. This way the number of columns increases to 43 via this process.

Following this, the dataset is Label encoded and therefore every team is mapped to a particular number.
Post encoding, the data appears as seen in *Fig0.1*.



```
print('Actual Data')
print(y_test)
print('Model 1 Data')
print(y_pred)
print('Model 2 Data')
print(y_pred1)
```

```
Actual Data
[ 6 12  5  6  1  3  1 12  6 12  6  8  1  3  8 11 10  8  7  6  5  1  6  3
 11  3 11  6  8  0  8  1  8  0  0  3  3 12  5  5  0  0 12  0  6  1  1  5
  6  1  6  5 12  1  4  0  7  6  0 12  2 11 11  0  5  0  6  0  0  3  1 11
  5 11  1  0 12  6 11  6 11  6  6  3 12  8  8  0  0  5  6 12  5  8  8  1
  8  9 12  0  8  3  8 11 12  3  3  5 12  2  8 12  0  3 11  8  5 12  0  0
  0  7  0 12  6 12  5 11  7  6 12  6 11  0 12  0  2  6  0  3  6  0  6  1
  0  3  3  5  9]
Model 1 Data
[ 6 12  5  6  1  3  1 12  6 12  6  8  1  3  8 11  1  8  7  6  5  1  6  3
 11  3 11  6  8  5  8  1  8  0  5  5  3 12  5  5  0  5 12  0  6  1  1  5
  6  1  6  5 12  1  4  0  0  6  5 12  0 11 11  0  5  0  6  0  0  3  1 11
  5 11  1  0 12  6 11  6 11  6  6  3 12  8  8  0  0  5  6 12  5  8  8  1
  8  9 12  0  8  3  8 11 12  3  3  5 12  2  8 12  0  3 11  8  5 12  9  0
  0  7  0 12  6 12  5 11  7  6 12  6 11  0 12  0  2  6  0  3  6  0  6  1
  0  3  3  5  9]
Model 2 Data
[11  5  5  8  1  3  0 12  6 12  6  3  5  1  5  6 10  8  0 11  5  6  6 11
  1  6 11  6  3  5  6  1  5  1  5  5  1 12  5  5 11  5  6  0 11  1  1  8
  6  1  6  1  0  1  5  8  5  6  5 12  1 11 11 11  5  6  6  0  0  5  1 11
  5 11  6  0  6  6  1  8  8  6  6  3  6  6  6 12 12  5  6 12  6  8  8  1
  8  6  1  0  3  5  8 11  1  3  0  5  3  2  8  8  0  6 11  3  6 12 11  0
  0  3 11 11  6 12  5 11  0  6  6  8  1  0 12 11  2  6 11 11  6  6  3  1
  6  3  3  1 12]
```

Fig 0.1: Demonstrating Team Label Encoder

On finishing pre-processing, the data was analysable and some crucial inferences could be made. For example:

1) Analysis on matches won by all IPL teams.

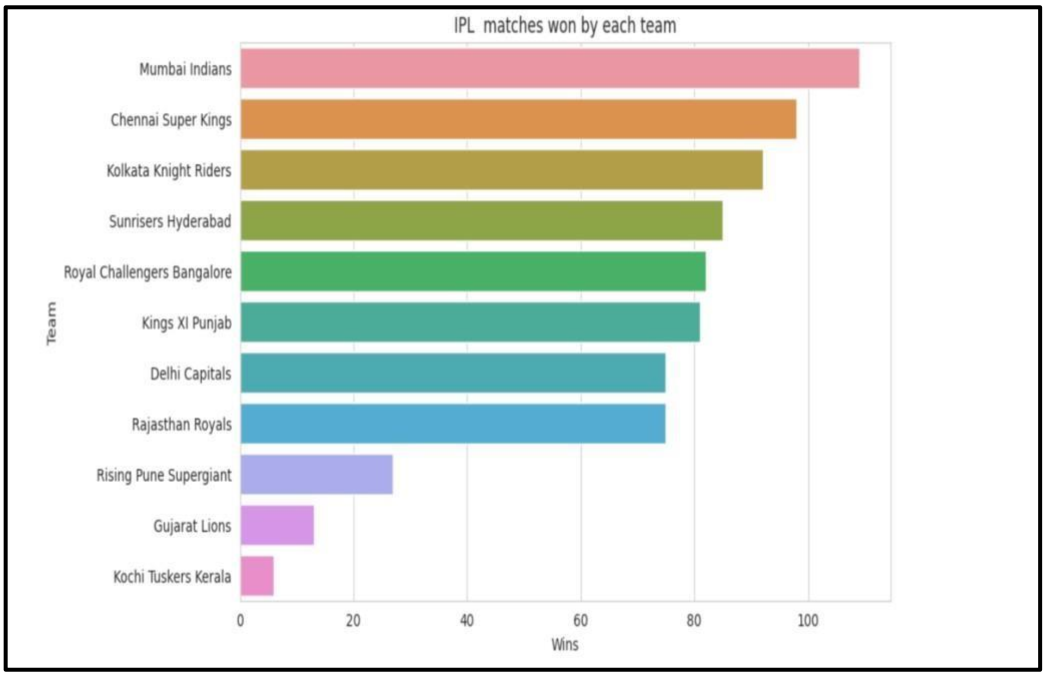
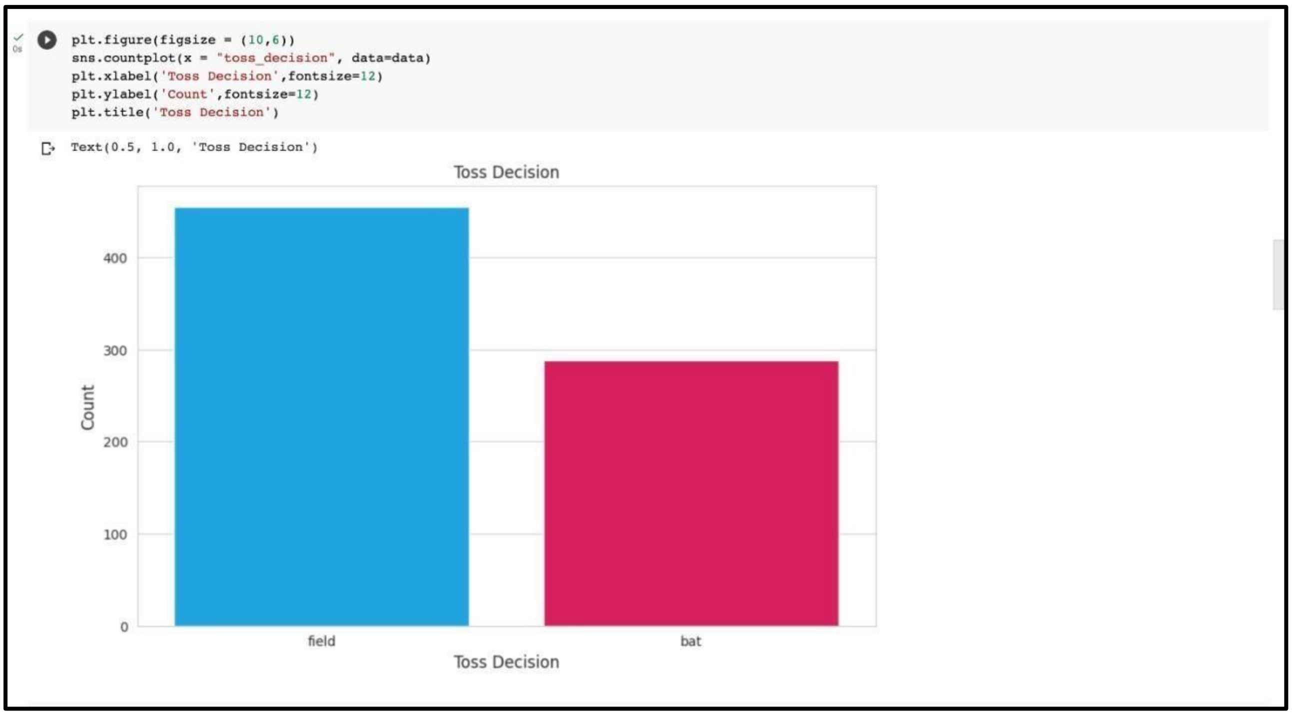


Fig 0.2: Number of matches won by each Team

2)Toss decisions made by teams



```
plt.figure(figsize = (10,6))
sns.countplot(x = "toss_decision", data=data)
plt.xlabel('Toss Decision',fontsize=12)
plt.ylabel('Count',fontsize=12)
plt.title('Toss Decision')
```

```
Text(0.5, 1.0, 'Toss Decision')
```

Fig 0.3: Toss decision made by Teams

Keeping all features or fields to create the model would result in reduced accuracy and some features were dropped that didn’t seem to affect the result. Some of these features were Season, city, player of match, etc.

Now the dataset is prepared for model creation and evaluation.

Model:

The objective of the prediction was to find the winning probability between two teams. The output is of the type win or lose, true or false, or simply a binary classification. Hence, the models chosen were most appropriate for binary classification.

The training-test data chosen split is an 80-20 split.

Logistic Regression:

A statistical technique called logistic regression is used to forecast binary outcomes, such as yes/no or 1/0, using one or more input variables, also referred to as independent or predictor variables. For each input data point, the technique generates a probability value between 0 and 1 by fitting a logistic function to the data. Any input value is translated by the logistic function, an S-shaped curve, into a chance value between 0 and 1. The logistic regression model employs this curve's parameters, which are estimated from the data, to forecast the likelihood of a binary result for a new input data point. In machine learning, logistic regression is frequently utilized, especially in fields like fraud detection, customer churn prediction, and medical diagnosis. It is a popular algorithm because of how easy it is to understand and use, as well as how well it can manage nonlinear relationships between the input and output variables.

Decision tree :

Different methods are used by decision trees to decide whether to split a junction into several subjunctions. With each subsequent generation of sub-nodes, the unity of the created sub-nodes improves. In other words, when the goal variable changes, the knot becomes more austere. The decision tree splits the networks into sub-nodes based on all significant traits and chooses the split that results in the most atypical sub-nodes. Regression algorithms are used when the outcome of the data is continuous as opposed to regression trees, which are used when the outcome of the data is discrete or categorical, such as when determining the relative importance of students in the classroom, a person's demise or tone preservation, a loan's blessing, whether a person has cancer or not, etc.

Random Forest(Ensemble Model):

Random Forest is a member of the family of ensemble learning techniques, which combine various models to increase prediction accuracy generally. A final prediction is made using the output of each decision tree in a Random Forest model, which uses numerous decision trees that have been trained on various subsets of the input data. The danger of overfitting the model to the training data is decreased because each decision tree in the model is trained on a random subset of the input variables. In many machine learning applications, including image identification, text classification, and financial forecasting, Random Forests are frequently used. They are renowned for their high precision, scalability, and capacity for handling complicated and cluttered data. One of the key benefits of Random Forest is that it offers a variable importance metric that can be used to pinpoint the most crucial input variables for result prediction. For feature selection and data exploration reasons, this can be helpful.

K-Nearest Neighbors (KNN):

K-Nearest Neighbors (KNN) is a supervised machine learning technique for classification and regression analysis. In KNN classification, the algorithm finds the K closest data points (neighbors) of the query point and places it in the category with the highest degree of similarity among its K neighbours. In KNN classification, K is a crucial hyperparameter, and the user can select its value. A low K value means that the classification choice was made using a sparse set of nearby data points, which could result in overfitting. A high value of K indicates that underfitting may happen if the categorization choice is based on a large number of nearby data points. Text classification, recommendation systems, and image classification are just a few of the problems that KNN classification can be used for. Despite being a straightforward and easy to understand approach, it could be computationally expensive due to large datasets.

XGBoost:

In machine learning and data science, Extreme Gradient Boosting, often known as XGBoost, is a popular and effective open-source gradient boosting framework. It aims to provide efficient and accurate predictions for tasks like classification, regression, and ranking. XGBoost builds a collection of decision trees in a sequential fashion. It starts with a simple decision tree and gradually adds more trees, each one aiming to correct the shortcomings of the one before it. During training, XGBoost uses a gradient descent optimisation technique to determine the best weights for each decision tree and reduce the loss function. XGBoost has a variety of advantages over existing boosting algorithms, including speed, scalability, the ability to tolerate missing values, and the ability to handle different data formats. It also includes a number of regularisation techniques, including early halting and L1 and L2 regularization, to prevent overfitting. For machine learning contests, XGBoost has emerged as one of the most well-liked algorithms. It is also frequently used in business for a number of purposes, including fraud detection, credit risk assessment, and recommendation systems. For machine learning contests, XGBoost has emerged as one of the most well-liked algorithms. It is also frequently used in business for a number of purposes, including fraud detection, credit risk assessment, and recommendation systems.

BLOCK DIAGRAM

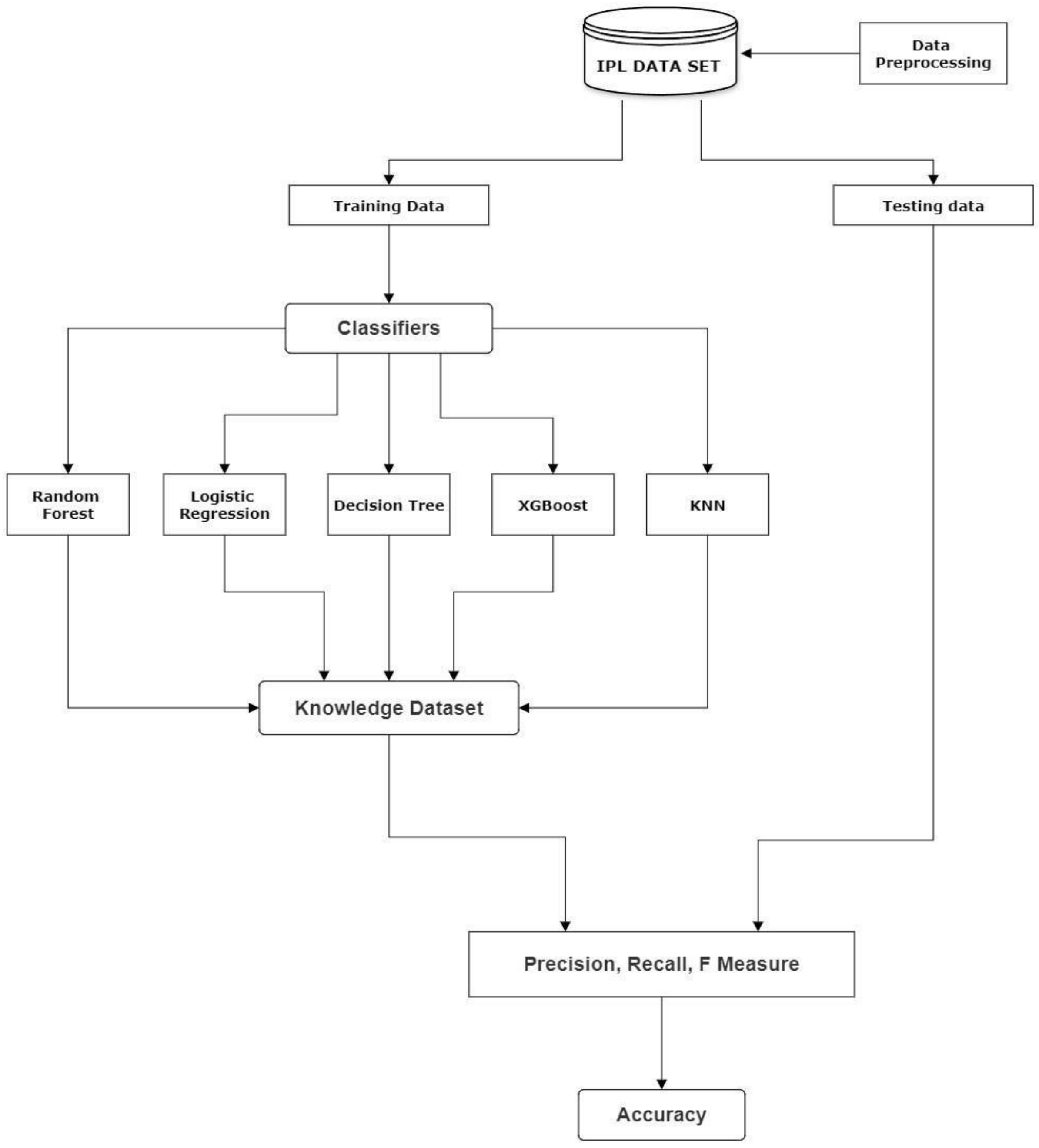


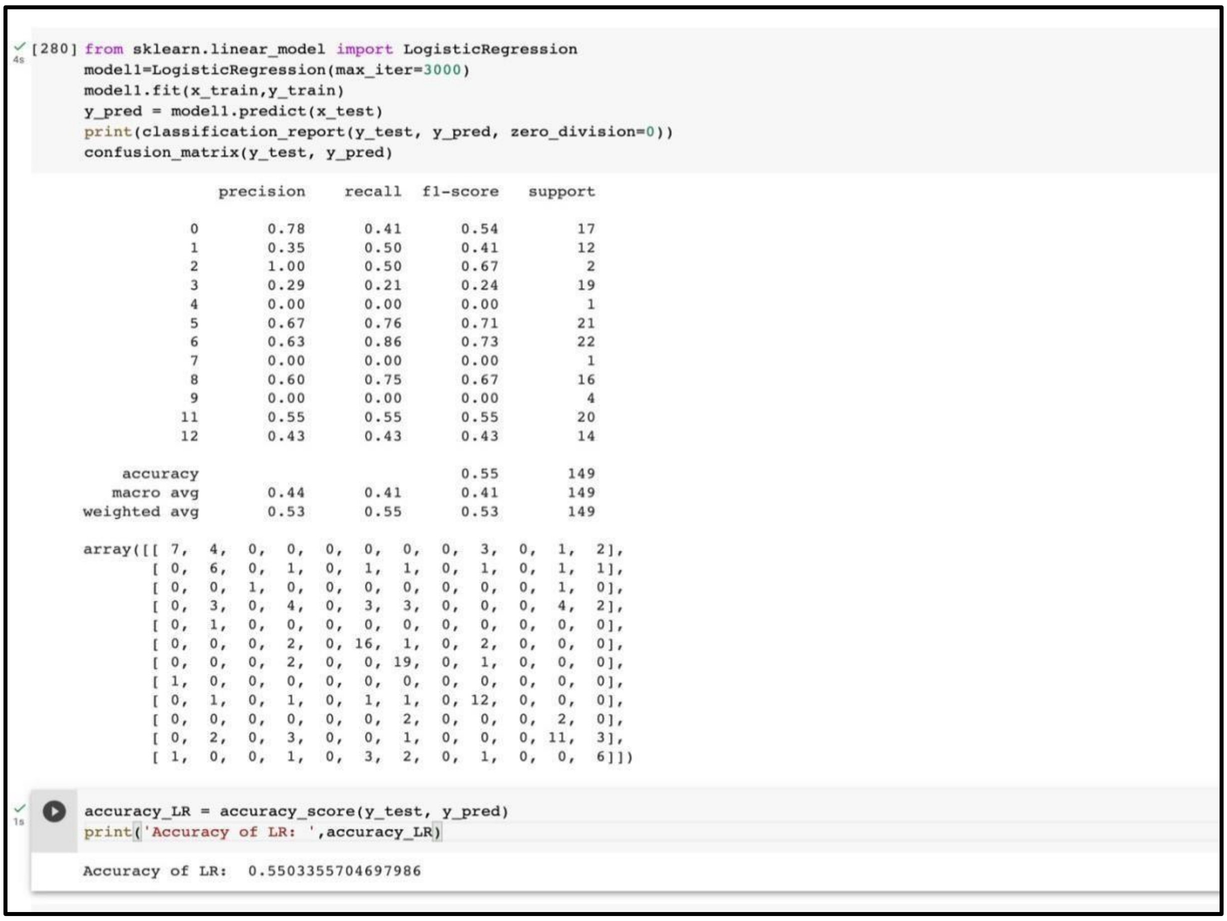
Fig 0.4: Block Diagram to show flow of model processing

Discussion and Results

Logistic Regression:

We used a logistic regression algorithm from Scikit-learn. As the hyperparameter to our algorithm, we set the max-iteration to 3000.

Maximum iterations before the optimisation algorithm must end, even if the convergence criterion has not yet been satisfied, are defined by the max_iterations parameter.



```
[280] from sklearn.linear_model import LogisticRegression
      model1=LogisticRegression(max_iter=3000)
      model1.fit(x_train,y_train)
      y_pred = model1.predict(x_test)
      print(classification_report(y_test, y_pred, zero_division=0))
      confusion_matrix(y_test, y_pred)
```

```
              precision    recall  f1-score   support

           0       0.78      0.41      0.54        17
           1       0.35      0.50      0.41        12
           2       1.00      0.50      0.67         2
           3       0.29      0.21      0.24        19
           4       0.00      0.00      0.00         1
           5       0.67      0.76      0.71        21
           6       0.63      0.86      0.73        22
           7       0.00      0.00      0.00         1
           8       0.60      0.75      0.67        16
           9       0.00      0.00      0.00         4
          11       0.55      0.55      0.55        20
          12       0.43      0.43      0.43        14

    accuracy                           0.55       149
   macro avg       0.44      0.41      0.41       149
weighted avg       0.53      0.55      0.53       149

array([[ 7,  4,  0,  0,  0,  0,  0,  0,  3,  0,  1,  2],
       [ 0,  6,  0,  1,  0,  1,  1,  0,  1,  0,  1,  1],
       [ 0,  0,  1,  0,  0,  0,  0,  0,  0,  0,  1,  0],
       [ 0,  3,  0,  4,  0,  3,  3,  0,  0,  0,  4,  2],
       [ 0,  1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  2,  0, 16,  1,  0,  2,  0,  0,  0],
       [ 0,  0,  0,  2,  0,  0, 19,  0,  1,  0,  0,  0],
       [ 1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  1,  0,  1,  0,  1,  1,  0, 12,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  2,  0,  0,  0,  2,  0],
       [ 0,  2,  0,  3,  0,  0,  1,  0,  0,  0, 11,  3],
       [ 1,  0,  0,  1,  0,  3,  2,  0,  1,  0,  0,  6]])
```

```
accuracy_LR = accuracy_score(y_test, y_pred)
print('Accuracy of LR: ',accuracy_LR)
```

```
Accuracy of LR:  0.5503355704697986
```

Fig 0.5: Logistic Regression report

On running the code, we discovered low levels of average-weighted accuracy and hence, this model is unsuitable due to the large deviation from true results.

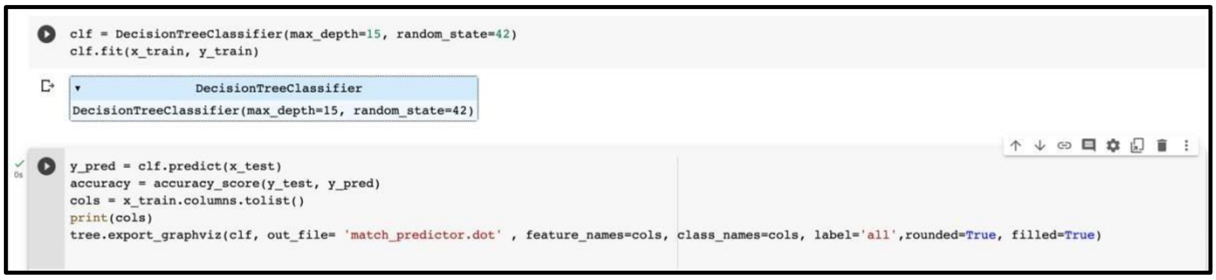
Decision Tree:

Decision Tree algorithm was picked up from the Scikit-learn library.

The decision tree was turned to have a max-height of 15 with a random state float of 42.

The highest number of decision nodes in the tree is constrained by the parameter max height, also referred to as max depth. This is done to limit the model's intricacy and avoid overfitting.

Random is a parameter used in some decision tree algorithms, such as random forest, to introduce randomness into the model. A random subset of features is taken into consideration for the split at each decision node in a random forest, where numerous decision trees are trained on various subsets of the data. This enhances the model's general functionality by lowering the correlation between the trees.

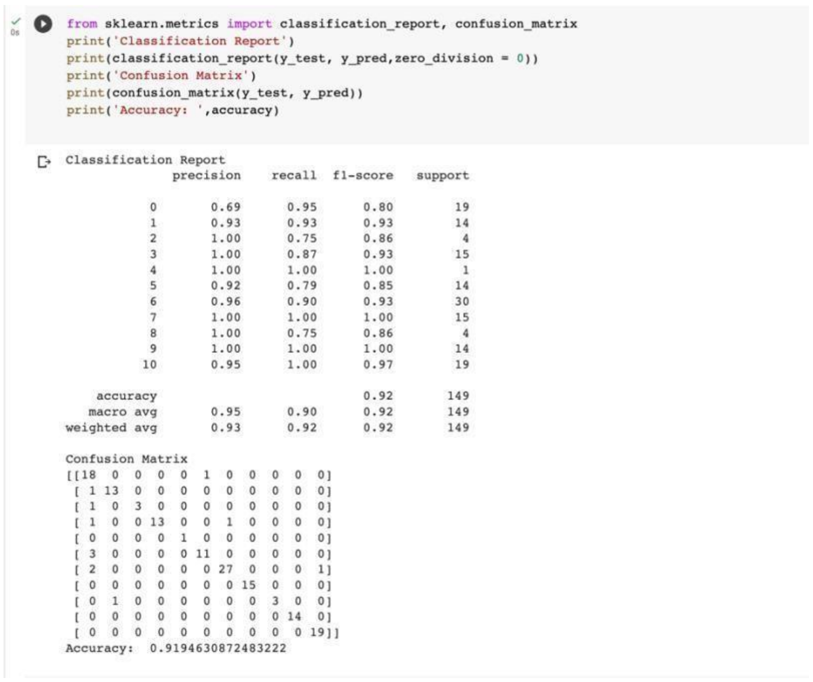


```
clf = DecisionTreeClassifier(max_depth=15, random_state=42)
clf.fit(x_train, y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier(max_depth=15, random_state=42)
```

```
y_pred = clf.predict(x_test)
accuracy = accuracy_score(y_test, y_pred)
cols = x_train.columns.tolist()
print(cols)
tree.export_graphviz(clf, out_file= 'match_predictor.dot' , feature_names=cols, class_names=cols, label='all',rounded=True, filled=True)
```

Fig 0.6: Decision Tree algorithm applied



```
from sklearn.metrics import classification_report, confusion_matrix
print('Classification Report')
print(classification_report(y_test, y_pred,zero_division = 0))
print('Confusion Matrix')
print(confusion_matrix(y_test, y_pred))
print('Accuracy: ',accuracy)
```

```
Classification Report
              precision    recall  f1-score   support

           0       0.69      0.95      0.80        19
           1       0.93      0.93      0.93        14
           2       1.00      0.75      0.86         4
           3       1.00      0.87      0.93        15
           4       1.00      1.00      1.00         1
           5       0.92      0.79      0.85        14
           6       0.96      0.90      0.93        30
           7       1.00      1.00      1.00        15
           8       1.00      0.75      0.86         4
           9       1.00      1.00      1.00        14
          10       0.95      1.00      0.97        19

    accuracy                           0.92       149
   macro avg       0.95      0.90      0.92       149
weighted avg       0.93      0.92      0.92       149

Confusion Matrix
[[18  0  0  0  0  1  0  0  0  0  0]
 [ 1 13  0  0  0  0  0  0  0  0  0]
 [ 1  0  3  0  0  0  0  0  0  0  0]
 [ 1  0  0 13  0  0  1  0  0  0  0]
 [ 0  0  0  0  1  0  0  0  0  0  0]
 [ 3  0  0  0  0 11  0  0  0  0  0]
 [ 2  0  0  0  0  0 27  0  0  0  1]
 [ 0  0  0  0  0  0  0 15  0  0  0]
 [ 0  1  0  0  0  0  0  0  3  0  0]
 [ 0  0  0  0  0  0  0  0  0 14  0]
 [ 0  0  0  0  0  0  0  0  0  0 19]]
Accuracy:  0.9194630872483222
```

Fig 0.7: Decision Tree Classification report

Decision Tree is able to deliver an average of 89.93% of accuracy which is a significant upgrade over Logistic Regression. The decision tree is a multilevel complex tree generated in order to make predictive decisions using this tree. This is the snippet of a small part of the tree made, however it is exported as a dot graph (file) and is included in the code for preview.

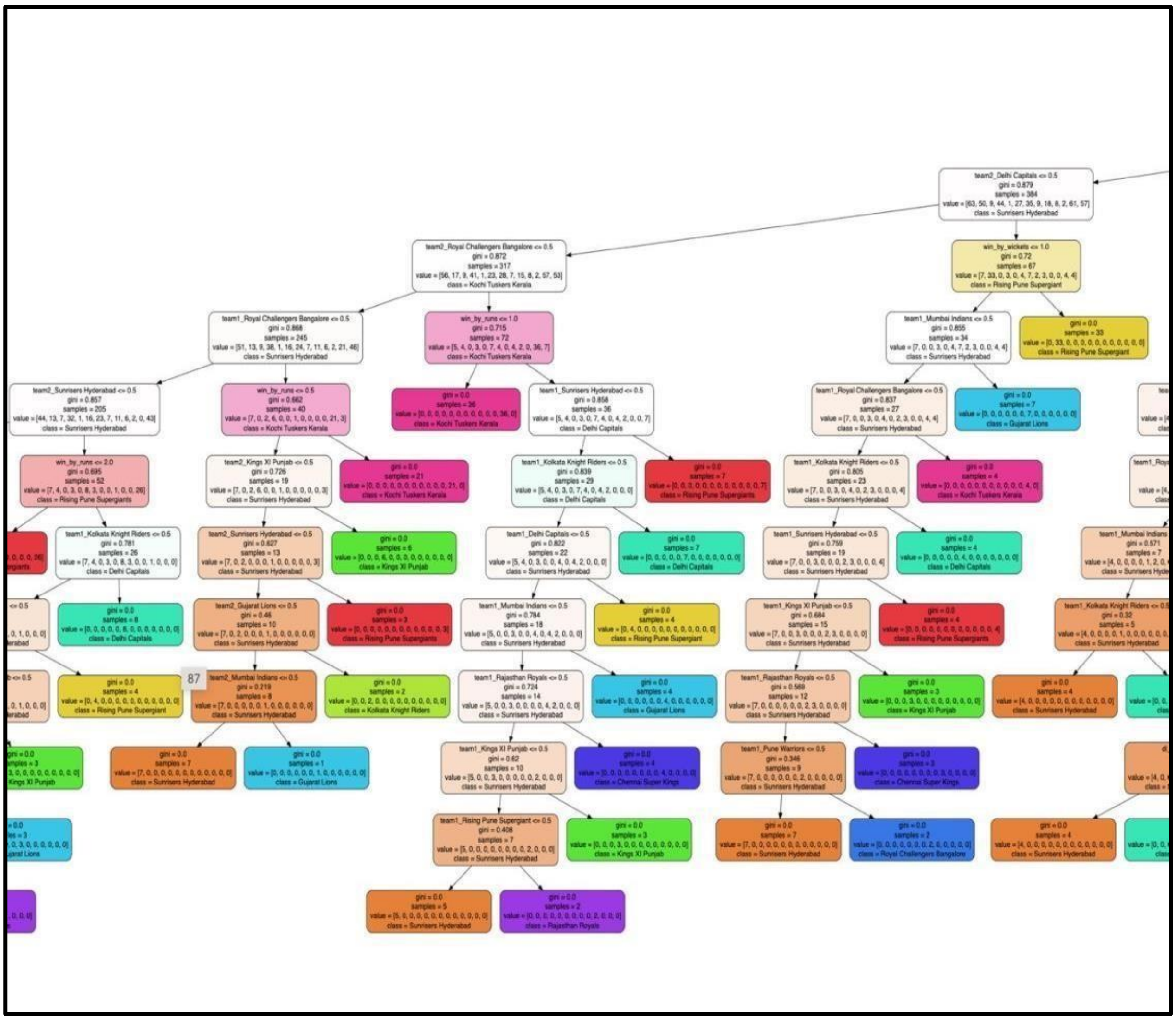


Fig 0.8: Decision Tree visualization snippet

Random Forest(Ensemble):

Random forest is a highly scalable algorithm that can handle large datasets with high dimensionality. It is also computationally efficient and can be trained in parallel, making it suitable for big data applications like the large amount of dataset being handled in our case.

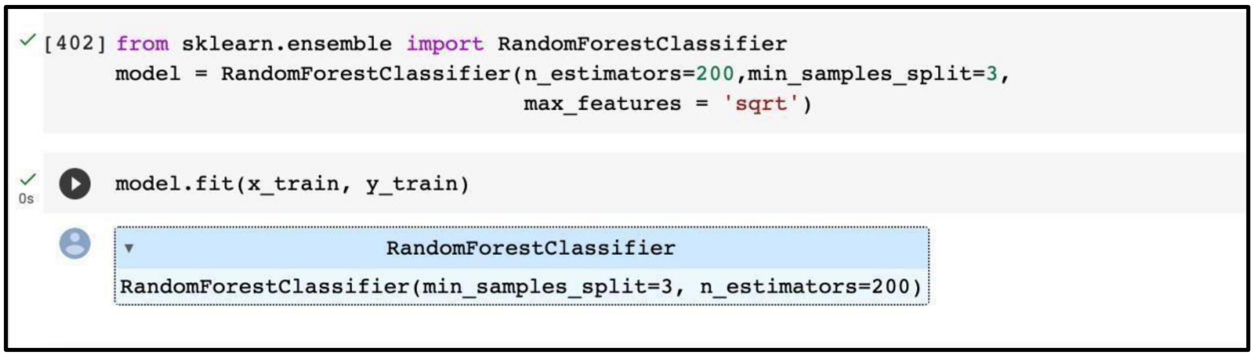
The hyperparameter were set as follows:

N_estimators: 200,

Min_sample_split=3,

Max_features = 'sqrt'

- n_estimators: The amount of decision trees that will be constructed in the random forest is specified by this parameter.
- min_samples_split: This parameter defines the bare minimum number of samples necessary to split a decision tree internal node. It aids in regulating the depth of the trunk and avoiding overfitting.
- max_features: The highest amount of features to take into account when determining the best split at each node is specified by this parameter.

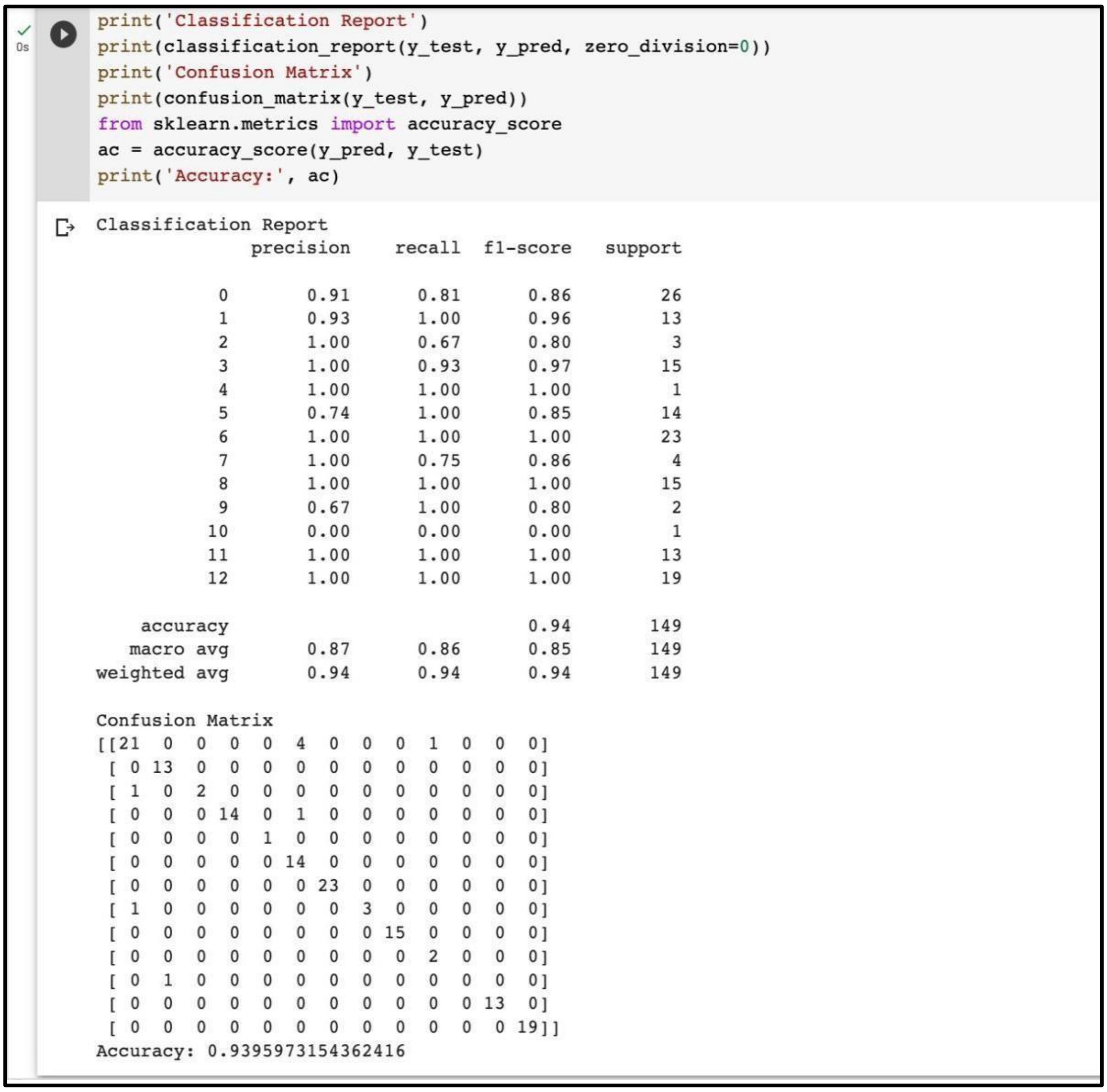


```
[402] from sklearn.ensemble import RandomForestClassifier
      model = RandomForestClassifier(n_estimators=200,min_samples_split=3,
                                     max_features = 'sqrt')
```

```
model.fit(x_train, y_train)
```

```
▾              RandomForestClassifier
RandomForestClassifier(min_samples_split=3, n_estimators=200)
```

Fig 0.9: Usage of sci-kit ensemble model



```
print('Classification Report')
print(classification_report(y_test, y_pred, zero_division=0))
print('Confusion Matrix')
print(confusion_matrix(y_test, y_pred))
from sklearn.metrics import accuracy_score
ac = accuracy_score(y_pred, y_test)
print('Accuracy:', ac)
```

```
Classification Report
              precision    recall  f1-score   support

           0       0.91      0.81      0.86        26
           1       0.93      1.00      0.96        13
           2       1.00      0.67      0.80         3
           3       1.00      0.93      0.97        15
           4       1.00      1.00      1.00         1
           5       0.74      1.00      0.85        14
           6       1.00      1.00      1.00        23
           7       1.00      0.75      0.86         4
           8       1.00      1.00      1.00        15
           9       0.67      1.00      0.80         2
          10       0.00      0.00      0.00         1
          11       1.00      1.00      1.00        13
          12       1.00      1.00      1.00        19

    accuracy                           0.94       149
   macro avg       0.87      0.86      0.85       149
weighted avg       0.94      0.94      0.94       149

Confusion Matrix
[[21  0  0  0  0  4  0  0  0  1  0  0  0]
 [ 0 13  0  0  0  0  0  0  0  0  0  0  0]
 [ 1  0  2  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0 14  0  1  0  0  0  0  0  0  0]
 [ 0  0  0  0  1  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 14  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 23  0  0  0  0  0  0]
 [ 1  0  0  0  0  0  0  3  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0 15  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  2  0  0  0]
 [ 0  1  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0 13  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0 19]]
Accuracy: 0.9395973154362416
```

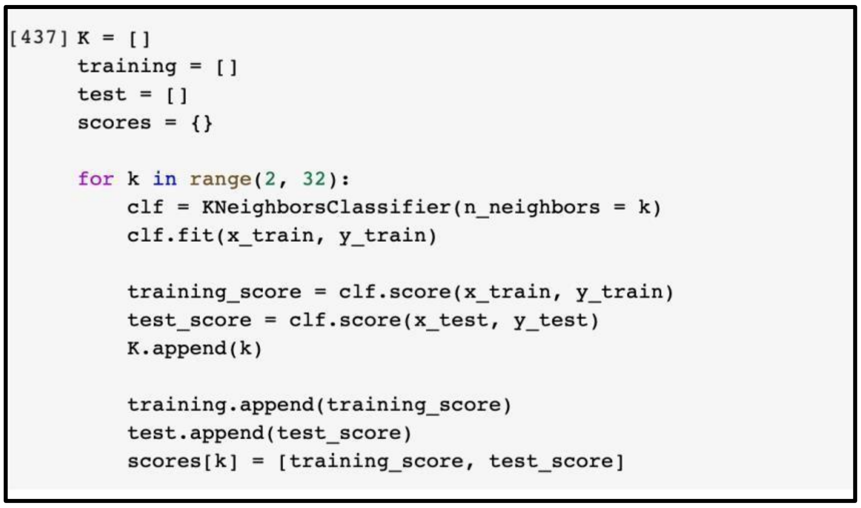
Fig 0.10: Random Forest algorithm report

The accuracy with this model touches almost 94% and is considered highly accurate in the space of cricket prediction. This model is accepted as the most accurate and appears to be a stagnant maxima at this accuracy.

K-Nearest Neighbours:

When the decision boundary is very non-linear and the data has a distinct class border, the KNN (K-Nearest Neighbors) approach can be useful. As it makes no assumptions regarding the distribution of the data, it is also helpful when there is no prior knowledge of that distribution.

KNN classification algorithm was imported from Sci-kit library. To find the optimal value of K, the number of clusters, an iteration was run from 2 to 32. This data was plotted on a graph to make the judgment of the closest points between test and training data.



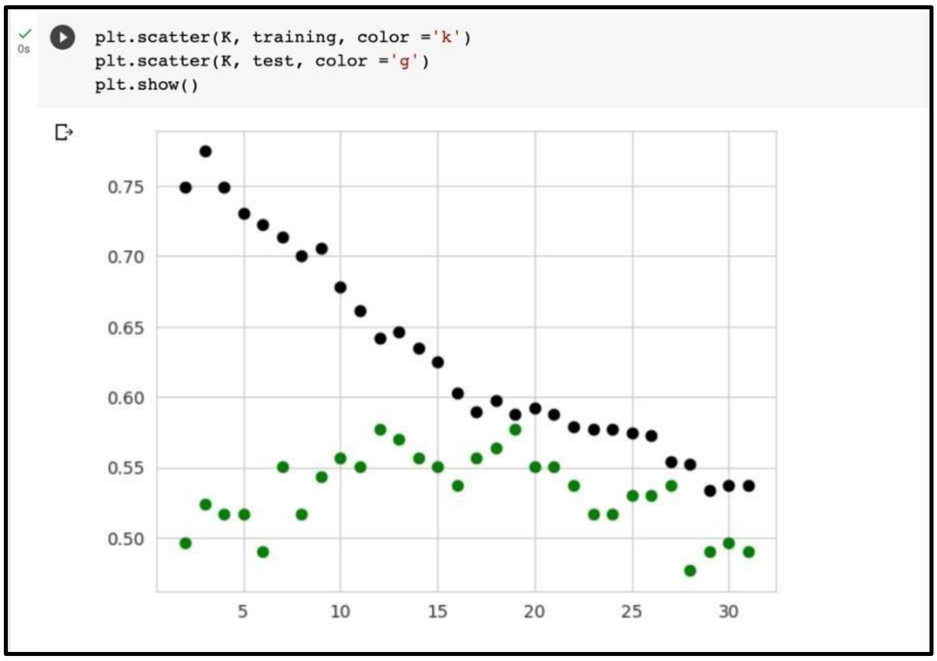
```
[437] K = []
      training = []
      test = []
      scores = {}

      for k in range(2, 32):
          clf = KNeighborsClassifier(n_neighbors = k)
          clf.fit(x_train, y_train)

          training_score = clf.score(x_train, y_train)
          test_score = clf.score(x_test, y_test)
          K.append(k)

          training.append(training_score)
          test.append(test_score)
          scores[k] = [training_score, test_score]
```

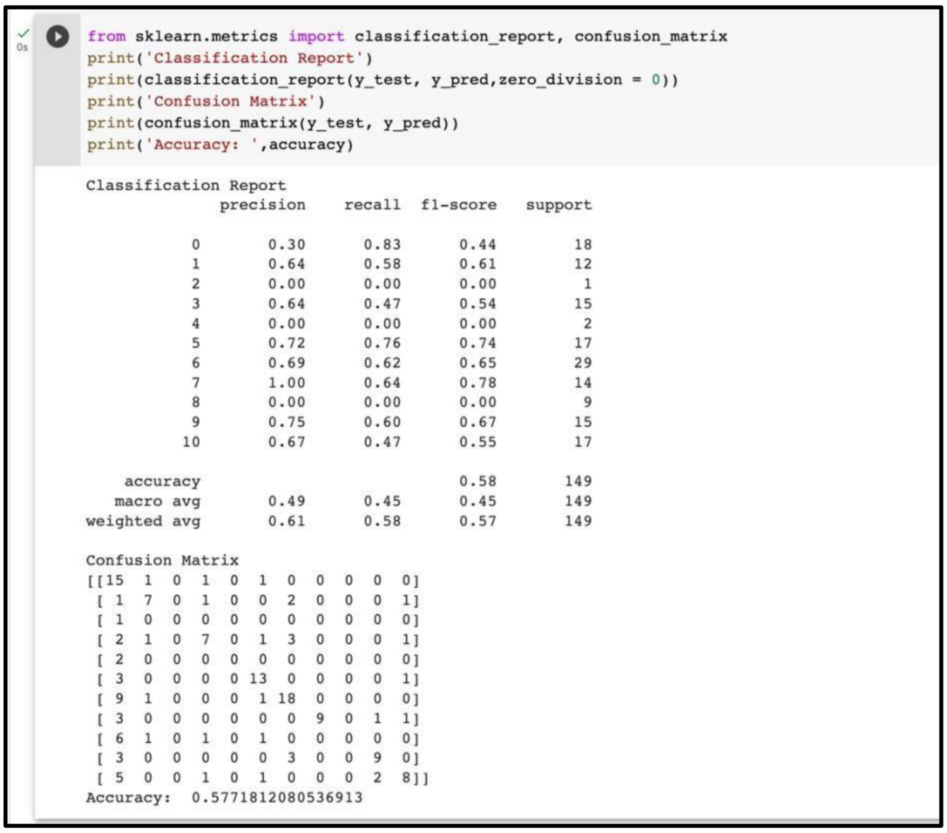
Fig 0.11: Usage of sci-kit KNN algorithm



```
plt.scatter(K, training, color ='k')
plt.scatter(K, test, color ='g')
plt.show()
```

Fig 0.12: Test v/s Train set scatter plot to obtain K

Hence, the intersecting point was found to be around a hyperparameter of K=19, thus giving the following results.



```python
from sklearn.metrics import classification_report, confusion_matrix
print('Classification Report')
print(classification_report(y_test, y_pred,zero_division = 0))
print('Confusion Matrix')
print(confusion_matrix(y_test, y_pred))
print('Accuracy: ',accuracy)
```

```
Classification Report
              precision    recall  f1-score   support

           0       0.30      0.83      0.44        18
           1       0.64      0.58      0.61        12
           2       0.00      0.00      0.00         1
           3       0.64      0.47      0.54        15
           4       0.00      0.00      0.00         2
           5       0.72      0.76      0.74        17
           6       0.69      0.62      0.65        29
           7       1.00      0.64      0.78        14
           8       0.00      0.00      0.00         9
           9       0.75      0.60      0.67        15
          10       0.67      0.47      0.55        17

    accuracy                           0.58       149
   macro avg       0.49      0.45      0.45       149
weighted avg       0.61      0.58      0.57       149

Confusion Matrix
[[15  1  0  1  0  1  0  0  0  0  0]
 [ 1  7  0  1  0  0  2  0  0  0  1]
 [ 1  0  0  0  0  0  0  0  0  0  0]
 [ 2  1  0  7  0  1  3  0  0  0  1]
 [ 2  0  0  0  0  0  0  0  0  0  0]
 [ 3  0  0  0  0 13  0  0  0  0  1]
 [ 9  1  0  0  0  1 18  0  0  0  0]
 [ 3  0  0  0  0  0  0  9  0  1  1]
 [ 6  1  0  1  0  1  0  0  0  0  0]
 [ 3  0  0  0  0  0  3  0  0  9  0]
 [ 5  0  0  1  0  1  0  0  0  2  8]]
Accuracy:  0.5771812080536913
```

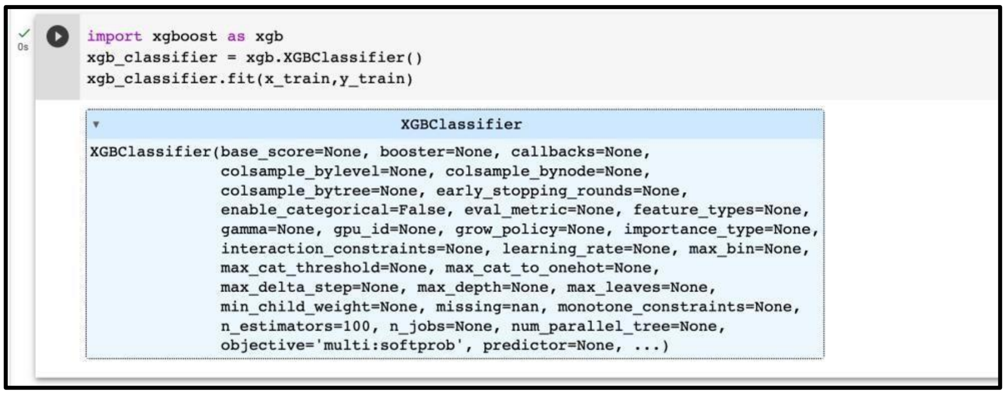
Fig 0.13: KNN classification report

The accuracy obtained from KNN classification algorithm was found to be below acceptable threshold as it was making the right predictions only about 60% of the time.

XGBoost (Extreme Gradient Boosting):

All things considered, XGBoost is useful when great prediction power and accuracy are needed, especially for huge datasets or high-dimensional data with non-linear correlations. It is a flexible and effective method that has a wide range of uses in data science and machine learning.

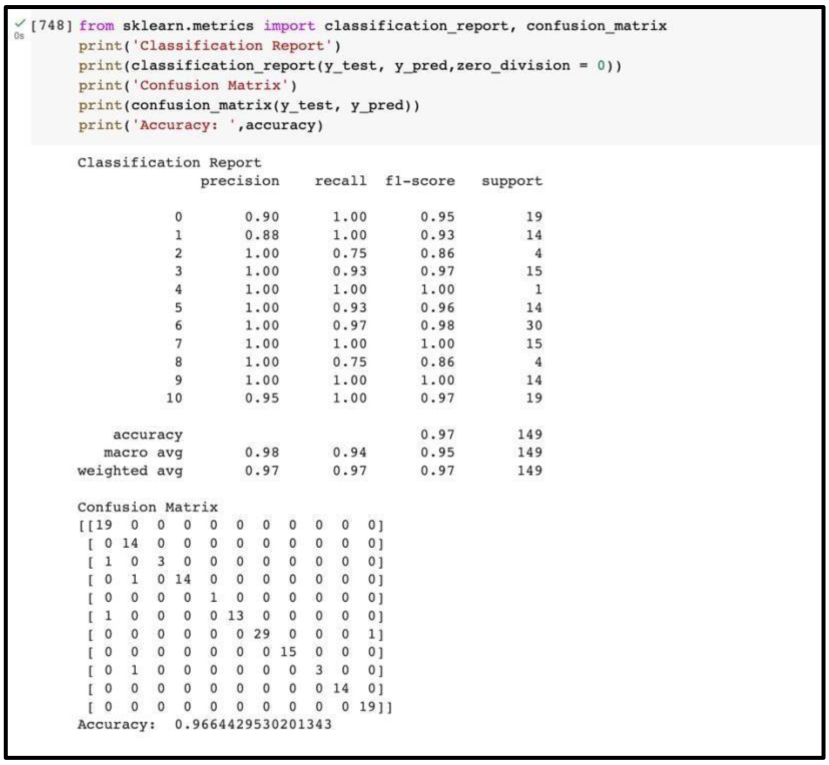
The algorithm was obtained from XGB open source library.



```
import xgboost as xgb
xgb_classifier = xgb.XGBClassifier()
xgb_classifier.fit(x_train,y_train)
```

```
XGBClassifier
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              n_estimators=100, n_jobs=None, num_parallel_tree=None,
              objective='multi:softprob', predictor=None, ...)
```

Fig 0.14: Usage of XGB library for xgboost algorithm



```
[748] from sklearn.metrics import classification_report, confusion_matrix
      print('Classification Report')
      print(classification_report(y_test, y_pred,zero_division = 0))
      print('Confusion Matrix')
      print(confusion_matrix(y_test, y_pred))
      print('Accuracy: ',accuracy)
```

```
Classification Report
              precision    recall  f1-score   support

           0       0.90      1.00      0.95        19
           1       0.88      1.00      0.93        14
           2       1.00      0.75      0.86         4
           3       1.00      0.93      0.97        15
           4       1.00      1.00      1.00         1
           5       1.00      0.93      0.96        14
           6       1.00      0.97      0.98        30
           7       1.00      1.00      1.00        15
           8       1.00      0.75      0.86         4
           9       1.00      1.00      1.00        14
          10       0.95      1.00      0.97        19

    accuracy                           0.97       149
   macro avg       0.98      0.94      0.95       149
weighted avg       0.97      0.97      0.97       149

Confusion Matrix
[[19  0  0  0  0  0  0  0  0  0  0]
 [ 0 14  0  0  0  0  0  0  0  0  0]
 [ 1  0  3  0  0  0  0  0  0  0  0]
 [ 0  1  0 14  0  0  0  0  0  0  0]
 [ 0  0  0  0  1  0  0  0  0  0  0]
 [ 1  0  0  0  0 13  0  0  0  0  0]
 [ 0  0  0  0  0  0 29  0  0  0  1]
 [ 0  0  0  0  0  0  0 15  0  0  0]
 [ 0  1  0  0  0  0  0  0  3  0  0]
 [ 0  0  0  0  0  0  0  0  0 14  0]
 [ 0  0  0  0  0  0  0  0  0  0 19]]
Accuracy:  0.9664429530201343
```

Fig 0.15: XGBoost algorithm report

The accuracy obtained was astounding from XGBoost. It was found to be highly accurate from a range of 95% upto 97% on account of dataset shuffling and randomization. This model delivered the highest accuracy and is chosen best for the problem statement.

RESULTS

Models	Precision	Recall	f1-score	Accuracy
Logistic Regression	0.53	0.55	0.53	55.03%
Decision Tree	0.93	0.92	0.92	91.95%
Random Forest (Ensemble)	0.94	0.94	0.94	93.96%
K-Nearest Neighbours	0.61	0.58	0.57	57.72%
XGBoost (Extreme Gradient Boosting)	0.97	0.97	0.97	96.64%

ANALYSIS

The results of the IPL match prediction project indicate notable variations in the accuracy of the models. Logistic Regression and K-Nearest Neighbours demonstrate lower accuracies of 55.03% and 57.72%, respectively. These models may struggle to capture the complex relationships and nonlinear patterns inherent in IPL match data.

On the other hand, Decision Tree performs significantly better, achieving an accuracy of 91.95%. Decision Trees excel in capturing complex interactions among features, making them suitable for IPL match predictions. However, there is a potential risk of overfitting to the training data.

The Random Forest model improves upon the performance of Decision Tree, reaching an accuracy of 93.96%. By combining multiple decision trees through ensemble learning, Random Forests mitigate overfitting and enhance accuracy, proving to be an effective approach for IPL match predictions.

The highest accuracy is achieved by XGBoost at 96.64%. XGBoost employs extreme gradient boosting, harnessing the power of ensemble methods and effectively capturing nonlinear patterns in the data. This model demonstrates the strongest predictive capabilities among the evaluated models.

In conclusion, the analysis reveals that XGBoost outperforms all other models, with an accuracy of 96.64%, making it the most accurate model for IPL match predictions. The Random Forest model also performs well, achieving an accuracy of 93.96%. Decision Tree shows promising results, while Logistic Regression and K-Nearest Neighbors exhibit lower accuracies. The findings highlight the importance of selecting the appropriate model for IPL match prediction tasks.

REFERENCES

[1] Vistro, Daniel, Leo Gertrude David, “The cricket winner prediction with application of machine learning and data analytics,” International Journal of Scientific and Technology Research, Volume 8, Issue 09, 2019.

[2] Jhanwar, Vikram, “Predicting the Outcome of ODI Cricket Matches: A Team Composition Based Approach," International Institution of Information Technology, Hyderabad, 2016.

[3] Lokhande, Chawan, Pramila, “Prediction of Live Cricket Score and Winning,” International Journal of Trend in Research and Development, Volume 5(1), (2018).

[4] Jaishankar, Rajkumar, “A review paper on cricket predictions using various machine learning algorithms and comparisons among them,” International Journal for Research in Applied Science and Engineering Technology, 2018.

[5] Rory, Fadi. “A Machine Learning Framework for Sport Result Prediction,” Applied Computing and Informatics, Volume 15, Issue 1 2017.

[6] Jayanth, Sandesh Bananki, Akas Anthony, Gududuru Abhilasha, Noorni , Gowri Srinivasa, “A team recommendation system and outcome prediction for the game of cricket,” Journal of sports analytics, vol 4, pp. 263-273, 2018.

[7] Akhil, Venkata, Venkatesh, Sai, Chavali, "Cricket score and winning prediction using data mining," International journal of advance research idea and innovations in technology, 2018.

[8] Tejinder, Vishal, Parteek, “Score and Winning Prediction in Cricket through Data Mining,” International Conference on Soft Computing Techniques and Implementations, 2015.

[9] Stylianos, William, “Using machine learning to predict the outcome of English county twenty over cricket matches,” Cornell university, 2015.

[10] Raj, Padma, “Application of association rule mining: A case study on team India,” 2013 International Conference on Computer Communication and Informatics, Coimbatore, India, January 2013.

[11] Siddharth Sinha “IPL Win Prediction System To Improve Team Performance using SVM” IJFGCN Vol. 13, (2020).

[12] Rabindra Lamsal and Ayesha Choudhary “Predicting Outcome of Indian Premier League (IPL) Matches Using Machine Learning” ResearchGate (2020).

[13] Rajesh Goel “Dynamic Cricket match outcome prediction” Journal of Sports Analytics 2021.

[14] Pabitra Kumar Dey, Gangotri Chakraborty, Purnendu Ruj and Suvobrata Sarkar, “A Data Mining Approach on Cluster Analysis of IPL”, International Journal of Machine Learning and Computing, Vol. 2, No. 4, pp. 351-354, 2012.

[15] Raza Ul Mustafa, M. Saqib Nawaz, M. Ikram Ullah Lali, Tehseen Zia and Waqar Mehmood, “Predicting the Cricket Match outcome using Crowd Opinions on Social Networks: A Comparative Study of Machine Learning Methods”, Malaysian Journal of Computer Science, Vol. 30, No. 1, pp. 63-76, 2017.

[16] Sanjay Gupta, Hitesh Jain, Asmit Gupta and Hemant Soni, “Fantasy League Team Prediction”, International Journal of Research in Science and Engineering, Vol. 6, No. 3, pp. 97- 103, 2017.

[17] Vignesh Veppur Sankaranarayanan and Junaed Sattar,“Auto-play: A Data Mining Approach to ODI Cricket Simulation and Prediction”, Proceedings of SIAM Conference on Data Mining, pp. 1-7, 2014.

[18] Parag Shah, “Predicting Outcome of Live Cricket Match using Duckworth-Lewis Par Score”, International Journal of Latest Technology in Engineering, Management and Applied Science, Vol. 6, No. 7, pp. 72-75, 2017.

[19] Amal Kaluarachchi and S. Aparna, “A Classification based Tool to Predict the outcome in ODI Cricket”, Proceedings of 5 th International Conference on Information and Automation for Sustainability, pp. 233-237, 2010.

[20] C. Deep Prakash Dayalbagh, C. Patvardhan and C. Vasantha Lakshmi, “Data Analytics based Deep Mayo Predictor for IPL-9”, International Journal of Computer Applications, Vol. 152, No. 6, pp. 6-11, 2016.

[21] Jayshree Hajgude, Aishwarya Parameshwaran, Krishna Nambi, Anupama Sakhalkar and Darshil Sanghvi, “IPL Dream Team-A Prediction Software Based on Data Mining and Statistical Analysis”, International Journal of Computer Engineering and Applications, Vol. 9, No. 4, pp. 113-119, 2015.

[22] Sonu Kumar and Sneha Roy, “Score Prediction and Player Classification Model in the Game of Cricket using Machine Learning”, International Journal of Scientific and Engineering Research, Vol. 9, No. 2, pp. 237-242, 2018.

[23] Rabindra Lamsal and Ayesha Choudhary, “Predicting Outcome of Indian Premier League (IPL) Matches Using Machine Learning”, arXiv:1809.09813 [stat.AP] (September 2018)

[24] Monali Shetty, Sankalp Rane, Chaitanya Pandita, Suyash Salvi, Machine learning- based Selection of Optimal sports Team based on the Players Performance, Proceedings of the Fifth International Conference on Communication and Electronics Systems (ICCES 2020), IEEE Conference Record, ISBN: 978-1-7281-5371-1, 2020.

[25] Prince Kansal, Pankaj Kumar, Himanshu Arya, Aditya Methaila, Player valuation in Indian premier league auction using data mining technique, International Conference on Contemporary Computing and Informatics (IC3I), 2729 Nov 2014