VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

PROJECT REPORT

DATA MINING TECHNIQUES [SWE2009]

Analysis and Knowledge Discovery from Cricket Data

Team Members:

Registration Number	**Name**
18MIS1029	Vaishnavi S
18MIS1059	Balaji Chandramouli

Acknowledgements
We are profoundly grateful to Dr. Saleena B for her expert guidance and continuous encouragement throughout to see that this project reaches its target from its commencement to its completion.

Abstract
With the rise of the Indian Premier League in 2008, cricket became a worldwide phenomenon as it continued to receive millions in viewership. As of 2019, the IPL has been valued at $6.7 billion. With so much at stake and every ball under scrutiny, technology is being used extensively to make the players plan and play better.. This project aims to analyze this large scale unstructured data and extract some useful information and solutions for the team management to identify the top players and assess the performances of the players and make the right choices in the upcoming seasons.

Introduction about the Project
Cricket is a game of numbers - runs, wickets, averages, strike rate, required rate and so on, the aggregation and analytics of data is a crucial part of the team's preparation. Insights into each player's strengths, weaknesses and even analysing the opposition players is important. Hence, this project will focus on such analysis of the teams, players and ball by ball data of all the editions of the IPL to gather insights and interesting statistics.The data is analyzed using python and uses the following 6 algorithms :- Gaussian Naive Bayes Classification, Decision Tree Classification, Random Forest Classification, KNN Classification, Logistic Regression and K Means Clustering to extract meaningful output.

Datasets used
IPL Ball-by-Ball 2008-2020.csv (Size = 193618) https://cricsheet.org/
Contains the ball by ball data of the whole Indian Premier League tournament.
Columns: match_id, season,start_date,venue,innings,ball,batting_team, bowling_team,striker,non_striker,bowler,runs_off_bat,extras,wides,no balls,byes, leg byes,penalty,wicket_type,player_dismissed,other_wicket_type,other_player_dismissed

IPL Matches 2008-2020.csv (Size = 817)
https://www.kaggle.com/patrickb1912/ipl-complete-dataset-20082020
Contains summary for every match in the tournament.
Columns :
Id, city, date, player_of_match, venue, neutral_venue, team1, team2, toss_winner, toss_decision, winner, runs, result, result_margin, eliminator, method, umpire1, umpire2

Analysis to be done

- If CSK and MI play at M.A Chidambaram Stadium, who is more likely to win when CSK has won the toss and chose to bat first? What is the distribution of batsmen data when it comes to runs scored and strike rate? (K Means Clustering)
- Is 185 runs a defendable/achievable total against KKR at Eden Gardens by DC? (Logistic Regression)
- If Suresh Raina scores 73 runs in a match where his team scores 160 runs, will he be regarded as Man of the Match?(K-Nearest Neighbours)
- Given the venue and toss details of a match which team is more likely to win (Gaussian Naive Bayes, Decision Tree and Random Forest)

Algorithms used

Naive-Bayes Classifier

A Naive Bayes classifier is a probabilistic machine learning model that's used for classification tasks. The main requirement of this algorithm is that the features should be independent. The following formula can be applied to calculate the probability of an event happening or not happening based on a number of features.

$$P(y|x_1, \ldots, x_n) = \frac{P(x_1|y)P(x_2|y)\ldots P(x_n|y)P(y)}{P(x_1)P(x_2)\ldots P(x_n)}$$

In cricket, there are certain pre-conditions of a match that may be useful in predicting which team has the upper hand in winning a match. The conditions are venue, pitch type, toss, toss decision, the playing XI and so on. Some of the features are selected and provided to the Naive Bayes Classifier to predict which team will win the match. This algorithm was chosen because of the fact it is one of the oldest methods used in solving problems like this and to compare the results with newer algorithms.

Decision Tree Classifier

Decision Tree is a supervised learning technique that can be used for both classification and Regression problems.It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome. Similar to using Naive-Bayes Classifier, this algorithm is used to classify the winner of a match based on certain conditions.

Random Forest Classifier

Random forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time. It takes multiple bootstrapped datasets from the initial dataset, individually prepares a decision tree for each bootstrapped dataset, and performs classification of the test data in each decision tree and for classification tasks, the output of the random forest is the class selected by most trees. Since this algorithm is an ensemble of decision tree classifiers, it is used to perform the classification of the winner of a match based on certain conditions.

Logistic Regression

Logistic Regression is a machine learning algorithm that uses the Sigmoid Function to calculate the probability of an event happening or not(0 or 1). It takes discrete (numerical) values as input, and gives a curve ranging from 0 to 1 based on input as output from which we can assign the threshold probability to classify it as the event happening or not. Using this algorithm, we can have the first innings score made by teams batting on a certain venue as input, and a binary value of winning the match or not as the output. Using this information, we can calculate odds, log of odds, and comparative increase in chances based on the runs scored.

K Means Clustering

K means algorithm identifies k number of centroids by averaging the data and then allocates every data to its nearest cluster. In the game of cricket the players are recognized either as a batsman or a bowler based on their performances. A player who performs well as a batsman, bowler and fielder is usually called an all rounder. Clustering the players as batsmen and bowlers will help us identify the top players as well as see their distribution when it comes to their runs scored. Additionally we can group the players as Ideal, Floater, Slogger and Bad (player is more likely to be a bowler) etc. Similarly bowlers can be ranked based on their economy rate. Thus K means clustering can be used to identify and analyse the star players.

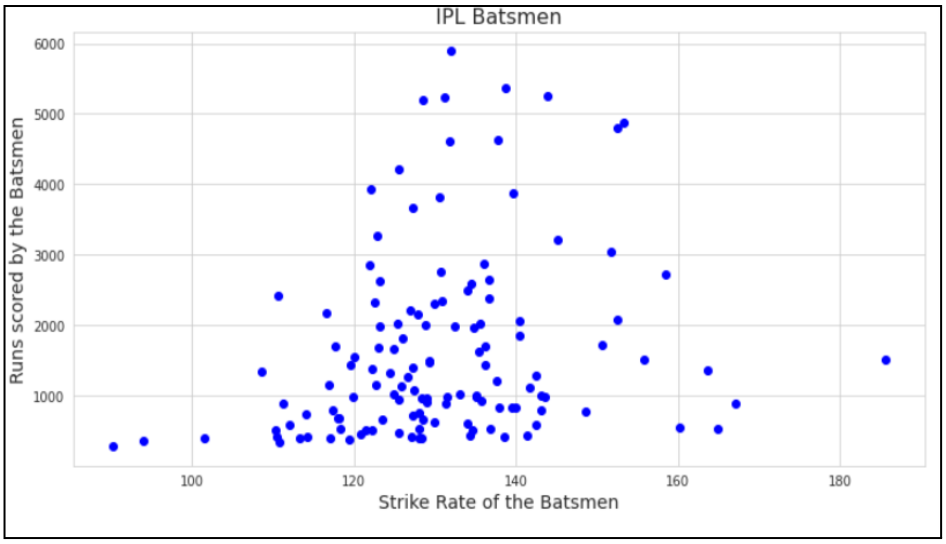
KNN Classification

KNN stands for k nearest neighbours and it assumes that similar things exist in close proximity. KNN can perform both classification and regression. It learns a function using labeled input data and produces an appropriate output if given a new unlabeled data. KNN algorithm basically calculates the distances between 2 points and classifies it similar to the closest point. KNN classification uses various distance metrics like Euclidean, Minowski, Manhattan etc. For our project we have trained the model using KNN with Minowski metric to identify if a player is the man of the match or not when the total runs of his team and his individual contribution is given.

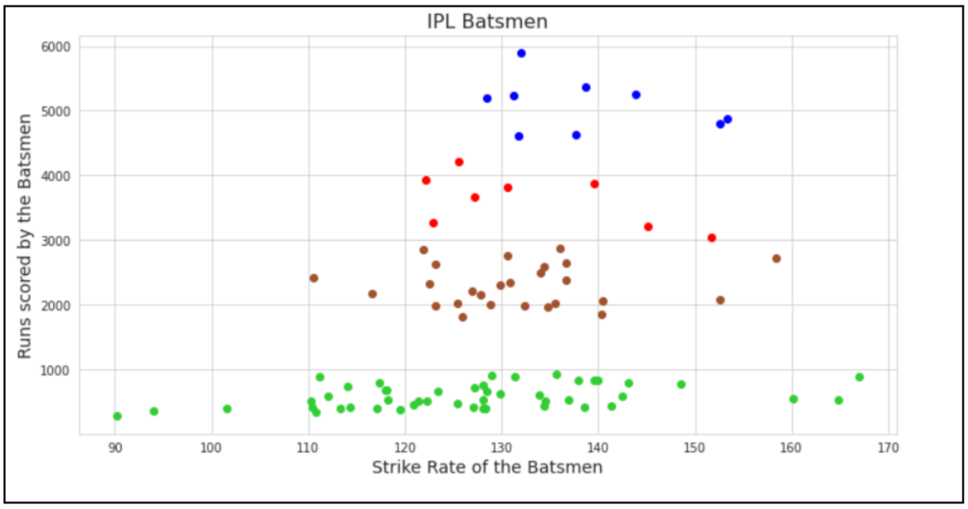
Implementation of each algorithm with screen shots – Proper interpretation of results

K Means Clustering - Clustering Batsmen and Bowlers according to their Performance in IPL

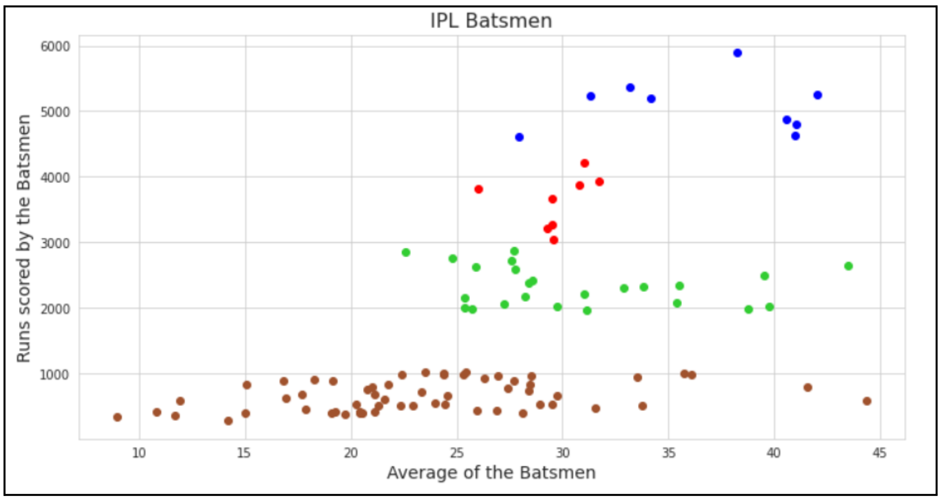
In Order to assess the performance of the batsmen their strike rate is calculated using the formula : Runs scored * 100 / Balls faced. When a scatter plot is made using the strike rate and runs scored the distribution of the Batsmen appears to be in largely four clusters: under 1000 , 2000 - 3000 ,3000 - 4000 & 5000 + runs.



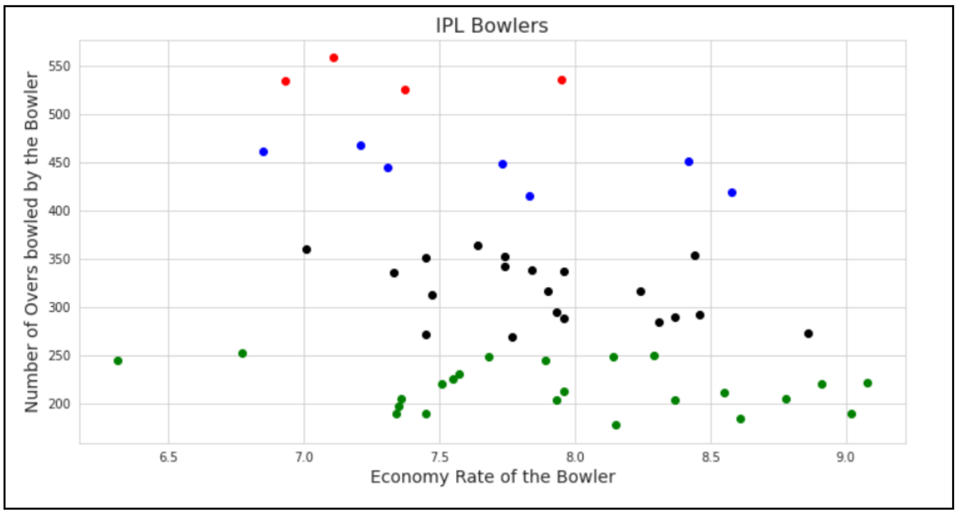
Of these, under 500 seems to have the highest density of dots. Clustering this distribution we get the following results,



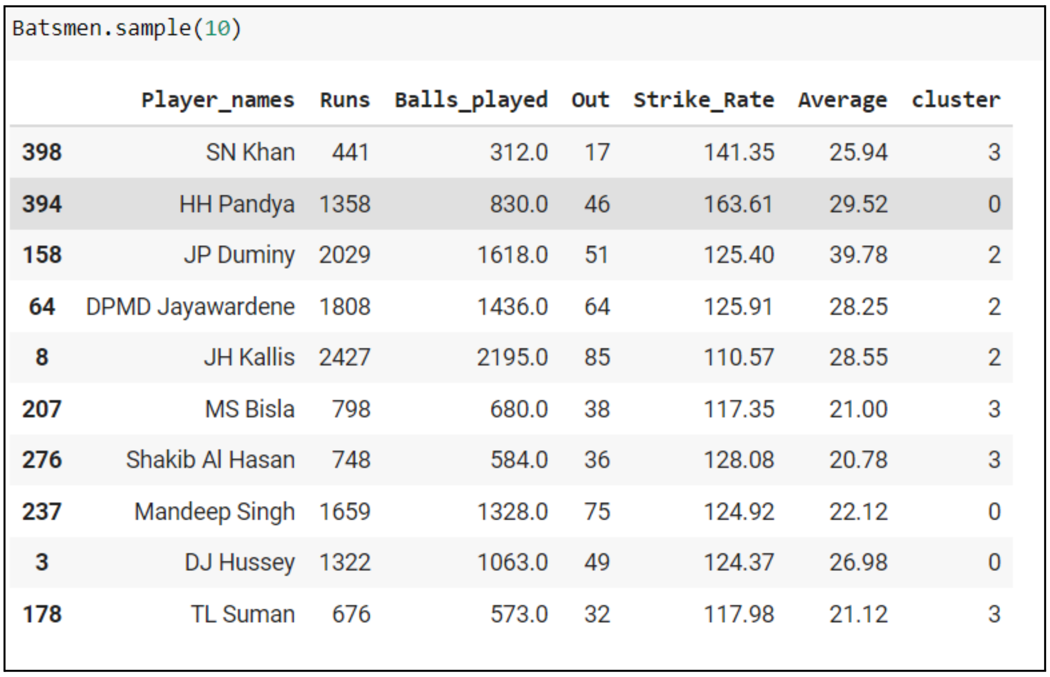
The Strike rate of the batsmen is not a valid metric to assess his performance over all the seasons of IPL. Hence we also use the average of the batsmen to assess their performance. The following distribution shows the clustering using the average of the batsmen. The average is calculated using the formula : Runs / No. of times the batsman was out



Similarly the bowlers were clustered using their economy rate which is calculated by the formula : Runs conceded / Overs bowled



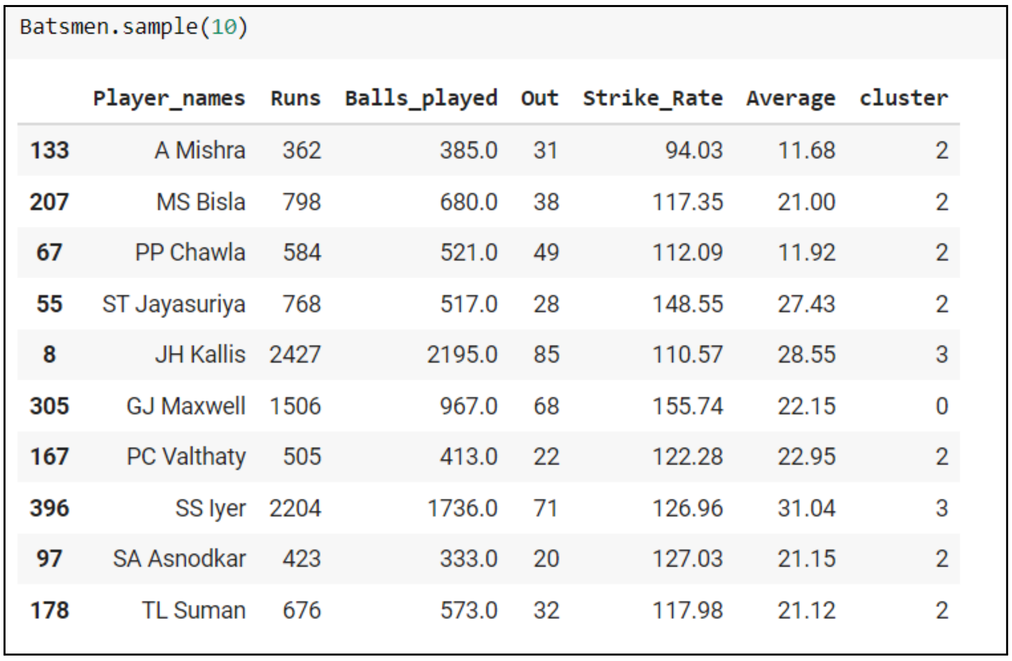
Clusters of 10 sample players - Strike rate



```
Batsmen.sample(10)
```

	Player_names	Runs	Balls_played	Out	Strike_Rate	Average	cluster
398	SN Khan	441	312.0	17	141.35	25.94	3
394	HH Pandya	1358	830.0	46	163.61	29.52	0
158	JP Duminy	2029	1618.0	51	125.40	39.78	2
64	DPMD Jayawardene	1808	1436.0	64	125.91	28.25	2
8	JH Kallis	2427	2195.0	85	110.57	28.55	2
207	MS Bisla	798	680.0	38	117.35	21.00	3
276	Shakib Al Hasan	748	584.0	36	128.08	20.78	3
237	Mandeep Singh	1659	1328.0	75	124.92	22.12	0
3	DJ Hussey	1322	1063.0	49	124.37	26.98	0
178	TL Suman	676	573.0	32	117.98	21.12	3

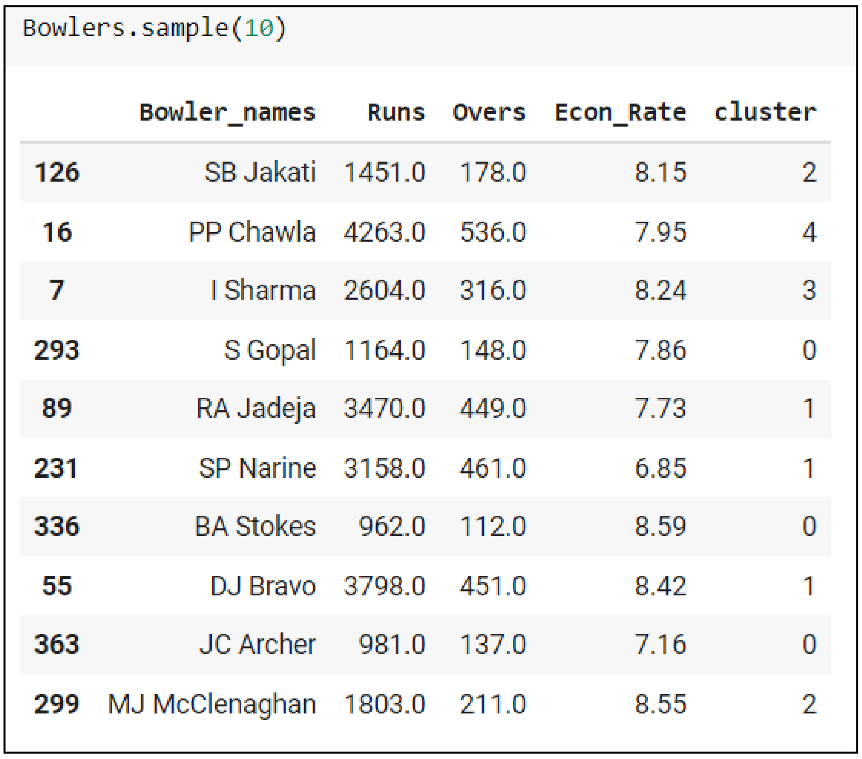
Clusters of 10 sample players - Average



```
Batsmen.sample(10)
```

	Player_names	Runs	Balls_played	Out	Strike_Rate	Average	cluster
133	A Mishra	362	385.0	31	94.03	11.68	2
207	MS Bisla	798	680.0	38	117.35	21.00	2
67	PP Chawla	584	521.0	49	112.09	11.92	2
55	ST Jayasuriya	768	517.0	28	148.55	27.43	2
8	JH Kallis	2427	2195.0	85	110.57	28.55	3
305	GJ Maxwell	1506	967.0	68	155.74	22.15	0
167	PC Valthaty	505	413.0	22	122.28	22.95	2
396	SS Iyer	2204	1736.0	71	126.96	31.04	3
97	SA Asnodkar	423	333.0	20	127.03	21.15	2
178	TL Suman	676	573.0	32	117.98	21.12	2

Clusters of 10 sample players - Economy rate (bowlers)



```
Bowlers.sample(10)
```

	Bowler_names	Runs	Overs	Econ_Rate	cluster
126	SB Jakati	1451.0	178.0	8.15	2
16	PP Chawla	4263.0	536.0	7.95	4
7	I Sharma	2604.0	316.0	8.24	3
293	S Gopal	1164.0	148.0	7.86	0
89	RA Jadeja	3470.0	449.0	7.73	1
231	SP Narine	3158.0	461.0	6.85	1
336	BA Stokes	962.0	112.0	8.59	0
55	DJ Bravo	3798.0	451.0	8.42	1
363	JC Archer	981.0	137.0	7.16	0
299	MJ McClenaghan	1803.0	211.0	8.55	2

KNN algorithm to find if a player is the man of the match or not

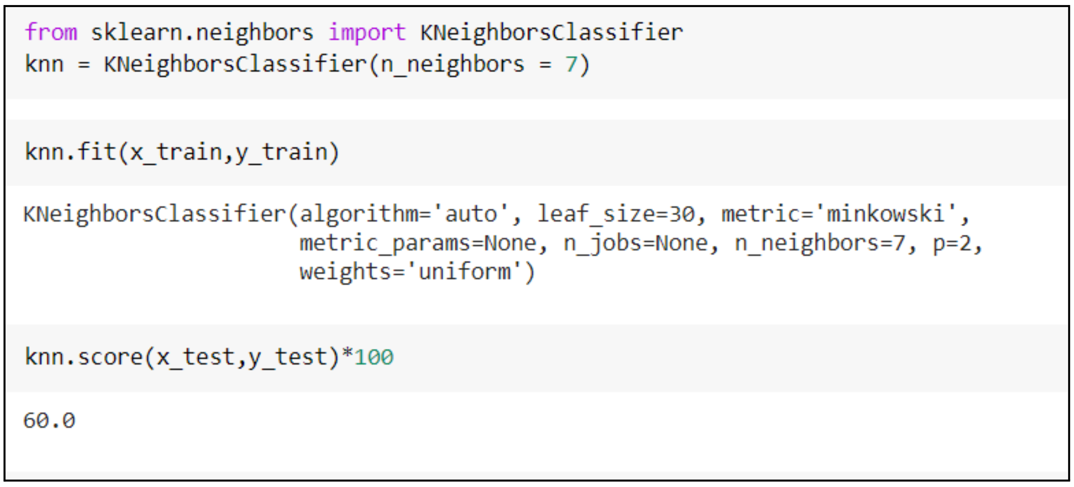
For this particular problem statement a new dataset with 25 records was created with the following data

- Player name
- Runs
- Team runs
- Is MoM (Man of the Match)

- MoM_Label (0 - No ; 1 - Yes)

The same problem was solved manually by hand using the euclidean metric and it was found that k=7 gave a more accurate result than other values. Hence the model was initially trained using the Minowski metric with k=7 neighbours and an accuracy of 60% was seen. To verify the best value of k an error rate graph was plotted and it was found that k=3 had lesser error which meant the accuracy would increase. The model was then retrained using k=3 and the accuracy increased to 80%. The screenshots of the results are attached below.

K=7



```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors = 7)
```

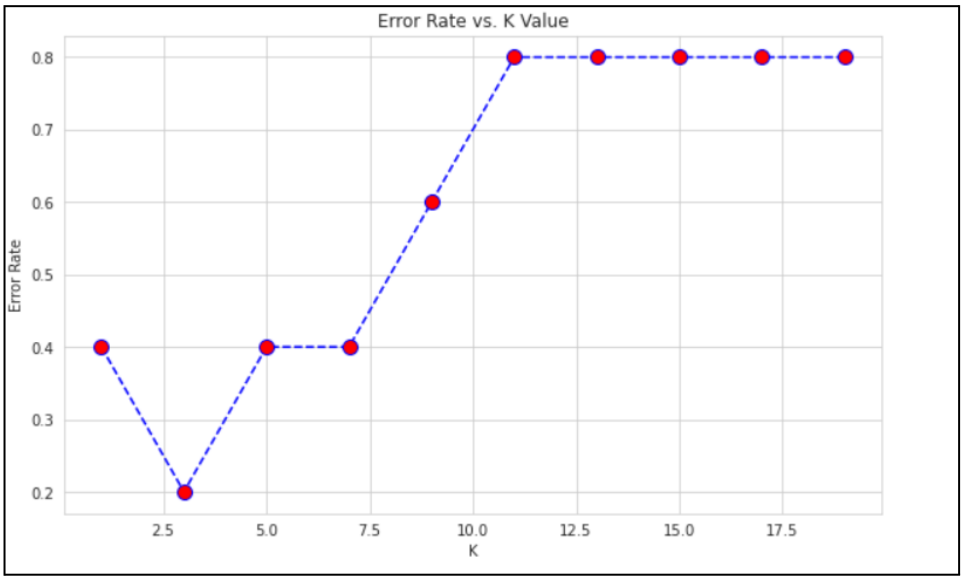
```
knn.fit(x_train,y_train)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=7, p=2,
                     weights='uniform')
```

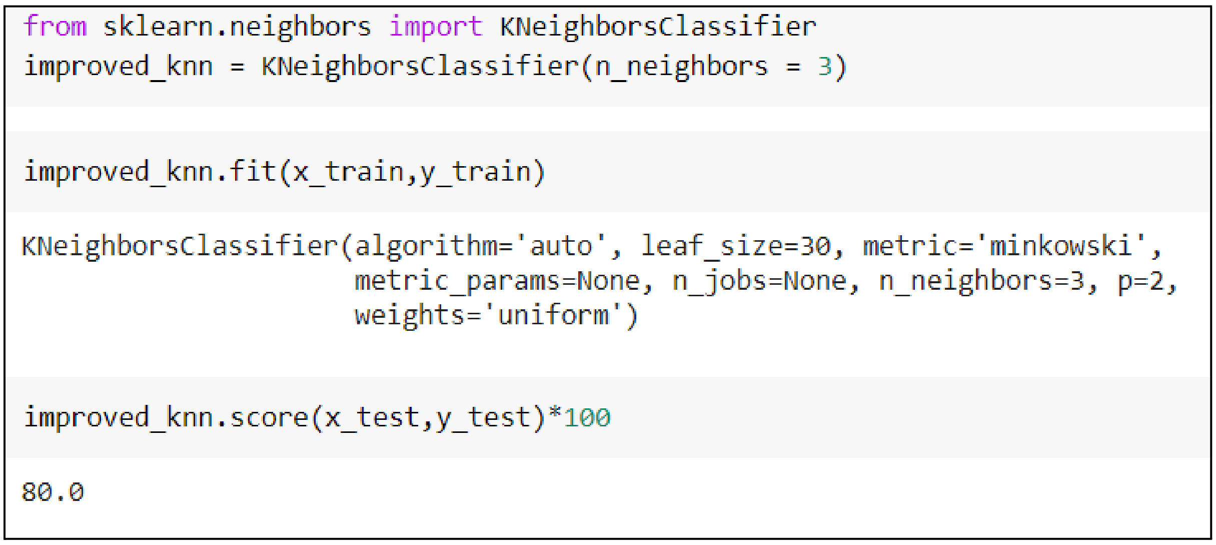
```
knn.score(x_test,y_test)*100
```

```
60.0
```

Error vs K values plot



K=3



```
from sklearn.neighbors import KNeighborsClassifier
improved_knn = KNeighborsClassifier(n_neighbors = 3)
```

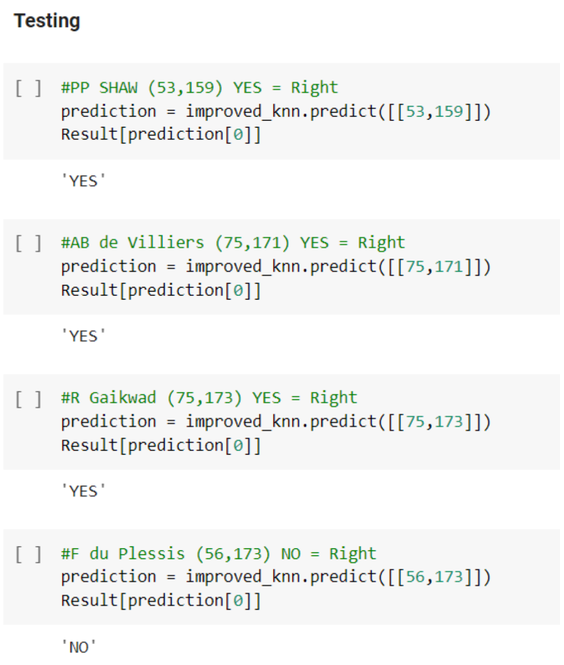
```
improved_knn.fit(x_train,y_train)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=3, p=2,
                     weights='uniform')
```

```
improved_knn.score(x_test,y_test)*100
```

```
80.0
```

Checking the model with a few sample data



Testing

```
[ ] #PP SHAW (53,159) YES = Right
    prediction = improved_knn.predict([[53,159]])
    Result[prediction[0]]
```

```
'YES'
```

```
[ ] #AB de Villiers (75,171) YES = Right
    prediction = improved_knn.predict([[75,171]])
    Result[prediction[0]]
```

```
'YES'
```

```
[ ] #R Gaikwad (75,173) YES = Right
    prediction = improved_knn.predict([[75,173]])
    Result[prediction[0]]
```

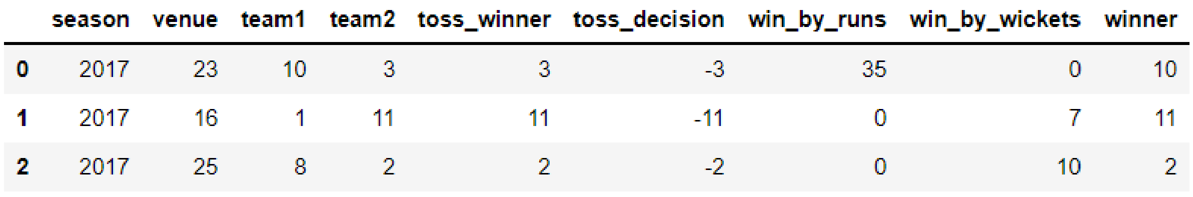
```
'YES'
```

```
[ ] #F du Plessis (56,173) NO = Right
    prediction = improved_knn.predict([[56,173]])
    Result[prediction[0]]
```

```
'NO'
```

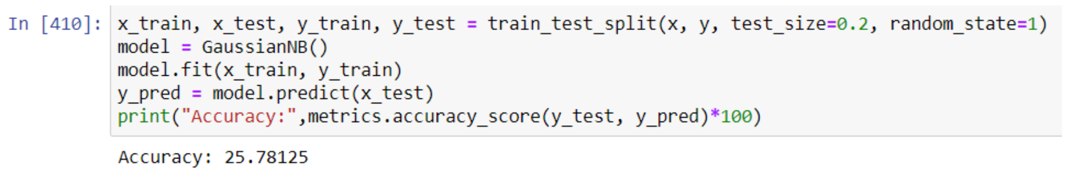
Naive Bayes Classifier to classify winner of the match based on various features

Preprocessed dataset:

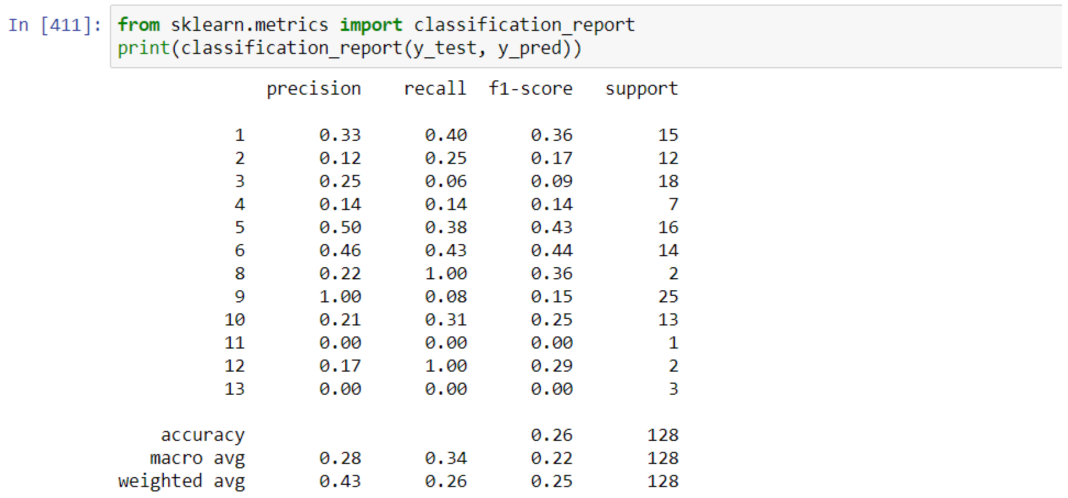


	season	venue	team1	team2	toss_winner	toss_decision	win_by_runs	win_by_wickets	winner
0	2017	23	10	3	3	-3	35	0	10
1	2017	16	1	11	11	-11	0	7	11
2	2017	25	8	2	2	-2	0	10	2

Performance Metrics:



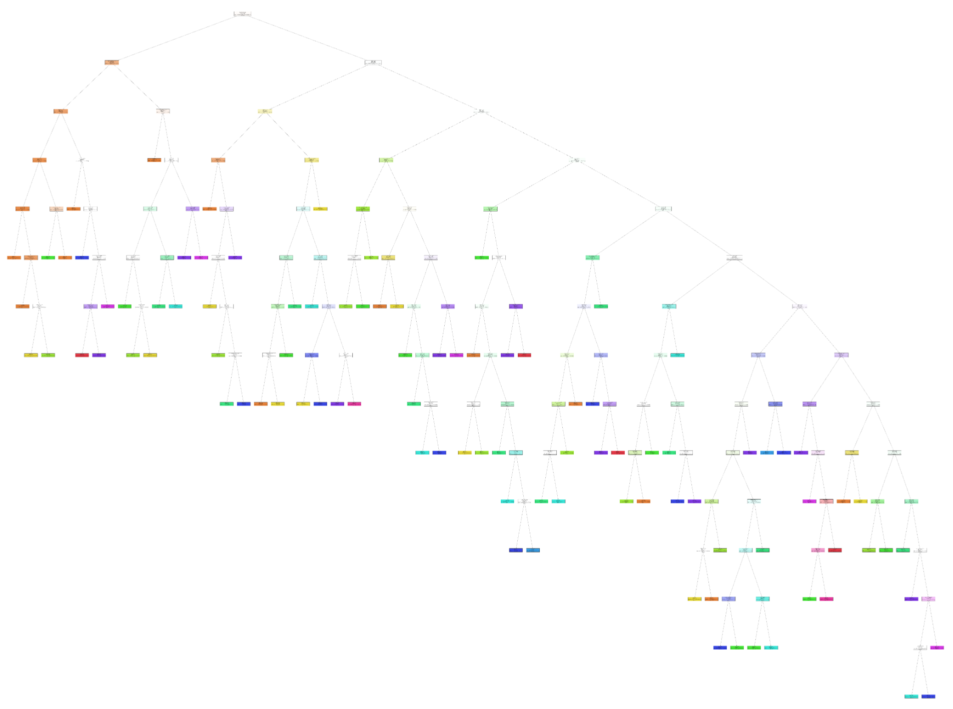
```
In [410]: x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=1)
          model = GaussianNB()
          model.fit(x_train, y_train)
          y_pred = model.predict(x_test)
          print("Accuracy:",metrics.accuracy_score(y_test, y_pred)*100)

          Accuracy: 25.78125
```



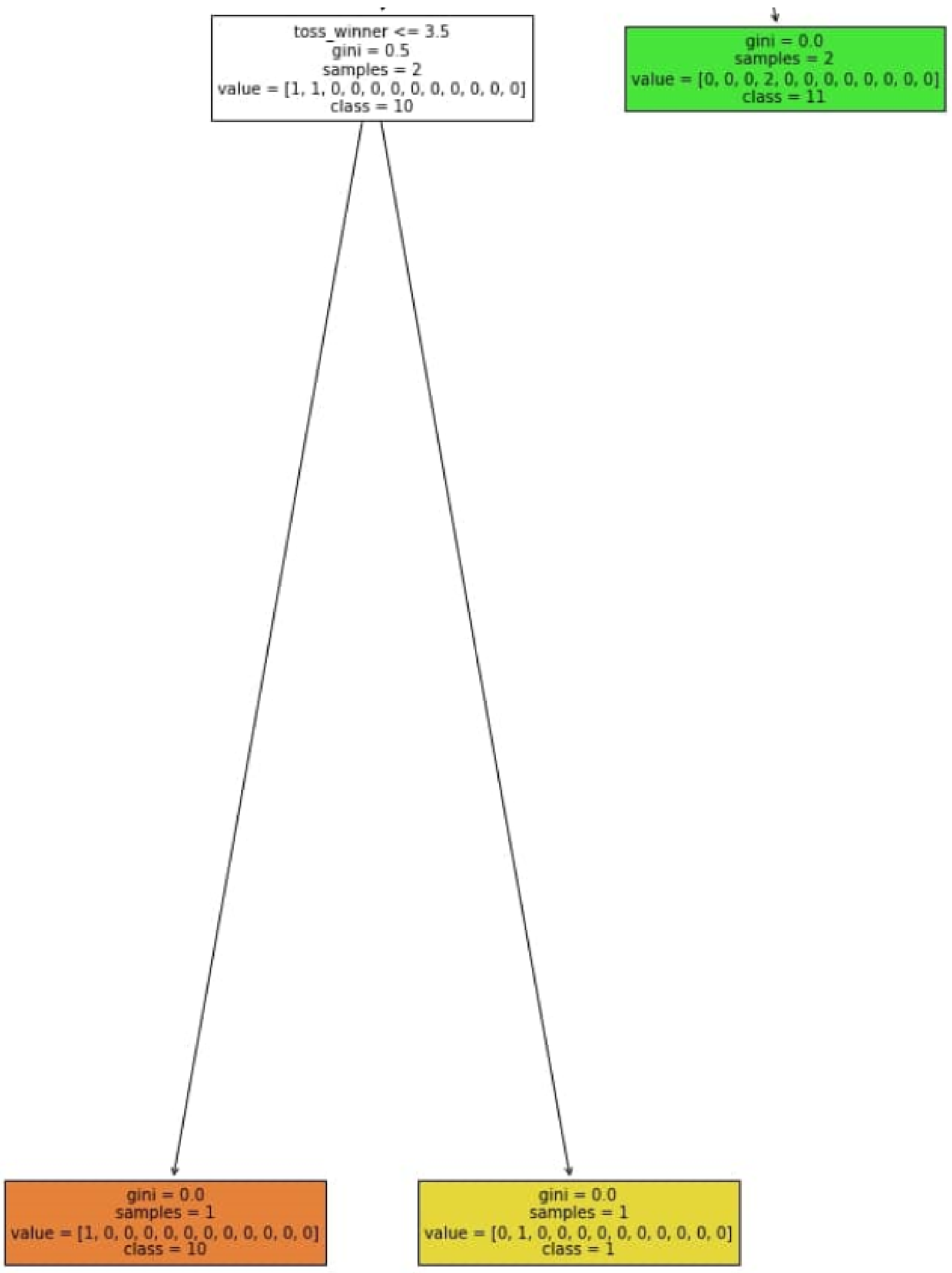
```
In [411]: from sklearn.metrics import classification_report
          print(classification_report(y_test, y_pred))

                        precision    recall  f1-score   support

                     1       0.33      0.40      0.36        15
                     2       0.12      0.25      0.17        12
                     3       0.25      0.06      0.09        18
                     4       0.14      0.14      0.14         7
                     5       0.50      0.38      0.43        16
                     6       0.46      0.43      0.44        14
                     8       0.22      1.00      0.36         2
                     9       1.00      0.08      0.15        25
                    10       0.21      0.31      0.25        13
                    11       0.00      0.00      0.00         1
                    12       0.17      1.00      0.29         2
                    13       0.00      0.00      0.00         3

              accuracy                           0.26       128
             macro avg       0.28      0.34      0.22       128
          weighted avg       0.43      0.26      0.25       128
```

Decision Tree Classifier to classify winner of the match based on various features

Output decision tree:

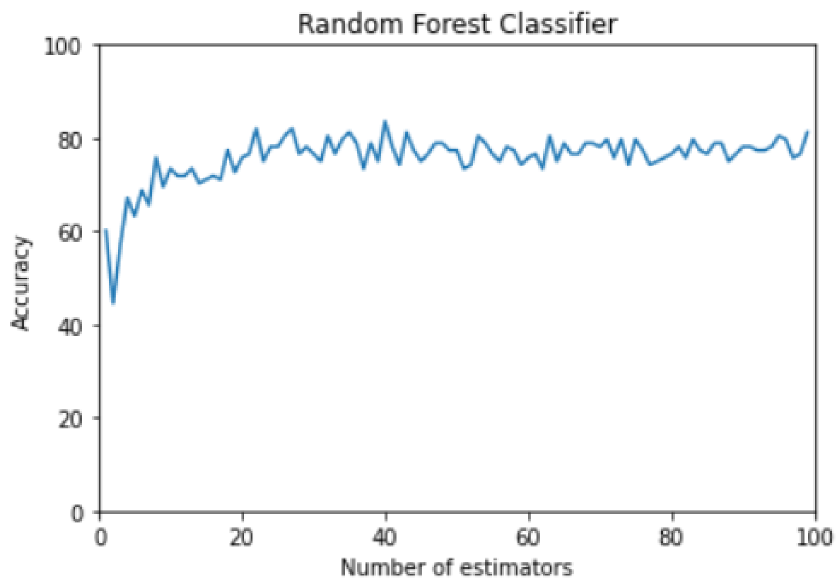


For better visibility:



toss_winner <= 3.5
gini = 0.5
samples = 2
value = [1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
class = 10
gini = 0.0
samples = 2
value = [0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0]
class = 11
gini = 0.0
samples = 1
value = [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
class = 10
gini = 0.0
samples = 1
value = [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
class = 1

Random Forest Classifier to classify winner of the match based on various features
The random forest algorithm was run in a loop from 1 to 100 to solve the problem with n bootstrapped data sets and decision trees and the accuracy for each iteration is calculated and graphed below.

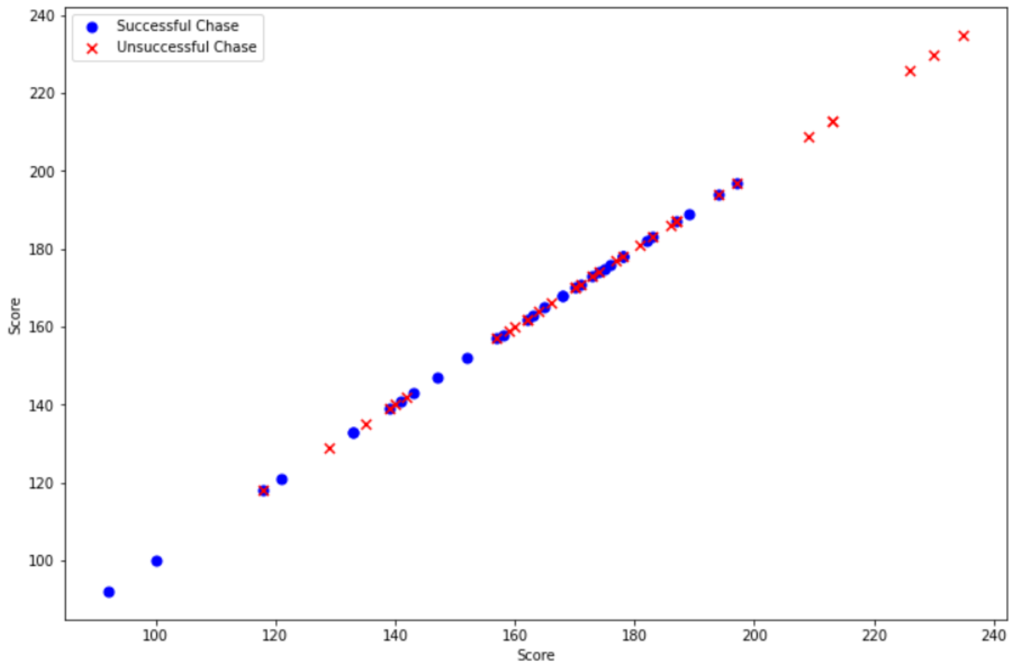


```
Maximum Accuracy: 83.59375  at number of estimators =  40
Minimum Accuracy: 44.53125  at number of estimators =  2
Average Accuracy: 74.9921875
```

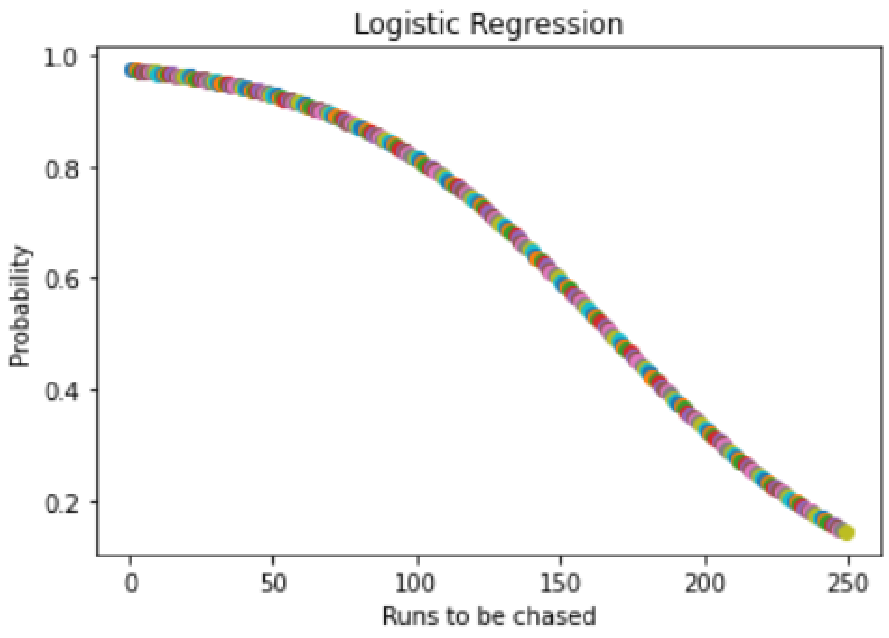
Logistic Regression to calculate the probability and odds of winning a match based on first innings score:

Graphing of first innings score and result of the match:

```
Out[561]: Text(0, 0.5, 'Score')
```



Graphing the data after running it through the sigmoid function to get the probabilities of a successful run chase:

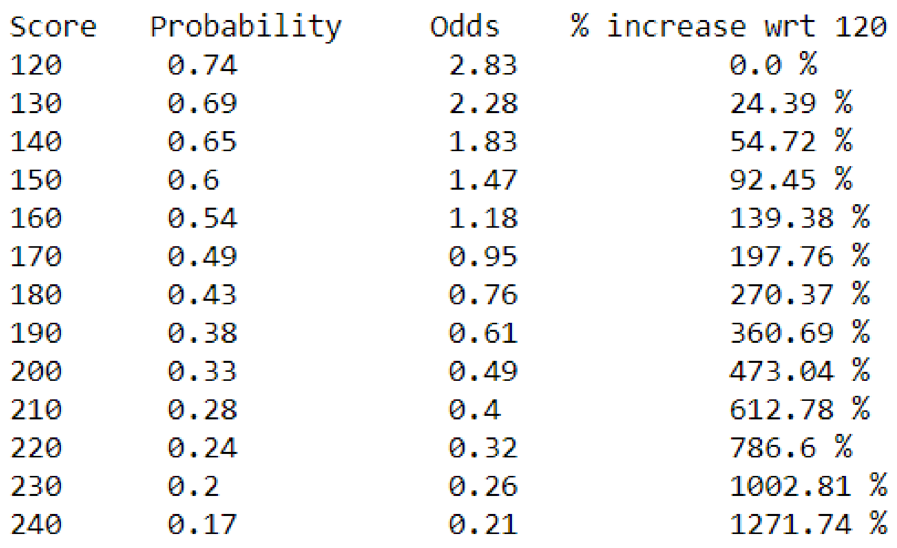


Comparative Analysis:
For the problem of classifying winner of the match based on various features, three algorithms were used and the best accuracies of each algorithm is compared:

Algorithm	Best Accuracy
Naive Bayes Classifier	25.78
Decision Tree Classifier	56.25
Random Forest Classifier	83.59

The results are not surprising as Naive Bayes Classifier is an outdated technique which has many disadvantages over large data sets having many features. Decision Tree Classifier gives a mediocre result but better than Naive Bayes. Random Forest is the best performing algorithm as it is an ensemble technique. It is also interesting to note that Random Forest gives similar accuracy scores when the number of estimators is small.

For the problem of calculating the probabilities of winning a match given the first innings score, the following table was constructed:



```
Score   Probability     Odds    % increase wrt 120
120      0.74            2.83           0.0 %
130      0.69            2.28           24.39 %
140      0.65            1.83           54.72 %
150      0.6             1.47           92.45 %
160      0.54            1.18           139.38 %
170      0.49            0.95           197.76 %
180      0.43            0.76           270.37 %
190      0.38            0.61           360.69 %
200      0.33            0.49           473.04 %
210      0.28            0.4            612.78 %
220      0.24            0.32           786.6 %
230      0.2             0.26           1002.81 %
240      0.17            0.21           1271.74 %
```

The table gives for each score, the probability of winning, the odds of winning and the odds ratio in the form of percentage increase in chance of winning the match with respect to a particular score.

Explain the technologies used and how they are integrated to work

The data analysis was done in Google Colab which is similar to the Jupyter notebook environment. It runs entirely on the cloud and the files are saved as a python notebook (ipynb). Python was used as it offers a wide range of data oriented feature packages and libraries like numpy, pandas etc that can simplify data processing. Numpy is a fundamental library that has multidimensional arrays and tools to manage large data. Pandas is an open source library written in python that has a variety of data structures and tools that help in manipulating numerical data. Scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms along with various performance metrics tools. Apart from these major packages other machine learning tools like matplotlib can also be imported easily in colab and can be helpful in data analysis as well as visualization.

Conclusion and Future Enhancements

The KNN algorithm currently works based on the runs of the batsmen. In cricket the man of the match can also be awarded to bowlers hence the bowling performances can also be included in the dataset and the model can be trained accordingly. The Kmeans clustering has clustered the players as a whole. This can be enhanced by clustering the players team wise and can help the owners during the auction to decide if the players should be retained and build the ideal team accordingly. Various other algorithms such as Xg Boosting, Cat Boost, Ada Boost, Light Gradient Boosting, Extra Tree Classifiers, etc. can be used to make the ensemble learning technique more accurate.

References

H. Barot, A. Kothari, P. Bide, B. Ahir and R. Kankaria, "Analysis and Prediction for the Indian Premier League," 2020 International Conference for Emerging Technology (INCET), Belgaum, India, 2020, pp. 1-7, doi: 10.1109/INCET49848.2020.9153972.

Sample Code

Gauusian Naive Bayes

```
from sklearn.naive_bayes import GaussianNB
x_model, x_test, y_model, y_test = train_test_split(x, y, test_size=0.2, random_state=1)
x_train, x_val, y_train, y_val = train_test_split(x_model, y_model, test_size=0.2,
random_state=42)

model = GaussianNB()
model.fit(x_train, y_train);
model.fit(x_test, y_test);
print(model.score(x_test,y_test)*100)
```

Decision Tree Classifier

```
x = dm1.iloc[:,0:8]
y = dm1.winner
feature_cols = ['season', 'venue', 'team1',
'team2','toss_winner','toss_decision','win_by_runs','win_by_wickets']
classns = ['10',  '1',  '8', '11',  '3',  '2',  '4',  '9',  '5',  '6', '12', '13']

X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=1)
clf = DecisionTreeClassifier()
clf = clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

Random Forest

```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=1)

from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
xp =[]
```

```
yp =[]
for i in range (1,100):
    clf=RandomForestClassifier(n_estimators=100)
    clf.fit(X_train,y_train)
    y_pred=clf.predict(X_test)
    p = metrics.accuracy_score(y_test, y_pred)
    xp.append(i)
    yp.append(p*100)
plt.plot(xp, yp)
plt.ylim(0,100)
plt.xlim(0,100)
plt.xlabel('Number of estimators')
plt.ylabel('Accuracy')
plt.title('Random Forest Classifier')
plt.show()
```

Logistic Regression

```
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(
        x, y, test_size = 0.2, random_state = 0)

from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(xtrain, ytrain)

y_pred = classifier.predict(xtest)

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(ytest, y_pred)
print ("Confusion Matrix : \n", cm)

from sklearn.metrics import accuracy_score
print ("Accuracy : ", accuracy_score(ytest, y_pred))

from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
logit_roc_auc = roc_auc_score(ytest, classifier.predict(xtest))
fpr, tpr, thresholds = roc_curve(ytest, classifier.predict_proba(xtest)[:,1])
```

```
plt.figure()
plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
plt.show()
```

KNN Classification

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=0)

error_rate = []
# Will take some time
for i in range(1,20,2):
 knn = KNeighborsClassifier(n_neighbors=i)
 knn.fit(x_train,y_train)
 pred_i = knn.predict(x_test)
 error_rate.append(np.mean(pred_i != y_test))

plt.figure(figsize=(10,6))
plt.plot(range(1,20,2),error_rate,color='blue', linestyle='dashed',
marker='o',markerfacecolor='red', markersize=10)
plt.title('Error Rate vs. K Value')
plt.xlabel('K')
plt.ylabel('Error Rate')

from sklearn.neighbors import KNeighborsClassifier
improved_knn = KNeighborsClassifier(n_neighbors = 3)
```

K Means Clustering

```
# To analyse the strike rate of a batsman, we need runs scored by him and the number of balls
he faced.
# The strike rate for the batsman can be calculated from these two values.
```

```
A = []
B = []
C = []
for i in runs["striker"].unique():
 a = runs["runs_off_bat"][runs["striker"] == i].sum() #sum of runs scored by the pla
 balls =  runs["match_id"][runs['striker']== i].count() - runs['noballs'][runs["striker"]==
i].sum() - runs['wides'][runs["striker"]== i].sum()
 out = runs["match_id"][runs['player_dismissed']== i].count()
# The strike rate for the batsman can be calculated from these two values.
 A.append(a)
 B.append(balls)
 C.append(out)

Batsmen = pd.DataFrame({"Player_names": runs["striker"].unique()})
Batsmen["Runs"] = A
Batsmen["Balls_played"] = B
Batsmen["Out"] = C

Batsmen["Strike_Rate"] = (Batsmen["Runs"] * 100 / Batsmen["Balls_played"]).round(2)
Batsmen["Average"] = (Batsmen["Runs"] / Batsmen["Out"]).round(2)

kmeans = KMeans(n_clusters=5)
kmeans.fit(Batsmen[["Strike_Rate", "Runs"]])
Batsmen["cluster"] = kmeans.labels_

fig = plt.figure(figsize=(12,6))

colors = ["blue", "sienna", "limegreen", "red", "indigo"]

for r in range(1,6):
    clustered_Batsmen = Batsmen[Batsmen["cluster"] == r]
    plt.scatter(clustered_Batsmen["Strike_Rate"], clustered_Batsmen["Runs"],
color=colors[r-1])

plt.title("IPL Batsmen", fontsize=16)
plt.xlabel("Strike Rate of the Batsmen", fontsize=14)
plt.ylabel("Runs scored by the Batsmen", fontsize=14)
plt.show()

# To analyse the performance of a bowler, we need runs given by him and the number of
overs he bowled.
# The economy rate of the bowler can be calculated from these two values.
D = []
E = []

for j in runs["bowler"].unique():
    #runs given by him can be calculated by the sum of the runs and extras given by a bowler
    d = runs["runs_off_bat"][runs["bowler"] == j].sum() + runs["wides"][runs["bowler"] ==
j].sum() + runs["noballs"][runs["bowler"] == j].sum()
        #counting the number of balls bowled by a bowler
    e = ((runs["ball"][runs["bowler"] == j].count() - runs["wides"][runs["bowler"] == j].sum()
- runs["noballs"][runs["bowler"] == j].sum()) // 6)

    D.append(d)
    E.append(e)

Bowlers = pd.DataFrame({"Bowler_names": runs["bowler"].unique()})

Bowlers["Runs"] = D
Bowlers["Overs"] = E
Bowlers.head()

# Economy rate can be calculated by dividing the total runs conceded by the number of overs
bowled.
Bowlers["Econ_Rate"] = (Bowlers["Runs"] / Bowlers["Overs"]).round(2)
Bowlers.head()

kmeans = KMeans(n_clusters=5)
kmeans.fit(Bowlers[["Econ_Rate", "Overs"]])
Bowlers["cluster"] = kmeans.labels_

fig = plt.figure(figsize=(12,6))

colors = ["blue", "green", "black", "red"]
```

```
for r in range(1,5):
    clustered_Bowlers = Bowlers[Bowlers["cluster"] == r]
    plt.scatter(clustered_Bowlers["Econ_Rate"], clustered_Bowlers["Overs"],
color=colors[r-1])

plt.title("IPL Bowlers", fontsize=16)
plt.xlabel("Economy Rate of the Bowler", fontsize=14)
plt.ylabel("Number of Overs bowled by the Bowler", fontsize=14)
plt.show()
```