IPL First Innings Score Prediction System

Name - Pavan Boina
Roll no - AM.EN.U4CSE19214

Link to ipynb file (i have also uploaded during submission)

Contents -

Problem Definition

As we know Cricket is the foremost loved game, after football and most loved game in our India. There are many people who try to predict the scores of matches based on their individual cricket sense. Cricket has certain rules and game systems. When a cricket match is played many factors come into consideration such as venue, players performance etc.
In this project we are going to predict the final score of a team in a match by analyzing data of previous matches. We use features such as current score and no of wickets and score in the last few overs. Our system finally presents quantitative results displayed by the best suitable algorithm having highest accuracy. Cricket is a sport with multiple formats, different playing standards and varying duration. T20 and ODI are two limited overs formats among them. We try to predict scores in these two formats in our project.

Datasets

We have found suitable datasets required for our project on the Kaggle website.

The data is about previous records of ball-to-ball information mapped by a batsman to a bowler which includes runs, wickets, overs, runs scored in the last 5 overs, wickets taken in the last 5 overs, strike, non-striker. The datasets are being used by various cricket teams to plan against opponent team players to compare their stats against their bowlers and vice versa.These features will be used to train our machine learning model.

Our Datasets are IPL,ODI and T20 scores datasets

Prepare Data

I have loaded the dataset(IPL)

Data Loading

```
# Importing essential libraries
import pandas as pd

# Loading the dataset
data = pd.read_csv('ipl.csv')
```

```
data.head()
```

	mid	date	venue	bat_team	bowl_team	batsman	bowler	runs	wickets	overs	runs_last_5	wickets_last_5	striker	non-striker	total
0	1	2008-04-18	M Chinnaswamy Stadium	Kolkata Knight Riders	Royal Challengers Bangalore	SC Ganguly	P Kumar	1	0	0.1	1	0	0	0	222
1	1	2008-04-18	M Chinnaswamy Stadium	Kolkata Knight Riders	Royal Challengers Bangalore	BB McCullum	P Kumar	1	0	0.2	1	0	0	0	222
2	1	2008-04-18	M Chinnaswamy Stadium	Kolkata Knight Riders	Royal Challengers Bangalore	BB McCullum	P Kumar	2	0	0.2	2	0	0	0	222
3	1	2008-04-18	M Chinnaswamy Stadium	Kolkata Knight Riders	Royal Challengers Bangalore	BB McCullum	P Kumar	2	0	0.3	2	0	0	0	222
4	1	2008-04-18	M Chinnaswamy Stadium	Kolkata Knight Riders	Royal Challengers Bangalore	BB McCullum	P Kumar	2	0	0.4	2	0	0	0	222

Pre-Processing

My dataset contains variables which does not play crucial role in predicting the score such as mid, batsmen, bowler, because batsmen and bowler will not continue until last moment so we can't rely on those constraints to fit our model, We are removing them from our data

```
# --- Data Cleaning ---
# Removing unwanted columns to make sure the variables which are not needed in generating model are being removed
columns_to_remove = ['mid', 'venue', 'batsman', 'bowler', 'striker', 'non-striker']
data.drop(labels=columns_to_remove, axis=1, inplace=True)
```

My data set contains teams that are not included in current ipl and they have played only less no of matches

```
data['bat_team'].unique()
```

```
array(['Kolkata Knight Riders', 'Chennai Super Kings', 'Rajasthan Royals',
       'Mumbai Indians', 'Deccan Chargers', 'Kings XI Punjab',
       'Royal Challengers Bangalore', 'Delhi Daredevils',
       'Kochi Tuskers Kerala', 'Pune Warriors', 'Sunrisers Hyderabad',
       'Rising Pune Supergiants', 'Gujarat Lions',
       'Rising Pune Supergiant'], dtype=object)
```

So i have created the array with the teams that the feasible and are currently playing in IPL

```
# Keeping only consistent teams
#These are array of teams that are currently playing in IPL
consistent_teams = ['Kolkata Knight Riders', 'Chennai Super Kings', 'Rajasthan Royals',
                    'Mumbai Indians', 'Kings XI Punjab', 'Royal Challengers Bangalore',
                    'Delhi Daredevils', 'Sunrisers Hyderabad']
```

I included the data that has the batting team and bowling team in this array and dropped all other data

```
#Dropping all other teams instead of these consistent teams from both batting and bowling rows
data = data[(data['bat_team'].isin(consistent_teams)) & (data['bowl_team'].isin(consistent_teams))]
```

And we can't predict the score in the first five overs because players will start innings slowly to analyse pitch conditions and sometimes wicket may slow their run rate so I removed the first five overs data.

```
# Removing the first 5 overs data in every match to avoid the false prediction
# We cant predict the score by seeing the score in 5 overs we need
# atleast 5+ overs to estimate considering real world situation
data = data[data['overs']>=5.0]
```

```
data.head()
```

	date	bat_team	bowl_team	runs	wickets	overs	runs_last_5	wickets_last_5	total
32	2008-04-18	Kolkata Knight Riders	Royal Challengers Bangalore	61	0	5.1	59	0	222
33	2008-04-18	Kolkata Knight Riders	Royal Challengers Bangalore	61	1	5.2	59	1	222
34	2008-04-18	Kolkata Knight Riders	Royal Challengers Bangalore	61	1	5.3	59	1	222
35	2008-04-18	Kolkata Knight Riders	Royal Challengers Bangalore	61	1	5.4	59	1	222
36	2008-04-18	Kolkata Knight Riders	Royal Challengers Bangalore	61	1	5.5	58	1	222

I converted the date from string into datetime object as i need to use it for later purpose

```
# Converting the column 'date' from string into datetime object
from datetime import datetime
data['date'] = data['date'].apply(lambda x: datetime.strptime(x, '%Y-%m-%d'))
```

My data has bat team and bowl team as strings and they are the primary key for prediction of score for any match so I used one hot encoding for conversion.

One hot encoding is one method of converting data to prepare it for an algorithm and get a better prediction. With one-hot, we convert each categorical value into a new categorical column and assign a binary value of 1 or 0 to those columns

```
# Converting categorical features using OneHotEncoding method
encoded_data = pd.get_dummies(data=data, columns=['bat_team', 'bowl_team'])
# Here we cannot apply teams directly so instead i hard coded all the teams in dataset
# 1 if it is batting or bowling or else 0
```

```
encoded_data.head()
```

ets_last_5	total	bat_team_Chennai Super Kings	bat_team_Delhi Daredevils	bat_team_Kings XI Punjab	...	bat_team_Royal Challengers Bangalore	bat_team_Sunrisers Hyderabad	bowl_team_Chennai Super Kings	bowl_team_Delhi Daredevils	bowl_team_ XI F
0	222	0	0	0	...	0	0	0	0	
1	222	0	0	0	...	0	0	0	0	
1	222	0	0	0	...	0	0	0	0	
1	222	0	0	0	...	0	0	0	0	
1	222	0	0	0	...	0	0	0	0	

Rearranged the columns for better implementation so the target value will be at last

```
# Rearranging the columns so target value will be at last
encoded_data = encoded_data[['date', 'bat_team_Chennai Super Kings', 'bat_team_Delhi Daredevils', 'bat_team_Kings XI Pun
              'bat_team_Kolkata Knight Riders', 'bat_team_Mumbai Indians', 'bat_team_Rajasthan Royals',
              'bat_team_Royal Challengers Bangalore', 'bat_team_Sunrisers Hyderabad',
              'bowl_team_Chennai Super Kings', 'bowl_team_Delhi Daredevils', 'bowl_team_Kings XI Punjab',
              'bowl_team_Kolkata Knight Riders', 'bowl_team_Mumbai Indians', 'bowl_team_Rajasthan Royals',
              'bowl_team_Royal Challengers Bangalore', 'bowl_team_Sunrisers Hyderabad',
              'overs', 'runs', 'wickets', 'runs_last_5', 'wickets_last_5', 'total']]
```

```
encoded_data.head()
```

l_team_Chennai Super Kings	...	bowl_team_Mumbai Indians	bowl_team_Rajasthan Royals	bowl_team_Royal Challengers Bangalore	bowl_team_Sunrisers Hyderabad	overs	runs	wickets	runs_last_5	wickets_last_5	total
0	...	0	0	1	0	5.1	61	0	59	0	222
0	...	0	0	1	0	5.2	61	1	59	1	222
0	...	0	0	1	0	5.3	61	1	59	1	222
0	...	0	0	1	0	5.4	61	1	59	1	222
0	...	0	0	1	0	5.5	61	1	58	1	222

Summarization

```
encoded_data.shape
```

```
(40108, 23)
```

```
encoded_data.columns
```

```
Index(['date', 'bat_team_Chennai Super Kings', 'bat_team_Delhi Daredevils',
       'bat_team_Kings XI Punjab', 'bat_team_Kolkata Knight Riders',
       'bat_team_Mumbai Indians', 'bat_team_Rajasthan Royals',
       'bat_team_Royal Challengers Bangalore', 'bat_team_Sunrisers Hyderabad',
       'bowl_team_Chennai Super Kings', 'bowl_team_Delhi Daredevils',
       'bowl_team_Kings XI Punjab', 'bowl_team_Kolkata Knight Riders',
       'bowl_team_Mumbai Indians', 'bowl_team_Rajasthan Royals',
       'bowl_team_Royal Challengers Bangalore',
       'bowl_team_Sunrisers Hyderabad', 'overs', 'runs', 'wickets',
       'runs_last_5', 'wickets_last_5', 'total'],
      dtype='object')
```

```
encoded_data.describe()
```

	bat_team_Chennai Super Kings	bat_team_Delhi Daredevils	bat_team_Kings XI Punjab	bat_team_Kolkata Knight Riders	bat_team_Mumbai Indians	bat_team_Rajasthan Royals	bat_team_Royal Challengers Bangalore	bat_team_Sunriser Hyderaba
count	40108.000000	40108.000000	40108.000000	40108.000000	40108.000000	40108.000000	40108.000000	40108.00000
mean	0.140570	0.111798	0.146654	0.116062	0.157076	0.112471	0.127231	0.08813
std	0.347582	0.315122	0.353765	0.320303	0.363877	0.315949	0.333236	0.28349
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
50%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
75%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.00000

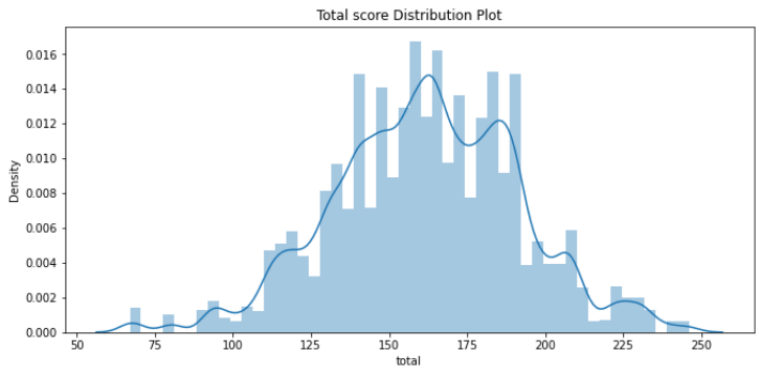
8 rows × 22 columns

Visualization

A distribution plot to know where the total runs density is

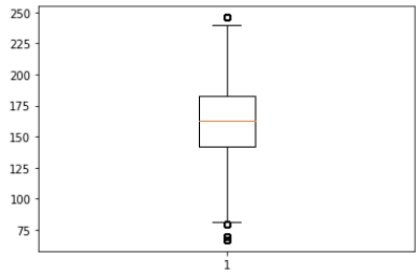
```
In [145]: from matplotlib import pyplot as plt
          import seaborn as sns
          import warnings
          warnings.simplefilter(action='ignore', category=FutureWarning)
          # This makes it very easy to spot anomalies, such as outliers
          plt.figure(figsize=[11,5])
          sns.distplot(encoded_data['total'])
          plt.title('Total score Distribution Plot')
```

```
Out[145]: Text(0.5, 1.0, 'Total score Distribution Plot')
```



```
In [116]: plt.boxplot(encoded_data.total)
```

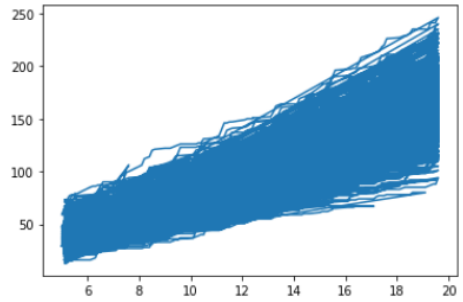
```
Out[116]: {'whiskers': [<matplotlib.lines.Line2D at 0x1afcc7f7190>,
                        <matplotlib.lines.Line2D at 0x1afcc7f7520>],
           'caps': [<matplotlib.lines.Line2D at 0x1afcc7f78b0>,
                    <matplotlib.lines.Line2D at 0x1afcc7f7c40>],
           'boxes': [<matplotlib.lines.Line2D at 0x1afcc7e9dc0>],
           'medians': [<matplotlib.lines.Line2D at 0x1afcc7f7fd0>],
           'fliers': [<matplotlib.lines.Line2D at 0x1afcc8053a0>],
           'means': []}
```



Mostly total score is in range 150 to 180

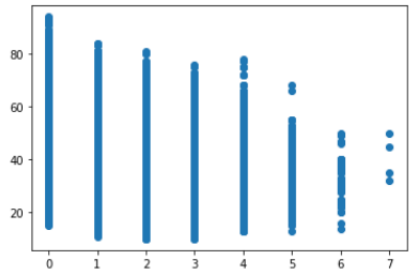
A graph to show how the runs vary with overs

```
plt.plot(encoded_data.overs,encoded_data.runs)
plt.show()
```



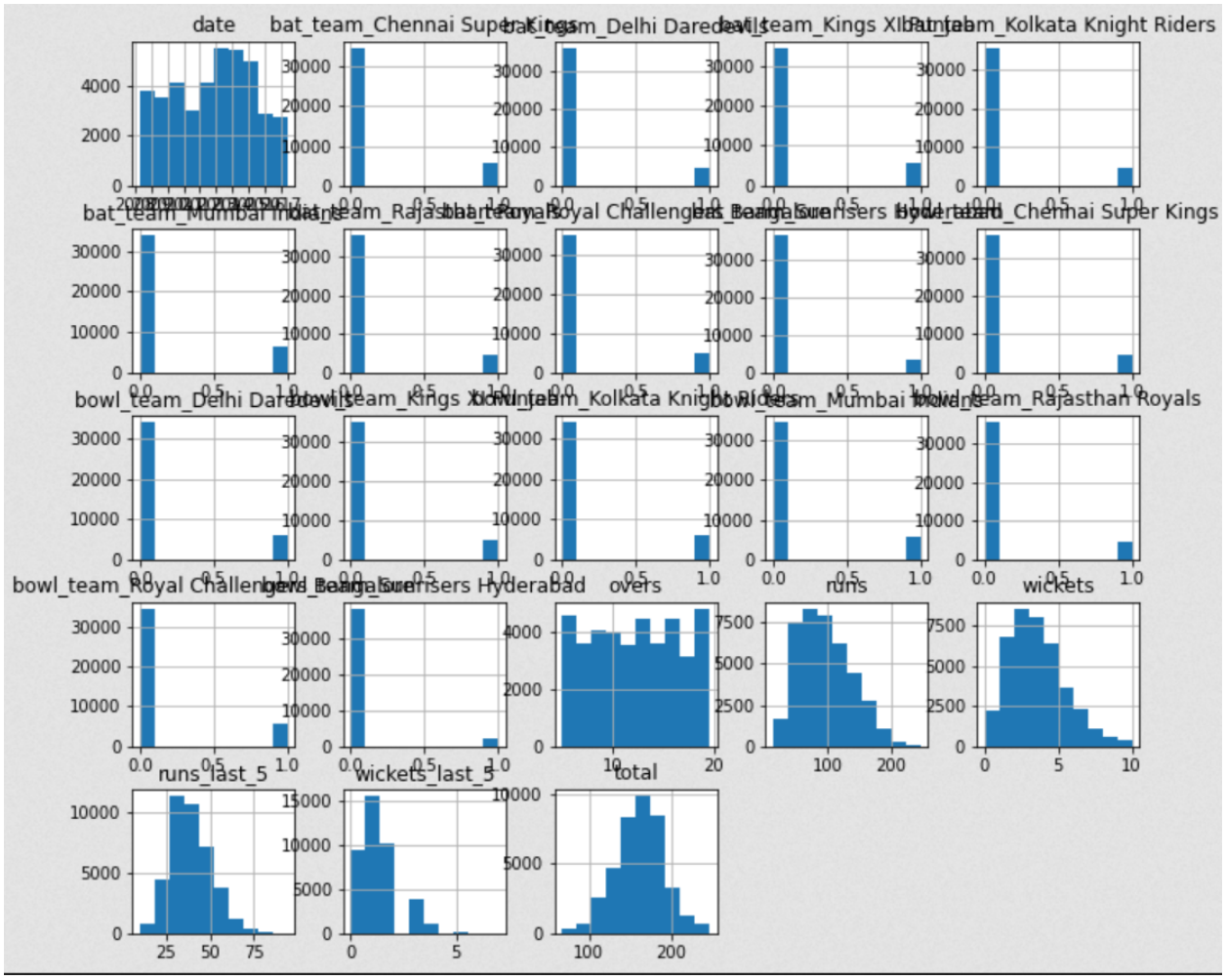
A scatter plot to show the relation between wickets and runs in last five overs

```
In [115]: plt.scatter(encoded_data.wickets_last_5,encoded_data.runs_last_5)
          plt.show()
```



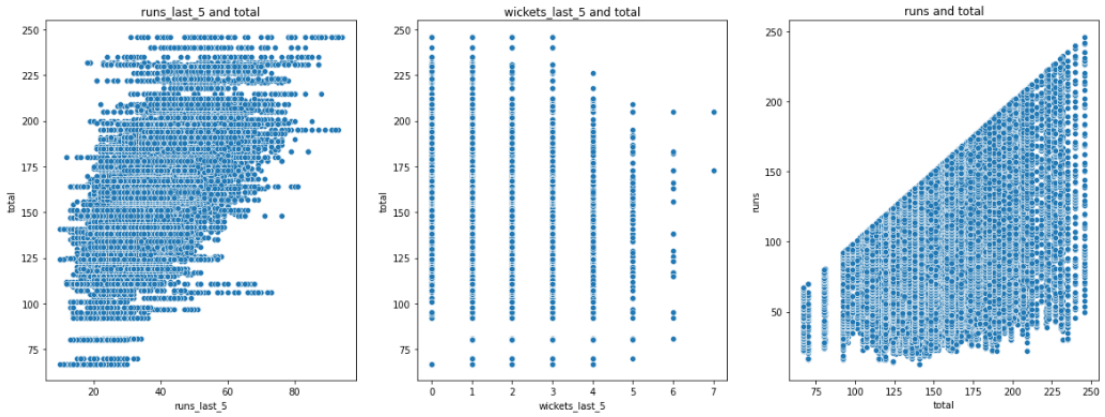
A histogram of the data

```
encoded_data.hist(figsize=[11,10])
plt.show()
```



Different scatterplots to visualize the relation between data

```
In [118]: import seaborn as sns
          plt.figure(figsize=[20,7])
          plt.subplot(1,3,1)
          plt.title("runs_last_5 and total")
          sns.scatterplot(x='runs_last_5',y='total',data=encoded_data)
          plt.subplot(1,3,2)
          plt.title("wickets_last_5 and total")
          sns.scatterplot(x='wickets_last_5',y='total',data=encoded_data)
          plt.subplot(1,3,3)
          plt.title("runs and total")
          sns.scatterplot(y='runs',x='total',data=encoded_data)

Out[118]: <AxesSubplot:title={'center':'runs and total'}, xlabel='total', ylabel='runs'>
```



Python packages

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from datetime import datetime
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import KFold
```

PANDAS

Pandas is one of the tools in Machine Learning which is used for data cleaning and analysis. Used pandas lo load data and do one hot encoding

NUMPY

Numpy stands for Numerical python.Numpy is an open source python library used to perform various mathematical and scientific tasks. Used numpy to convert array to numpy arrays

MATPLOTLIB

Data visualization is your main way to communicate with non-Data Wizards. Matplotlib is the basis of image visualization in Python. From visualizing your edge detection algorithm to looking at distributions in your data.Used this to do various visualisations on our dataset

KFold

K-fold cross validation method allows the use of different training and testing data which will avoid the overfitting and give better generalization ability. Performed k fold validation on my data set.

Train_test_split

train_test_split is a function in Sklearn model selection for splitting data arrays into two subsets. Performed k fold validation by using thai package to split our data.

KNeighborsClassifier,LinearRegression,RandomForestRegressor

Evaluated my model on these three different algorithms to know the accuracy of my model. Used these packages to perform evaluation.

Machine Learning Algorithms

Instead of separating our dataset based on percentage I separated my database based on year so I can train my model well.

```
#In datasets related to time and date we have to split train and test sample based on date.
# I have seperated dataset based on year 2017 i.e => after 2016 comes under testing and remaining under tra
X_train = encoded_data.drop(labels='total', axis=1)[encoded_data['date'].dt.year <= 2016]
X_test = encoded_data.drop(labels='total', axis=1)[encoded_data['date'].dt.year >= 2017]
```

```
y_train = encoded_data[encoded_data['date'].dt.year <= 2016]['total'].values
y_test = encoded_data[encoded_data['date'].dt.year >= 2017]['total'].values
```

```
#Removing date column as we dont need that to train model
X_train.drop(labels='date', axis=True, inplace=True)
X_test.drop(labels='date', axis=True, inplace=True)
```

Date is used to separate the dataset, it will not play any role in my model so I removed the date inplace.

Accuray

```
#To find accuracy of our model
def custom_accuracy(y_test,y_pred,thresold):
    ans = 0
    n = len(y_pred)
    for i in range(n):
        if(abs(y_pred[i]-y_test[i]) <= thresold):
            ans += 1
    return ((ans/n)*100)
```

As our prediction is a number and it may not come as accurate as real as we know these games are unpredictable exactly so i took threshold to measure the accuracy of the model.

KNN Implementation

- K-NN algorithm stores all the data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suited category by using K- NN algorithm.
- It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.So it is very slow at running time.

```
knn = KNeighborsClassifier(n_neighbors = 5)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)

k_acc = custom_accuracy(y_test,y_pred,20)
print("Custom accuracy for knn:" , k_acc)
```

```
Custom accuracy for knn: 64.00287976961843
```

KNN Implementation from scratch

```
def euclidDistance(a,b):
    return np.linalg.norm(a-b)
```

```
def nearestNeighbors(trainingData,testSample,k):
    distances = []
    for x in trainingData:
        distances.append([x,euclidDistance(x[0:testSample.shape[0]],testSample)])
    distances.sort(key = lambda x : x[-1])
    distances = np.array(distances, dtype = 'object')
    return distances[0:k]
```

```
def predict(trainingData,testData,k):
    res = []
    for x in testData:
        neighbors = nearestNeighbors(trainingData,x,k)
        values = []
        for y in neighbors:
            values.append(y[0][-1])
        res.append(max(set(values),key = values.count))
    return res
```

```
trainingData = np.column_stack((np.array(X_train.values),np.array(y_train)))
testingData = np.array(X_test.values[0:30])
pred = predict(trainingData,testingData,5)
print("Custom accuracy for KNN(scratch):" , custom_accuracy(y_test,pred,20))
```

```
Custom accuracy for KNN(scratch): 60.0
```

- euclidDistance function is used to find the distance between two points.
- nearestNeighbors is used to calculate and store distances for every point in testing data later sort the distances and return the first k neighbours.
- predict function iterates test dataset and predict the score for every point and returns the prediction at last.

K-fold cross validation(KNN)

```
kf = KFold(n_splits=10, random_state=None)
accuracy = []
avg_a = 0
max_a = 0
min_a = 100

for train_index, test_index in kf.split(X):
    x1, x2 = X[train_index], X[test_index]
    y1, y2 = y[train_index], y[test_index]

    knn = KNeighborsClassifier(n_neighbors = 5)
    knn.fit(x1, y1)
    y3 = knn.predict(x2)

    accu = custom_accuracy(y2,y3,20)
    accuracy += [accu]
    max_a = max(max_a,accu)
    min_a = min(min_a,accu)
    avg_a += accu

print("Accuracies -",accuracy)
print("Maximum Accuracy -",max_a)
print("Minimum Accuracy -",min_a)
print("Average Accuracy -",avg_a/10)
```

```
Accuracies - [62.05435053602593, 67.71378708551484, 64.94639740713039, 62.777362253802046, 66.66666666666666, 64.6472
2014460234, 64.54749439042632, 64.57242582897034, 61.770573566084785, 64.3142144638404]
Maximum Accuracy - 67.71378708551484
Minimum Accuracy - 61.770573566084785
Average Accuracy - 64.4010492343064
```

Linear regression

- Linear regression attempts to model the relationship between variables by fitting a linear equation to observed data. All variables except total are considered to be an explanatory variable, and the total score is considered to be a dependent variable.
- Linear regression performs the task to predict a dependent variable value based on a given independent variables . So, this regression technique finds out a linear relationship between input and output.

```
lr = LinearRegression()
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)
score = lr.score(X_test,y_test)*100
l_acc = custom_accuracy(y_test,y_pred,20)

print("R square value:" , score)
print("Custom accuracy for linear regression:" , l_acc)
```

```
R square value: 75.19522885796084
Custom accuracy for linear regression: 80.99352051835854
```

K-fold cross validation(Linear regression)

```
In [132]: kf = KFold(n_splits=10, random_state=None)
          accuracy = []
          avg_a = 0
          max_a = 0
          min_a = 100

          for train_index, test_index in kf.split(X):
              x1, x2 = X[train_index], X[test_index]
              y1, y2 = y[train_index], y[test_index]

              lr = LinearRegression()
              lr.fit(x1, y1)
              y3 = lr.predict(x2)

              accu = custom_accuracy(y2,y3,20)
              accuracy += [accu]
              max_a = max(max_a,accu)
              min_a = min(min_a,accu)
              avg_a += accu

          print("Accuracies -",accuracy)
          print("Maximum Accuracy -",max_a)
          print("Minimum Accuracy -",min_a)
          print("Average Accuracy -",avg_a/10)

          Accuracies - [77.03814510097232, 81.12690102218897, 78.98279730740464, 74.56993268511593, 82.4731987035652, 77.262528
          04786836, 75.24308152580403, 76.96335078534031, 77.03241895261846, 81.74563591022444]
          Maximum Accuracy - 82.4731987035652
          Minimum Accuracy - 74.56993268511593
```

RandomForest Regressor

- The random forest algorithm establishes the outcome based on the predictions of the decision trees. It predicts by taking the average or mean of the output from various trees. Increasing the number of trees increases the precision of the outcome.

```
In [133]: rf = RandomForestRegressor(n_estimators=100,max_features=None)
          rf.fit(X_train, y_train)
          y_pred = rf.predict(X_test)
          score = rf.score(X_test,y_test)*100
          r_acc = custom_accuracy(y_test,y_pred,20)

          print("R square value:" , score)
          print("Custom accuracy for linear regression:" , r_acc)

          R square value: 66.94433446155729
          Custom accuracy for linear regression: 74.15406767458603
```

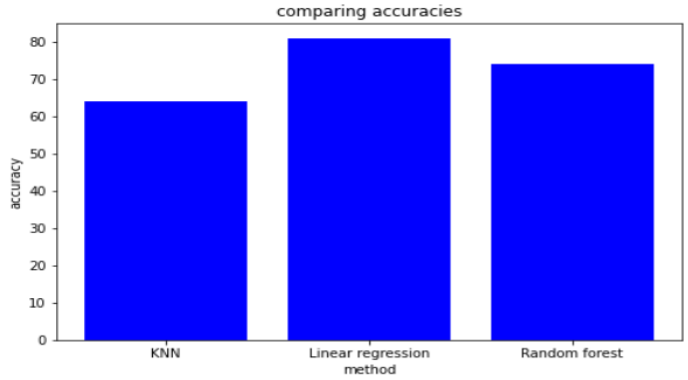
K-fold cross validation(Random forest regression)

```
In [134]: kf = KFold(n_splits=10, random_state=None)
          accuracy = []
          avg_a = 0
          max_a = 0
          min_a = 100

          for train_index, test_index in kf.split(X):
              x1, x2 = X[train_index], X[test_index]
              y1, y2 = y[train_index], y[test_index]

              rf = RandomForestRegressor(n_estimators=100,max_features=None)
              rf.fit(x1, y1)
              y3 = rf.predict(x2)

              accu = custom_accuracy(y2,y3,20)
              accuracy += [accu]
              max_a = max(max_a,accu)
              min_a = min(min_a,accu)
              avg_a += accu

          print("Accuracies -",accuracy)
          print("Maximum Accuracy -",max_a)
          print("Minimum Accuracy -",min_a)
          print("Average Accuracy -",avg_a/10)

          Accuracies - [69.80802792321117, 77.76115681874845, 76.86362503116429, 69.75816504612317, 78.20992271254052, 72.22637
          746197955, 72.97432061829967, 74.96883570182, 77.23192019950125, 75.6857855361596]
          Maximum Accuracy - 78.20992271254052
          Minimum Accuracy - 69.75816504612317
          Average Accuracy - 74.54881370495477
```

Accuracy plot

Accuracy plot

```
|: colnames = ["accuracy","names"]
   acc = [[k_acc,"KNN"],[l_acc,"Linear regression"],[r_acc,"Random forest"]]

   mydataframe = pd.DataFrame(acc,columns=colnames)
   plt.figure(figsize=(8,5))
   plt.bar("names", "accuracy", data = mydataframe, color = "blue")
   plt.xlabel("method")
   plt.ylabel("accuracy")
   plt.title("comparing accuracies")
   plt.show()
```



Out of all the three algorithms linear regression gives the best outcome for my model with an accuracy of 81%.

Model evaluation

Lets Predict the score

```
a = np.array([0,1,2,3,4,5,6,7])
b = np.zeros((a.size, a.max()+1))
b[np.arange(a.size),a] = 1
```

```
for i in range(len(consistent_teams)):
    print(i,consistent_teams[i])
bat_team = int(input("Please input the bat team from the list(0-7): "))
print()
print("You chose: ",consistent_teams[bat_team])
bat_team = b[bat_team]
```

```
0 Kolkata Knight Riders
1 Chennai Super Kings
2 Rajasthan Royals
3 Mumbai Indians
4 Kings XI Punjab
5 Royal Challengers Bangalore
6 Delhi Daredevils
7 Sunrisers Hyderabad
Please input the bat team from the list(0-7): 5

You chose:  Royal Challengers Bangalore
```

```
for i in range(len(consistent_teams)):
    print(i,consistent_teams[i])
bowl_team = int(input("Please input the bowl team from the list(0-7): "))
print()
print("You chose: ",consistent_teams[bowl_team])
bowl_team = b[bowl_team]
```

```
0 Kolkata Knight Riders
1 Chennai Super Kings
2 Rajasthan Royals
3 Mumbai Indians
4 Kings XI Punjab
5 Royal Challengers Bangalore
6 Delhi Daredevils
7 Sunrisers Hyderabad
Please input the bowl team from the list(0-7): 6

You chose:  Delhi Daredevils
```

```
overs = float(input("Please enter the current over(>5.0): "))

runs = int(input("Please enter the current score: "))

wickets = int(input("Please enter the wickets fallen: "))

runs_last_5 = int(input("Please enter the runs scored in last 5 overs: "))

wickets_last_5 = int(input("Please enter the wickets fallen in last 5 overs: "))
```

```
Please enter the current over(>5.0): 12
Please enter the current score: 144
Please enter the wickets fallen: 4
Please enter the runs scored in last 5 overs: 53
Please enter the wickets fallen in last 5 overs: 1
```

```
#making a vector out of the values
data=[]
for i in bat_team:
    data.append(i)
for i in bowl_team:
    data.append(i)
data.append(overs)
data.append(runs)
data.append(wickets)
data.append(runs_last_5)
data.append(wickets_last_5)
data=np.array([data])
```

```
print("Predicted Score from all three regressor's are: ")
pred=int(lr.predict(data))
print(pred)
print(int(knn.predict(data)))
print(int(rf.predict(data)))
```

```
Predicted Score from all three regressor's are:
202
211
216
```

```
print("So the average score is from {} to {}".format(pred-5,pred+5))
```

```
So the average score is from 197 to 207
```

We got the best result for linear regression so our predicted value should be the result from linear regression which will be around 202 in the above case.