EAST WEST UNIVERSITY

Report on Data Science Assignment of CSE 464

Group Members:

1: Alinoor Hossain Olip

ID#2014-3-60-008

2: Hasan Shakif Tushar

ID#2014-1-60-094

3: Mamunur Rashid

ID# 2014-2-60-023

4: Aiswarja Saha Joye

ID# 2014-2-60-114

i) Reading CSV file:

The first thing to work with data is to read the dataset. In Jupyter Notebook we first imported pandas package and then read the csv file which contains individual player's test match data.

```
In [1]: import pandas as pd
tf = pd.read_csv('F:\dataset1_test.csv')
tf1=tf.dropna()
%matplotlib inline
import pandas as pd

df = pd.DataFrame(tf1) #  Two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes

df.columns = ["Batsman","Runs","Mins","BF","four","six","SR","Pos","Dismissal","Inns"]

print(df)
batsman1= df[df.Batsman==1]
batsman2=df[df.Batsman==2]
batsman3=df[df.Batsman==3]
batsman4=df[df.Batsman==4]
batsman5=df[df.Batsman==5]
batsman6=df[df.Batsman==6]

print(batsman1)
print(batsman2)
print(batsman3)
print(batsman4)
print(batsman5)
print(batsman6)
```

```
     Batsman  Runs   Mins   BF  four  six      SR  Pos Dismissal  Inns
0          1    33   78.0   56     5    0   58.92    3    caught     2
1          1    33   77.0   49     6    0   67.34    3    caught     4
2          1   215  503.0  346    25    1   62.13    3       lbw     1
3          1    58   67.0   48     9    0  120.83    3    bowled     3
4          1     7   26.0   18     1    0   38.88    3    caught     1
5          1     8   40.0   27     1    0   29.62    3    caught     3
6          1     6    2.0    3     1    0  200.00    3    caught     1
7          1     5   20.0    9     1    0   55.55    3    caught     3
8          1   143  387.0  252    17    2   56.74    3    bowled     1
9          1    48   98.0   78     8    0   61.53    4    bowled     1
10         1     1    8.0    2     0    0   50.00    4    caught     3
11         1    27   91.0   68     3    0   39.70    4    caught     1
12         1   138  272.0  185    18    0   74.59    3    caught     3
```

292	6	52	NaN	53	6	1	98.11	1	bowled	1
293	6	41	NaN	62	6	0	66.12	1	caught	3
294	6	4	NaN	3	1	0	133.33	1	caught	2
295	6	2	NaN	7	0	0	28.57	1	lbw	4

We cleaned our dataset using dropna() function then then we renamed the columns and split the dataset for individual batsman.

ii) Comparisons of Performance:

a) Fourth Innings average run and ball faced

In [2]:
```
#4th innings stat of individual batsman
batsman1_4= df[(df.Batsman==1)&(df.Inns==4)]
batsman2_4=df[(df.Batsman==2)&(df.Inns==4)]
batsman3_4=df[(df.Batsman==3)&(df.Inns==4)]
batsman4_4=df[(df.Batsman==4)&(df.Inns==4)]
batsman5_4=df[(df.Batsman==5)&(df.Inns==4)]
batsman6_4=df[(df.Batsman==6)&(df.Inns==4)]
print(batsman1_4)
```

	Batsman	Runs	Mins	BF	four	six	SR	Pos	Dismissal	Inns
1	1	33	77.0	49	6	0	67.34	3	caught	4
14	1	14	35.0	24	2	0	58.33	3	lbw	4
20	1	53	78.0	46	4	1	115.21	4	not out	4
22	1	55	151.0	125	1	0	44.00	4	lbw	4
24	1	30	57.0	58	3	0	51.72	5	caught	4
26	1	8	17.0	14	1	0	57.14	3	bowled	4
28	1	34	118.0	91	5	0	37.36	4	caught	4
32	1	40	84.0	52	5	0	76.92	4	caught	4
41	1	28	69.0	48	3	0	58.33	3	lbw	4
47	1	37	149.0	99	3	0	37.37	4	caught	4
49	1	16	24.0	9	3	0	177.77	3	caught	4

In [3]:
```
# total innings played by individual batsman in 4th innings
inns1=batsman1_4.shape[0]
inns2=batsman2_4.shape[0]
inns3=batsman3_4.shape[0]
inns4=batsman4_4.shape[0]
inns5=batsman5_4.shape[0]
inns6=batsman6_4.shape[0]
#notout in 4th innings
notout1=batsman1_4[(batsman1_4.Dismissal=="not out")]
notout2=batsman2_4[(batsman2_4.Dismissal=="not out")]
notout3=batsman3_4[(batsman3_4.Dismissal=="not out")]
notout4=batsman4_4[(batsman4_4.Dismissal=="not out")]
notout5=batsman5_4[(batsman5_4.Dismissal=="not out")]
notout6=batsman6_4[(batsman6_4.Dismissal=="not out")]
finns1=inns1-notout1.shape[0]
finns2=inns2-notout2.shape[0]
finns3=inns3-notout3.shape[0]
finns4=inns4-notout4.shape[0]
finns5=inns5-notout5.shape[0]
finns6=inns6-notout6.shape[0]
runs1=batsman1_4.Runs.sum()
runs2=batsman2_4.Runs.sum()
runs3=batsman3_4.Runs.sum()
runs4=batsman4_4.Runs.sum()
runs5=batsman5_4.Runs.sum()
runs6=batsman6_4.Runs.sum()
#average runs in 4th innings
avg1=runs1/finns1
avg2=runs2/finns2
avg3=runs3/finns3
avg4=runs4/finns4
avg5=runs5/finns5
avg6=runs6/finns6
avg_list=[avg1,avg2,avg3,avg4,avg5,avg6]
avg_list
```

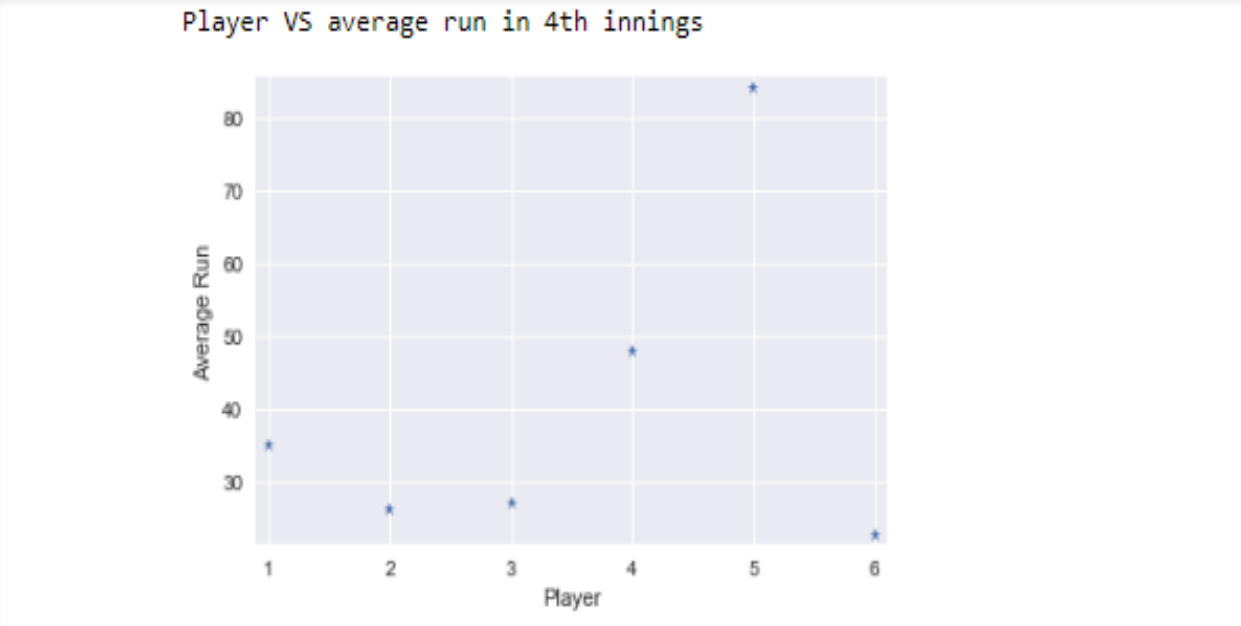
For finding average runs we have to divide total runs by innings but the notout innings must be excluded. inns1=batsman1_4.shape[0] gives the total rows for batsman 1 which is apparently the innings number. notout1=batsman1_4[(batsman1_4.Dismissal=="not out")] gives the total number of times the batsman1 was not out. For finding total innings that the batsmen1 were out we wrote this code finns1=inns1-notout1.shape[0]. We stored batsman1 total runs in runs1 (runs1=batsman1_4.Runs.sum()). Then we have calculated the average and pushed them into a list for plotting in a graph.

```
avg6=runs6/finns6
avg_list=[avg1,avg2,avg3,avg4,avg5,avg6]
avg_list
```

Out[3]: [34.8, 26.0, 27.0, 47.72727272727273, 84.2, 22.6]

Then we plotted the graph. In x axis we took the batsman number and y axis we took the average list.

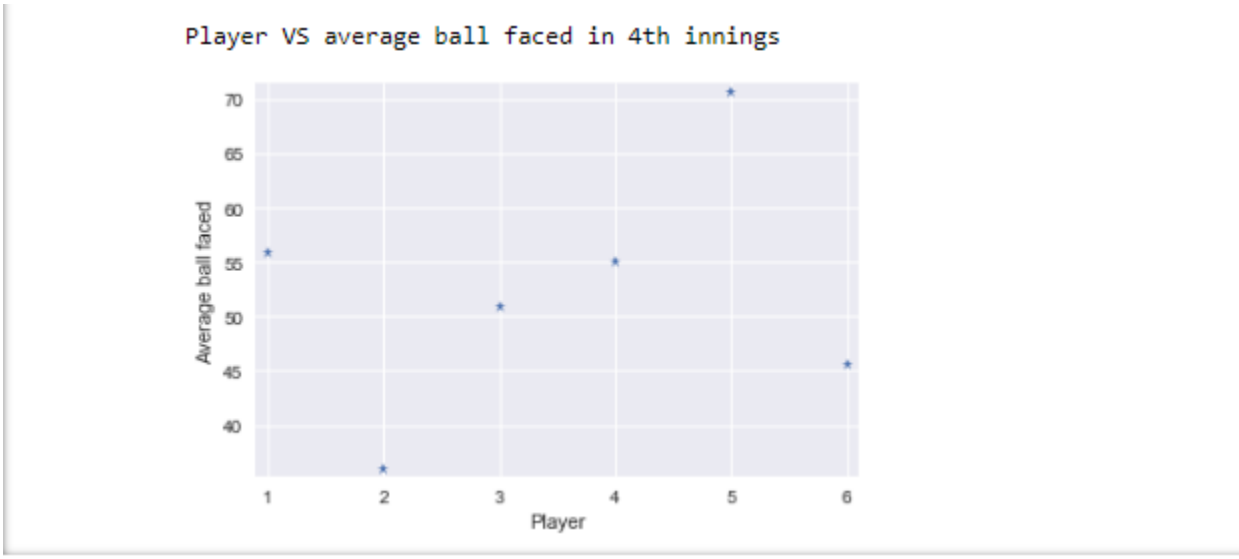
In [4]:
```
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
player = [1, 2, 3, 4, 5, 6]
print("Player VS average run in 4th innings")
_ = plt.plot(player, avg_list, marker = '*', linestyle = 'none')
_ = plt.margins(0.02)
_ = plt.xlabel ('Player')
_ = plt.ylabel ('Average Run')
plt.show()
```



This graph shows that player 5 has the highest average among the batsman. That means for draw or win, batsman5 has the highest impact then player4, player1, player3, player2, player6 respectively.

We also calculated the average ball faced in 4th innings for individual player.

```
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()

print("Player VS average ball faced in 4th innings")
_ = plt.plot(player,avg_bf_list, marker = '*', linestyle = 'none')
_ = plt.margins(0.02)
_ = plt.xlabel ('Player')
_ = plt.ylabel ('Average ball faced')
plt.show()
```



The result is almost same as above graph but player6 has surprisingly high value than player2 in terms of ball facing. The average of player2 is slightly better in 4th innings than player6. But player6 has survived in the pitch better playing over twice as much ball than player2.

So combining both player6 has the higher impact than player2 in 4th innings.

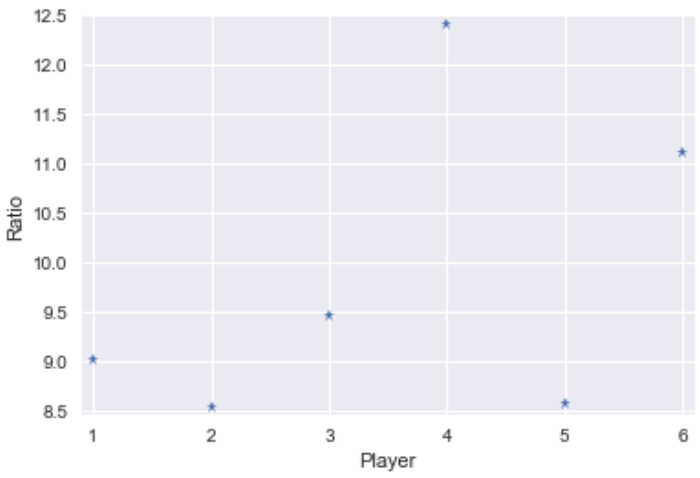
b)Strike rate and boundaries ratio

From the ratio we can assume about how a batsman score. The higher ratio means that the batsman is consistent in an innings to take runs with single, doubles and triples.

In [6]:
```
#Summation of strike rate for individual batsman
SR1=batsman1.SR.sum()
SR2=batsman2.SR.sum()
SR3=batsman3.SR.sum()
SR4=batsman4.SR.sum()
SR5=batsman5.SR.sum()
SR6=batsman6.SR.sum()

#total boundaries of individual batsman
bn1=(batsman1.four.sum()) + (batsman1.six.sum())
bn2=(batsman2.four.sum()) + (batsman2.six.sum())
bn3=(batsman3.four.sum()) + (batsman3.six.sum())
bn4=(batsman4.four.sum()) + (batsman4.six.sum())
bn5=(batsman5.four.sum()) + (batsman5.six.sum())
bn6=(batsman6.four.sum()) + (batsman6.six.sum())

#Ratio of Strike rate and Boundaries
rat_sn1=SR1/bn1
rat_sn2=SR2/bn2
rat_sn3=SR3/bn3
rat_sn4=SR4/bn4
rat_sn5=SR5/bn5
rat_sn6=SR6/bn6

ratio_list=[rat_sn1,rat_sn2,rat_sn3,rat_sn4,rat_sn5,rat_sn6]
ratio_list
```

In [7]:
```
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
player = [1, 2, 3, 4, 5, 6]
print("Player VS Ratio of Strike Rate and Boundaries")
_ = plt.plot(player, ratio_list, marker = '*', linestyle = 'none')
_ = plt.margins(0.02)
_ = plt.xlabel ('Player')
_ = plt.ylabel ('Ratio')
plt.show()
```

```
Player VS Ratio of Strike Rate and Boundaries
```



This graph shows that player4 has the highest ratio so this player is always busy to take runs in spite of scoring fewer boundaries.

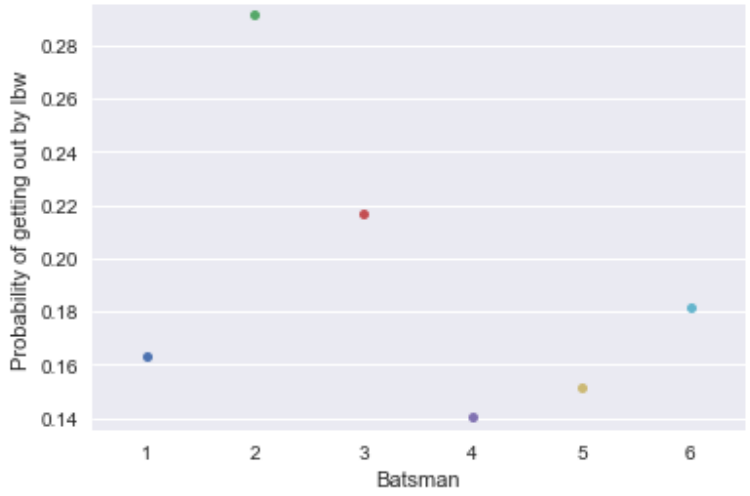
c) Probability of getting out by lbw

The lower probability indicates the better footwork.

In [24]:
```
lbw1=batsman1[(batsman1.Dismissal=="lbw")]

lbw2=batsman2[(batsman2.Dismissal=="lbw")]
lbw3=batsman3[(batsman3.Dismissal=="lbw")]
lbw4=batsman4[(batsman4.Dismissal=="lbw")]
lbw5=batsman5[(batsman5.Dismissal=="lbw")]
lbw6=batsman6[(batsman6.Dismissal=="lbw")]



rlbw1=lbw1.shape[0]/finns11
rlbw2=lbw2.shape[0]/finns22
rlbw3=lbw3.shape[0]/finns33
rlbw4=lbw4.shape[0]/finns44
rlbw5=lbw5.shape[0]/finns55
rlbw6=lbw6.shape[0]/finns66

lbw_list=[rlbw1,rlbw2,rlbw3,rlbw4,rlbw5,rlbw6]
lbw_list
```

Out[24]:
```
[0.16326530612244897,
 0.2916666666666667,
 0.21666666666666667,
 0.14035087719298245,
 0.15151515151515152,
 0.18181818181818182]
```

In [26]:
```
import seaborn as sns
sns.set()
Batsman=[1,2,3,4,5,6]
_ = sns.swarmplot(x= Batsman, y = lbw_list, data = df)
_ = plt.margins(0.02)
_ = plt.xlabel ('Batsman')
_ = plt.ylabel ('Probability of getting out by lbw')
plt.show()
```



From the graph we can see batsman4 is the best at footwork among the batsman and then batsman5, batsman1, batsman6, batsman3 and batsman2 respectively.

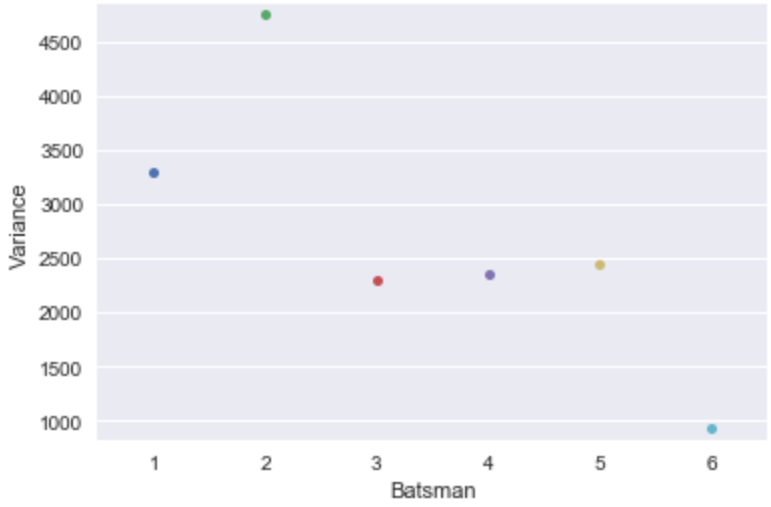
d)Consistency by variance of runs

We built a function to calculate variance as built in function (var()) will count not out innings.

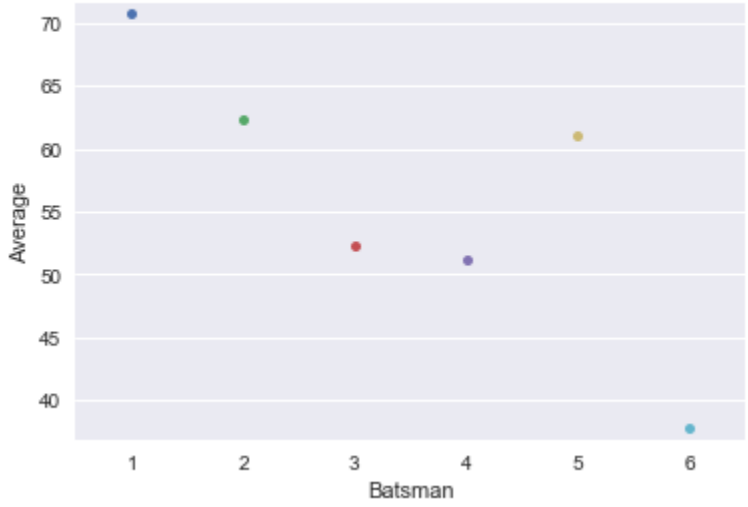
In [9]:
```
#listing only values of Runs Column
new1=batsman1.Runs
new2=batsman2.Runs
new3=batsman3.Runs
new4=batsman4.Runs
new5=batsman5.Runs
new6=batsman6.Runs

#user defined function for calculating variance respect to average runs

def runs_variance(new1, avg11):
    variance = 0
    for i in new1:
        variance += (avg11 - i) ** 2
    return ( variance / len(new1))

def runs_variance(new2, avg22):
    variance = 0
    for i in new2:
        variance += (avg22 - i) ** 2
    return ( variance / len(new2))
```

In [15]:
```
import seaborn as sns
sns.set()
player=[1,2,3,4,5,6]
print("Player vs Variance ")
_ = sns.swarmplot(x= player, y = var_list, data = df,dodge=True)
_ = plt.margins(0.02)
_ = plt.xlabel ('Batsman')
_ = plt.ylabel ('Variance')
plt.show()


import seaborn as sns
sns.set()
player=[1,2,3,4,5,6]
print("Player vs Average ")
_ = sns.swarmplot(x= player, y = tot_avg_list, data = df)
_ = plt.margins(0.02)
_ = plt.xlabel ('Batsman')
_ = plt.ylabel ('Average')
plt.show()
```

Player vs Variance



Player vs Average

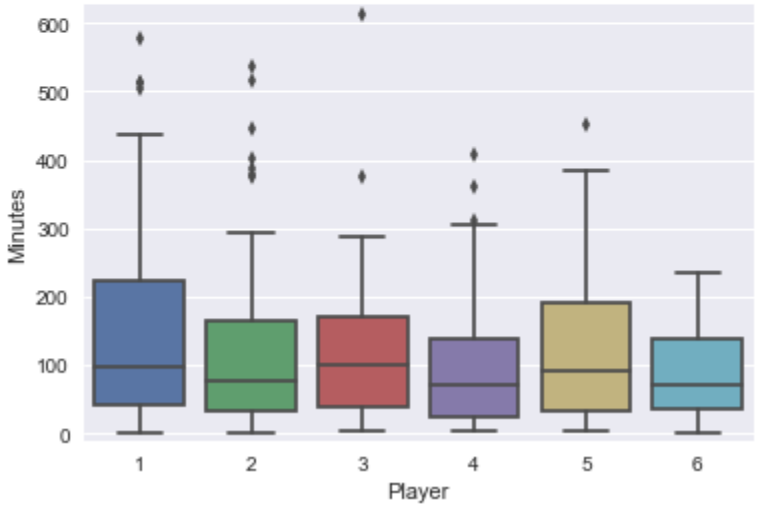


From the graph of player vs average and player vs variance we can see batsman1 is best among them as it got the highest average as well as comparatively lower variance in respect to average. On the other hand batsman2 has the slightly higher average than batsman5 but the variance of batsman2 is much higher than batsman5 which makes batsman5 more consistent than batsman2.

e)Boxplot measure of minutes played

```
In [18]: import seaborn as sns
sns.set()
print("Player vs Minutes Played")
_ = sns.boxplot(x = 'Batsman', y = 'Mins', data = df)
_ = plt.margins(0.02)
_ = plt.xlabel ('Player')
_ = plt.ylabel ('Minutes')
plt.show()
```

```
Player vs Minutes Played
```



From the above plot we see all the batsman boxplot are left skewed 50 percent of data for batsman 1 and 2 is greater than this value 100 and rest of the batsman are below 100. Upper Quartile of batsman 1 is 2005 and rest of the below 200 minutes and lower Quartile for batsman 1, 3 are less than 25 percent data 50 minutes and rest of the batsman are below of this 50 percent. The minimum value of the all batsman is 0 minutes excluding outliers. Maximum value for batsman_1 around 420, batsman_12,3,4 are around 300 ,batsman_4 around 200 and batsman_5 around 130 excluding outliers.

So, we can summarize that batsman_1 is the best performer for test match and also can make more runs in that particular test match.

iii) Calculating Career Average:

We have calculated career average in normal and t distribution with 85% confidence interval. We have calculated the z score and t score from scipy.stats package with built in function.

```
In [ ]: #calculating z score
        import scipy.stats as st
        z=st.norm.ppf(.85)

        #finding t values
        from scipy import stats

        t1= (stats.t.ppf(0.85, (finns11)-1))
        t2= (stats.t.ppf(0.85, (finns22)-1))
        t3= (stats.t.ppf(0.85, (finns33)-1))
```

Then we defined function for calculating career average for t and z distribution and calculated the career average intervals.

```
In [ ]: #functions for calculating confidence interval normal distribution

        def z_plus(meanx,stdy,nz):
            return (meanx+(z*(stdy/nz)))

        def z_minus(meanx,stdy,nz):
            return (meanx-(z*(stdy/nz)))

        career1p=z_plus(mean1,std1,n1)
        career1m=z_minus(mean1,std1,n1)


        #confidence interval using t distribution
        def t_plus(meanx,stdy,nz,tx):
            return (meanx+(tx*(stdy/nz)))

        def t_minus(meanx,stdy,nz,tx):
            return (meanx-(tx*(stdy/nz)))


        careertp1=t_plus(mean1,std1,n1,t1)
        careertm1=t_minus(mean1,std1,n1,t1)
```

We found the following values

```
USING NORMAL DISTRIBUTION

Career Average of Player 1 is between 79.30579012406116 to 62.32686293716333
according to normal distribution with 85% confidence interval

Career Average of Player 2 is between 72.67641688946777 to 52.031916443865555
according to normal distribution with 85% confidence interval

Career Average of Player 3 is between 58.75544056141477 to 45.94455943858523
according to normal distribution with 85% confidence interval

Career Average of Player 4 is between 57.794704341443214 to 44.485997412942744
according to normal distribution with 85% confidence interval

Career Average of Player 5 is between 69.99092885956728 to 52.13028326164485
according to normal distribution with 85% confidence interval

Career Average of Player 6 is between 44.52087815453343 to 31.02457639092112
according to normal distribution with 85% confidence interval
```

```
USING t DISTRIBUTION

Career Average of Player 1 is between 79.39850463946942 to 62.23414842175507
according to T distribution with 95% confidence interval

Career Average of Player 2 is between 72.7915726551507 to 51.91676067818263
according to T distribution with 95% confidence interval

Career Average of Player 3 is between 58.81223759004939 to 45.88776240995061
according to T distribution with 95% confidence interval

Career Average of Player 4 is between 57.85689887132565 to 44.42380288306031
according to T distribution with 85% confidence interval

Career Average of Player 5 is between 70.13802003269973 to 51.983192088512396
according to T distribution with 85% confidence interval

Career Average of Player 6 is between 44.69170099409336 to 30.85375355136119
according to T distribution with 85% confidence interval
```

We can see both distributions are giving almost same values in 95% confidence interval.

iv) Player Prediction

From normal distribution we took the average run range and then calculated average strike rate with same confidence interval. We found closest match to the real data and predicted player.

```
Average SR of Player 1 is between 66.99534683716513 to 56.90360053125594
according to normal distribution with 85% confidence interval

Average SR of Player 2 is between 56.42846848429704 to 50.003454592626
according to normal distribution with 85% confidence interval

Average SR of Player 3 is between 61.71418442600048 to 55.12112807399955
according to normal distribution with 85% confidence interval

Average SR of Player 4 is between 83.28652357580718 to 72.00567981402337
according to normal distribution with 85% confidence interval

Average SR of Player 5 is between 59.43923667752497 to 50.09234226984348
according to normal distribution with 85% confidence interval

Average SR of Player 6 is between 52.43630398297478 to 42.514605107934315
according to normal distribution with 85% confidence interval
```

Player Number	Player Name	Average Run Range from Data	Actual Run Average	Average Strike Rate Range from Data	Actual SR Average
1	Steven Smith	79.30 to 62.32	61.38	66.99 to 56.90	55.49
2	Virat Kohli	72.67 to 52.03	53.40	56.42 to 50.00	58.27
3	Joe Root	58.75 to 45.94	52.29	61.71 to 55.12	55.52
4	David Warner	57.79 to 44.48	48.20	83.28 to 72.00	74.50
5	Kane Williamson	69.99 to 52.13	51.11	59.43 to 50.09	50.43
6	Tamim Iqbal	44.52 to 31.02	38.68	52.43 to 42.51	55.51