In [567]:

```
from IPython.display import Markdown, display
def printmd(string):
    display(Markdown(string))
printmd('**Pulkit Batra\'s creation for Flipr Hackathon 2020**')
```

Pulkit Batra's creation for Flipr Hackathon 2020

In [568]:

```
from IPython.display import Markdown, display
def printmd(string):
    display(Markdown(string))
printmd('**Training Data Set**')
```

Training Data Set

In [569]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_excel(r'C:\Users\Pulkit-PC\Desktop\FliprHackathon\traindata.xlsx')  #importing the cs
v file from the repective directory
df
```

Out[569]:

	PLAYER	Mat	Inns	NO	2018_Runs	HS	Avg	BF	SR	100	50	4s	6s	2019_Runs
0	Aaron Finch	10	9	1	134	46	16.75	100	134.00	0	0	6	8	160
1	AB de Villiers	12	11	2	480	90*	53.33	275	174.54	0	6	39	30	424
2	Abhishek Sharma	3	3	2	63	46*	63	33	190.90	0	0	3	5	63
3	Ajinkya Rahane	15	14	1	370	65*	28.46	313	118.21	0	1	39	5	396
4	Alex Hales	6	6	0	148	45	24.66	118	125.42	0	0	13	6	165
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
95	Virat Kohli	14	14	3	530	92*	48.18	381	139.10	0	4	52	18	488
96	Washington Sundar	7	6	3	65	35	21.66	38	171.05	0	0	5	4	64
97	Wriddhiman Saha	11	10	2	122	35	15.25	102	119.60	0	0	17	1	115
98	Yusuf Pathan	15	13	4	260	45*	28.88	200	130.00	0	0	22	11	296
99	Yuvraj Singh	8	6	0	65	20	10.83	73	89.04	0	0	6	2	67

100 rows × 14 columns

In [570]:

```
cols = df.columns.tolist()
print(cols)
```

```
['PLAYER', 'Mat', 'Inns', 'NO', '2018_Runs', 'HS', 'Avg', 'BF', 'SR', 100, 50, '4s', '6s',
'2019_Runs']
```

In [573]:

```
temp=cols[12]
cols[12]=cols[4]
cols[4]=temp
df = df[cols]
temp=df.copy()  # Will be working on temp now.
```

In [574]:

```
from IPython.display import Markdown, display
def printmd(string):
    display(Markdown(string))
printmd('**Cleaning the data set and setting appropriate data types**')
```

Cleaning the data set and setting appropriate data types

In [575]:

```
temp.dtypes
```

Out[575]:

```
PLAYER        object
Mat            int64
Inns           int64
NO             int64
6s             int64
HS            object
Avg           object
BF             int64
SR           float64
100            int64
50             int64
4s             int64
2018_Runs      int64
2019_Runs      int64
dtype: object
```

In [576]:

```
from IPython.display import Markdown, display
def printmd(string):
    display(Markdown(string))
printmd('**Setting data type of PLAYER to string**')
```

Setting data type of PLAYER to string

In [577]:

```
temp['PLAYER']=temp['PLAYER'].astype('str')
```

In [578]:

```
def printmd(string):
    display(Markdown(string))
printmd('**Setting the data type of HS to int**')
```

Setting the data type of HS to int

In [579]:

```
temp['HS']=temp['HS'].astype('str')
temp['HS'] = temp['HS'].str.replace('*', '')
temp['HS']=temp['HS'].astype('int')
```

In [580]:

```
def printmd(string):
    display(Markdown(string))
printmd('**Setting the data type of Avg to float and assigning average to players who have only re
mained not out throughout**')
```

Setting the data type of Avg to float and assigning average to players who have only remained not out throughout

Setting the data type of Avg to float and assigning average to players who have only remained not out throughout

In [581]:

```
temp['Avg']=temp['Avg'].astype('str')
temp['Avg'] = temp['Avg'].str.replace('-', '-1')
temp['Avg']=temp['Avg'].astype('float')
```

In [582]:

```
index=0
for i in temp['Avg']:
    #print(i)
    if i==-1.0:
        t=temp.loc[[index]]
        t['Avg']=float(t['HS'])
        #print(t)
    index=index+1
```

In [583]:

```
def printmd(string):
    display(Markdown(string))
printmd('**Now the data type of each column is appropriate and all the null/no values have been fi
xed.**')
```

Now the data type of each column is appropriate and all the null/no values have been fixed.

In [584]:

```
temp.dtypes
```

Out[584]:

```
PLAYER        object
Mat            int64
Inns           int64
NO             int64
6s             int64
HS             int32
Avg          float64
BF             int64
SR           float64
100            int64
50             int64
4s             int64
2018_Runs      int64
2019_Runs      int64
dtype: object
```

In [585]:

```
def printmd(string):
    display(Markdown(string))
printmd('**Time to create and train our model**')
```

Time to create and train our model

In [586]:

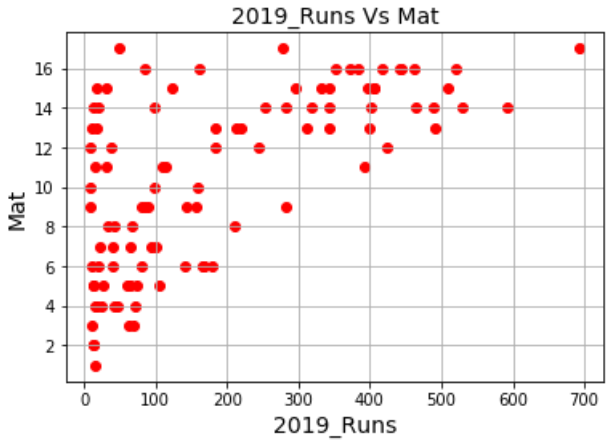
```
print(cols)
```

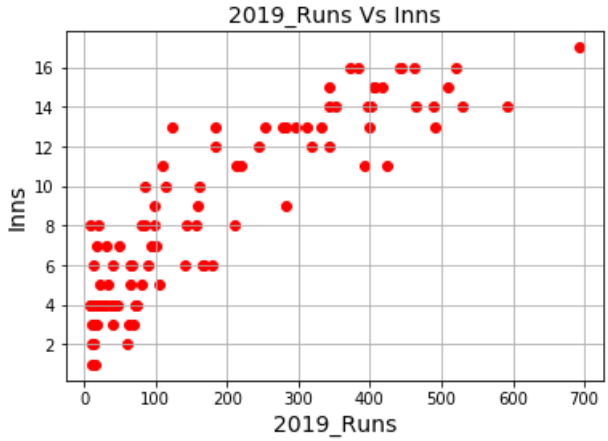
```
['PLAYER', 'Mat', 'Inns', 'NO', '6s', 'HS', 'Avg', 'BF', 'SR', 100, 50, '4s', '2018_Runs',
'2019_Runs']
```

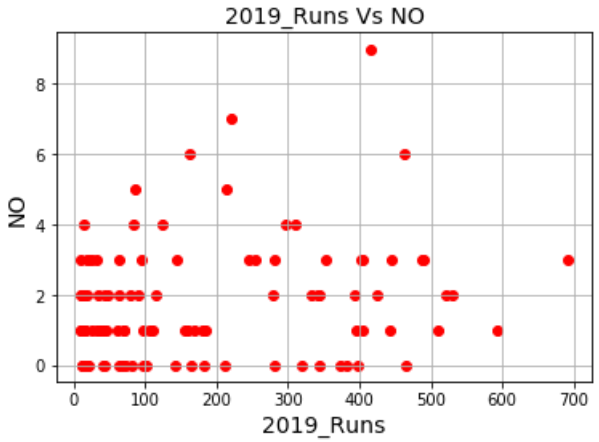
In [587]:

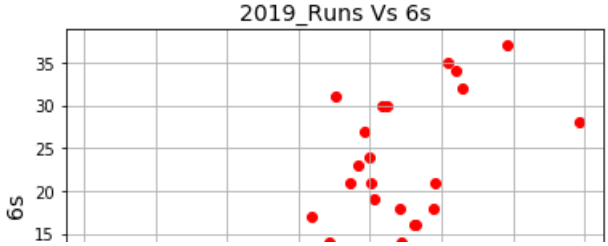
```
from pandas import DataFrame
```

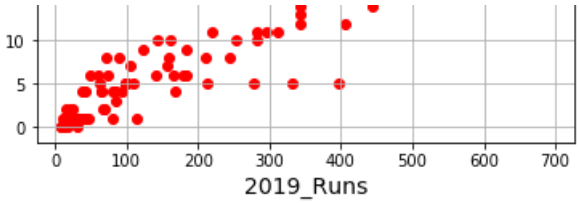
```
import matplotlib.pyplot as plt

for i in temp.columns:
    if (i!='PLAYER') and (i!='2019_Runs'):
        plt.scatter(temp['2019_Runs'], temp[i], color='red')
        ti='2019_Runs Vs ' + str(i)
        plt.title(ti, fontsize=14)
        plt.xlabel('2019_Runs', fontsize=14)
        plt.ylabel(i, fontsize=14)
        plt.grid(True)
        plt.show()
```

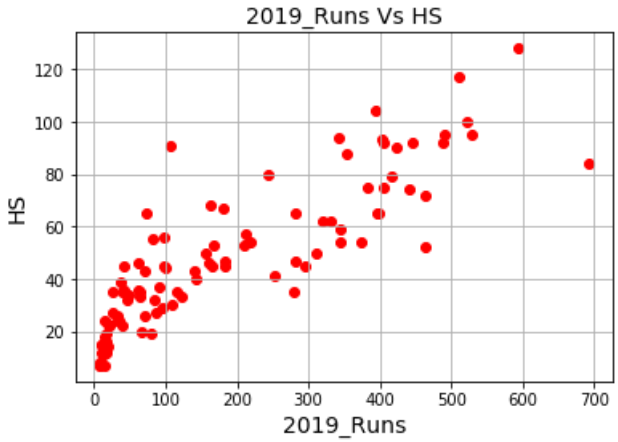


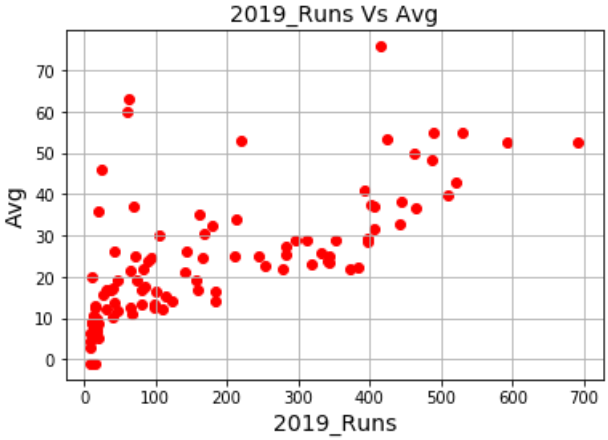


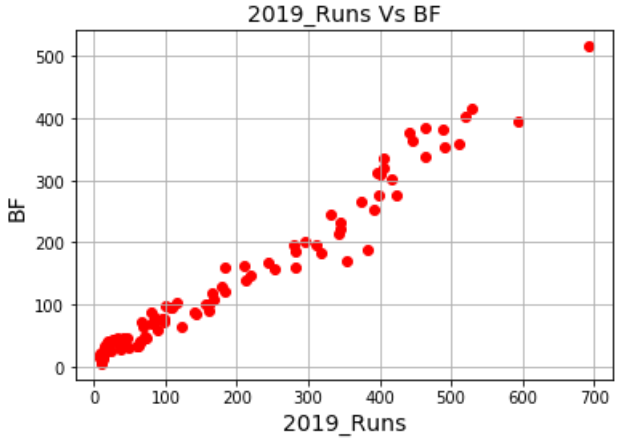


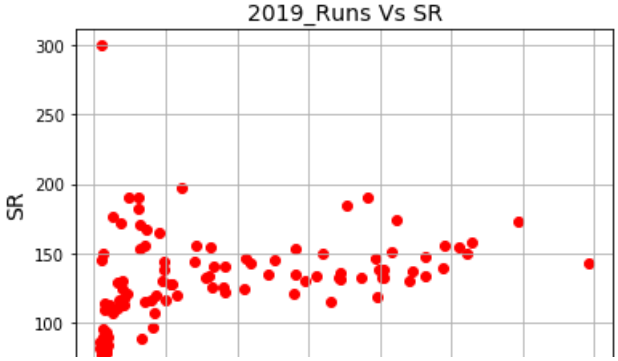


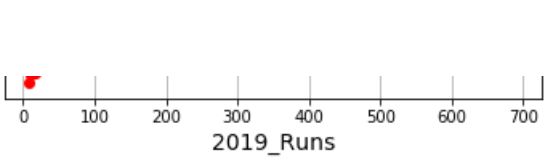

10
5
0
0
100
200
300
400
500
600
700
2019_Runs

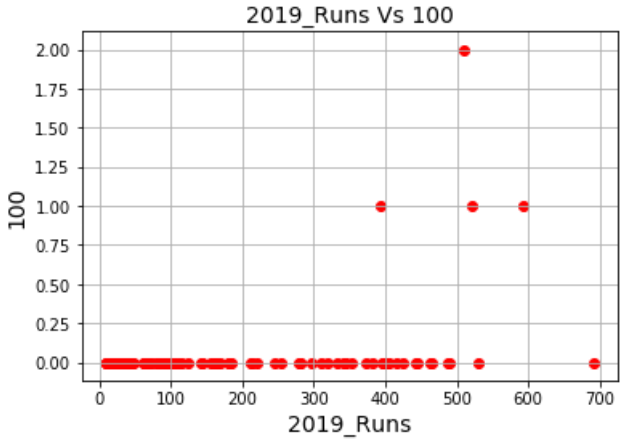

2019_Runs Vs HS
HS
120
100
80
60
40
20
0
100
200
300
400
500
600
700
2019_Runs

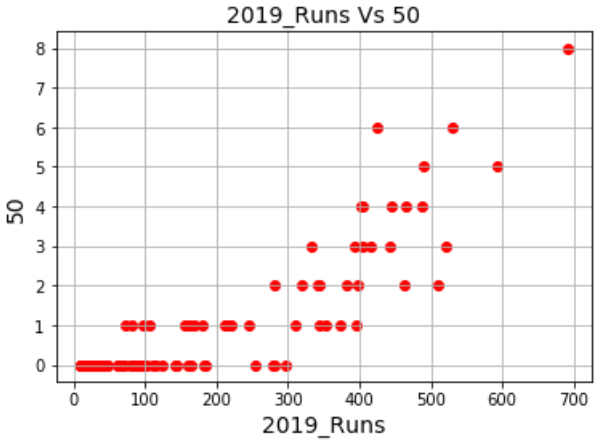

2019_Runs Vs Avg
Avg
70
60
50
40
30
20
10
0
0
100
200
300
400
500
600
700
2019_Runs

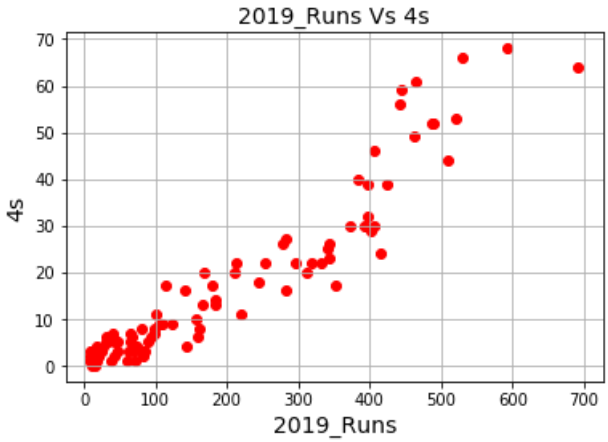

2019_Runs Vs BF
BF
500
400
300
200
100
0
0
100
200
300
400
500
600
700
2019_Runs

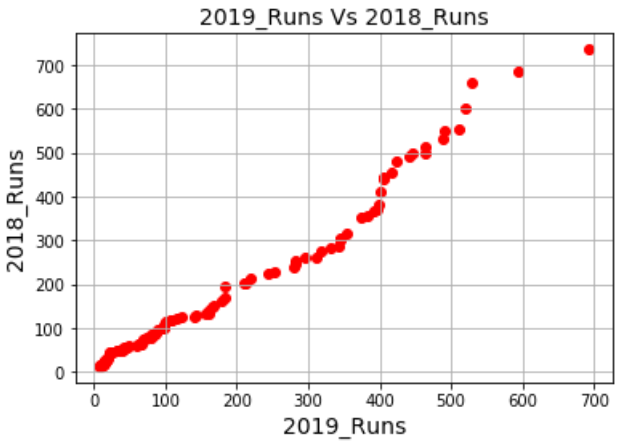

2019_Runs Vs SR
SR
300
250
200
150
100


0
100
200
300
400
500
600
700
2019_Runs


2019_Runs Vs 100
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
100
0
100
200
300
400
500
600
700
2019_Runs


2019_Runs Vs 50
8
7
6
5
4
3
2
1
0
50
0
100
200
300
400
500
600
700
2019_Runs


2019_Runs Vs 4s
70
60
50
40
30
20
10
0
4s
0
100
200
300
400
500
600
700
2019_Runs


2019_Runs Vs 2018_Runs
700
600
500
400
300
200
100
0
2018_Runs
0
100
200
300
400
500
600
700
2019_Runs

In [588]:

```
def printmd(string):
    display(Markdown(string))
printmd('**After analyzing the graphs for 2018_Runs vs other parameters, the following conclusions
occur -**')
printmd('**1. Mat(Matches): There is no definite relation between matches played and runs as bowle
rs might also play many matches but score less runs.**')
printmd('**2. Inns(Innings): Graph is indicating that more innings is leading to more runs.**')
printmd('**3. NO(Not Outs): There is no definite relation between number of not outs and runs as r
emaining not out doesnt necessarily mean that runs have been scored.**')
printmd('**4. 6s(Sixes): Graph is indicating that more number of sixes is leading to more runs.**'
)
printmd('**5. HS(Highest Score): Graph is indicating that players who have scored more runs tend t
o have a greater highest score.**')
printmd('**5. Avg(Average): Graph is indicating that players who have scored more runs tend to hav
e a greater average.**')
printmd('**6. BF(Balls Faced): The number of balls faced is our first breakthrough here. It is giv
ing the most clear indication that players who have scored more runs have faced more balls.**')
printmd('**7. SR(Strike Rate): There is no definite relation between strike rate and runs.**')
printmd('**8. 100(100s scored): There is no definite relation between 100s scored and runs.**')
printmd('**9. 50(50s scored): Not a very definite relation.**')
printmd('**10. 4s(Sixes): Graph is indicating that more number of fours is leading to more
runs.**')
printmd('**11. 2018_Runs(Runs in 2018): The number of runs in 2018 is our second breakthrough here
. It is giving the most clear indication that players who have scored more runs in 2019 have score
d more runs in 2018.**')
```

After analyzing the graphs for 2018_Runs vs other parameters, the following conclusions occur -

1. Mat(Matches): There is no definite relation between matches played and runs as bowlers might also play many matches but score less runs.

2. Inns(Innings): Graph is indicating that more innings is leading to more runs.

3. NO(Not Outs): There is no definite relation between number of not outs and runs as remaining not out doesnt necessarily mean that runs have been scored.

4. 6s(Sixes): Graph is indicating that more number of sixes is leading to more runs.

5. HS(Highest Score): Graph is indicating that players who have scored more runs tend to have a greater highest score.

5. Avg(Average): Graph is indicating that players who have scored more runs tend to have a greater average.

6. BF(Balls Faced): The number of balls faced is our first breakthrough here. It is giving the most clear indication that players who have scored more runs have faced more balls.

7. SR(Strike Rate): There is no definite relation between strike rate and runs.

8. 100(100s scored): There is no definite relation between 100s scored and runs.

9. 50(50s scored): Not a very definite relation.

10. 4s(Sixes): Graph is indicating that more number of fours is leading to more runs.

11. 2018_Runs(Runs in 2018): The number of runs in 2018 is our second breakthrough here. It is giving the most clear indication that players who have scored more runs in 2019 have scored more runs in 2018.

In [589]:

```
def printmd(string):
    display(Markdown(string))
printmd('**Based on the conclusion, the parameters that we will consider to train our model will b
e :**')
printmd('**Inns, 6s, HS, Avg, BF, 4s,2018_Runs**')
```

Based on the conclusion, the parameters that we will consider to train our model will be :

Inns, 6s, HS, Avg, BF, 4s,2018_Runs

In [590]:

```
def printmd(string):
    display(Markdown(string))
printmd('**Applying Multi Variate Linear Regression**')
```

Applying Multi Variate Linear Regression

In [591]:

```
from pandas import DataFrame
from sklearn import linear_model
import statsmodels.api as sm
from sklearn.metrics import mean_squared_error, r2_score
import math

X = temp[['2018_Runs','BF','Avg','Inns','6s','4s','HS']] # here we have 2 variables for multiple r
egression. If you just want to use one variable for simple linear regression, then use X = df['Int
erest_Rate'] for example.Alternatively, you may add additional variables within the brackets
Y = temp['2019_Runs']

# with sklearn
regr = linear_model.LinearRegression()
regr.fit(X, Y)

#print('Intercept: \n', regr.intercept_)
#print('Coefficients: \n', regr.coef_)


for index, row in temp.iterrows():
    Inns = row['Inns']
    s6 = row['6s']
    HS = row['HS']
    Avg = row['Avg']
    BF=row['BF']
    s4=row['4s']
    Runs=row['2018_Runs']
    p=regr.predict([[Runs,BF,Avg,Inns,s6,s4,HS]])
    #print ('Predicted Runs:', math.floor(abs(p[0])) , ' Actual Runs : ',row['2019_Runs'])

Y1=regr.predict(X)
rmse = mean_squared_error(Y,Y1)
r2 = r2_score(Y, Y1)
#print('Variance score: %.2f' % regr.score(X, Y))
print('R2 score: ', r2)
```

```
R2 score:  0.9848913675237715
```

In [592]:

```
def printmd(string):
    display(Markdown(string))
printmd('**Our training model has yielded quite good results. Now lets test it on test data.**')
```

Our training model has yielded quite good results. Now lets test it on test data.

In [593]:

```
from IPython.display import Markdown, display
def printmd(string):
    display(Markdown(string))
printmd('**Testing Data Set**')
```

Testing Data Set

In [594]:

```
dftest = pd.read_excel(r'C:\Users\Pulkit-PC\Desktop\FliprHackathon\testdata.xlsx')  #importing the csv file from the repective directory
dftest
```

Out[594]:

	PLAYER	Mat	Inns	NO	2019_Runs	HS	Avg	BF	SR	100	50	4s	6s
0	David Warner	12	12	2	692	100*	69.2	481	143.86	1	8	57	21
1	KL Rahul	14	14	3	593	100*	53.9	438	135.38	1	6	49	25
2	Quinton de Kock	16	16	1	529	81	35.26	398	132.91	0	4	45	25
3	Shikhar Dhawan	16	16	1	521	97*	34.73	384	135.67	0	5	64	11
4	Andre Russell	14	13	4	510	80*	56.66	249	204.81	0	4	31	52
...	...	...	...	...	...	...	...	...	...	...	...	...	...
95	Carlos Brathwaite	2	2	0	11	6	5.5	10	110.00	0	0	1	0
96	Ishant Sharma	13	3	3	10	10*	-	3	333.33	0	0	1	1
97	Shakib Al Hasan	3	1	0	9	9	9	10	90.00	0	0	0	0
98	Pawan Negi	7	4	0	9	5	2.25	12	75.00	0	0	1	0
99	Tim Southee	3	1	1	9	9*	-	9	100.00	0	0	0	0

100 rows × 13 columns

In [595]:

```
temp2=dftest.copy()
temp2.dtypes
```

Out[595]:

```
PLAYER        object
Mat            int64
Inns           int64
NO             int64
2019_Runs      int64
HS            object
Avg           object
BF             int64
SR           float64
100            int64
50             int64
4s             int64
6s             int64
dtype: object
```

In [596]:

```
from IPython.display import Markdown, display
def printmd(string):
    display(Markdown(string))
printmd('**Cleaning the data set and setting appropriate data types**')
```

Cleaning the data set and setting appropriate data types

In [597]:

```
temp2['PLAYER']=temp2['PLAYER'].astype('str')
temp2['HS']=temp2['HS'].astype('str')
temp2['HS'] = temp2['HS'].str.replace('*', '')
temp2['HS']=temp2['HS'].astype('int')
temp2['Avg']=temp2['Avg'].astype('str')
temp2['Avg'] = temp2['Avg'].str.replace('-', '-1')
temp2['Avg']=temp2['Avg'].astype('float')
index=0
```

```
for i in temp2['Avg']:
    #print(i)
    if i==-1.0:
        t=temp2.loc[[index]]
        t['Avg']=float(t['HS'])
        #print(t)
    index=index+1
```

In [598]:

```
temp2.dtypes
```

Out[598]:

```
PLAYER        object
Mat            int64
Inns           int64
NO             int64
2019_Runs      int64
HS             int32
Avg          float64
BF             int64
SR           float64
100            int64
50             int64
4s             int64
6s             int64
dtype: object
```

In [599]:

```
from IPython.display import Markdown, display
def printmd(string):
    display(Markdown(string))
printmd('**Testing our model on test data set and predicting values for 2020**')
```

Testing our model on test data set and predicting values for 2020

In [600]:

```
predicted_price={}

for index, row in temp2.iterrows():
    name = row['PLAYER']
    Inns = row['Inns']
    s6 = row['6s']
    HS = row['HS']
    Avg = row['Avg']
    BF=row['BF']
    s4=row['4s']
    Runs=row['2019_Runs']
    p=regr.predict([[Runs,BF,Avg,Inns,s6,s4,HS]])
    #print (name + ' Predicted Runs in 2020 :', math.floor(abs(p[0])) , ' ,2019 Runs :
',row['2019_Runs'])
    predicted_price[index]=math.floor(abs(p[0]))
```

In [601]:

```
from IPython.display import Markdown, display
def printmd(string):
    display(Markdown(string))
printmd('**Writing into a csv file**')
```

Writing into a csv file

In [602]:

```
import csv

with open('solution.csv', mode='w+',newline='') as solution:
```

```
    sol = csv.writer(solution, delimiter=',', quotechar='"', quoting=csv.QUOTE_MINIMAL)
    sol.writerow(['Player_Name','Predicted_Runs_in_2020'])
    i=0
    for index, row in temp2.iterrows():
        runs = str(predicted_price[i])
        name=row['PLAYER']
        sol.writerow([name,runs])
        i=i+1
```