In [1]:

```
import pandas as pd
import numpy as np

xl6 = pd.read_excel("Matches2016.xlsx", sheet_name='NData')
xl7 = pd.read_excel("Matches2017.xlsx", sheet_name='NData')
xl8 = pd.read_excel("Matches2018.xlsx", sheet_name='NData')

el6 = pd.read_excel("Matches2016.xlsx", sheet_name='EData')
el7 = pd.read_excel("Matches2017.xlsx", sheet_name='EData')
el8 = pd.read_excel("Matches2018.xlsx", sheet_name='EData')

esl6 = pd.read_excel("Matches2016.xlsx", sheet_name='E2Data')
esl7 = pd.read_excel("Matches2017.xlsx", sheet_name='E2Data')
esl8 = pd.read_excel("Matches2018.xlsx", sheet_name='E2Data')

ewr6 = pd.read_excel("Matches2016.xlsx", sheet_name='WRData')
ewr7 = pd.read_excel("Matches2017.xlsx", sheet_name='WRData')
ewr8 = pd.read_excel("Matches2018.xlsx", sheet_name='WRData')

ef6 = pd.read_excel("Matches2016.xlsx", sheet_name='FData')
ef7 = pd.read_excel("Matches2017.xlsx", sheet_name='FData')
ef8 = pd.read_excel("Matches2018.xlsx", sheet_name='FData')

r8 = pd.read_excel("Matches2018.xlsx", sheet_name='RData')

#data6 = np.array(xl6)
#data7 = np.array(xl7)
#data8 = np.array(xl8)

result6 = xl6.iloc[:,0].astype('int')
result7 = xl7.iloc[:,0].astype('int')
result8 = xl8.iloc[:,0].astype('int')
```

DATA 2016

In [2]:

```
data6 = xl6.iloc[:,1:159]
datae6 = el6.iloc[:,1:81]
dataes6 = esl6.iloc[:,1:107]
datawr6 = ewr6.iloc[:,[1,3]]
dataf6 = ef6.iloc[:,1:110]
```

DATA 2017

In [3]:

```
data7 = xl7.iloc[:,1:159]
datae7 = el7.iloc[:,1:81]
dataes7 = esl7.iloc[:,1:107]
datawr7 = ewr7.iloc[:,[1,3]]
dataf7 = ef7.iloc[:,1:110]
```

DATA 2018

In [4]:

```
data8 = xl8.iloc[:,1:159]
datae8 = el8.iloc[:,1:81]
dataes8 = esl8.iloc[:,1:107]
datawr8 = ewr8.iloc[:,[1,3]]
dataf8 = ef8.iloc[:,1:110]

datar8 = r8
```

In [5]:

```
import time

def tic():
    global start_time
    start_time = time.time()

def tac():
    global t_used
    t_used = time.time() - start_time
    print('Time Used: {} s'.format(t_used))
```

Prediction of Regular Player

In [6]:

```
from sklearn.naive_bayes import GaussianNB

x_train = np.concatenate((data6,data7))
y_train = np.concatenate((result6,result7))

tic()
model = GaussianNB()
model.fit(x_train,y_train)
tac()

t_train = t_used
```

```
Time Used: 0.019565582275390625 s
```

In [7]:

```
record8 = data8

tic()
predicted = model.predict(record8)
tac()

t_test = t_used

print(predicted)
```

```
Time Used: 0.0156404972076416 s
[1 1 0 ... 0 0 1]
```

In [8]:

```
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score

acc = accuracy_score(result8,predicted)

print("\nAccuracy: {0:.6f}\n".format(acc))
print("\nTime Used(Training): {0:.6f} s\n".format(t_train))
print("\nTime Used(Prediction): {0:.6f} s\n".format(t_test))
print("\nConfusion Matrix: \n\n {}\n".format(confusion_matrix(result8,predicted)))
print("\nClassification Report:\n\n {}\n".format(classification_report(result8,predicted)))
```

```
Accuracy: 0.606504


Time Used(Training): 0.019566 s


Time Used(Prediction): 0.015640 s


Confusion Matrix:

 [[308 210]
 [274 438]]


Classification Report:

              precision    recall  f1-score   support

           0       0.53      0.59      0.56       518
           1       0.68      0.62      0.64       712

   micro avg       0.61      0.61      0.61      1230
   macro avg       0.60      0.60      0.60      1230
weighted avg       0.61      0.61      0.61      1230
```

Prediction After First Extration

In [9]:

```
xe_train = np.concatenate((datae6,datae7))
ye_train = np.concatenate((result6,result7))

tic()
model = GaussianNB()
model.fit(xe_train,ye_train)
tac()

te_train = t_used
```

```
Time Used: 0.006844043731689453 s
```

In [10]:

```
tic()
predictede = model.predict(datae8)
tac()

te_test = t_used

print(predictede)
```

```
Time Used: 0.004889011383056641 s
[0 1 0 ... 0 1 0]
```

In [11]:

```
acce = accuracy_score(result8,predictede)

print("\nAccuracy: {0:.6f}\n".format(acce))
print("\nTime Used(Training): {0:.6f} s\n".format(te_train))
print("\nTime Used(Prediction): {0:.6f} s\n".format(te_test))
print("\nConfusion Matrix: \n\n {}\n".format(confusion_matrix(result8,predictede)))
print("\nClassification Report:\n\n {}\n".format(classification_report(result8,predictede)))
```

```
Accuracy: 0.625203


Time Used(Training): 0.006844 s


Time Used(Prediction): 0.004889 s


Confusion Matrix:

 [[288 230]
 [231 481]]


Classification Report:

              precision    recall  f1-score   support

           0       0.55      0.56      0.56       518
           1       0.68      0.68      0.68       712

   micro avg       0.63      0.63      0.63      1230
   macro avg       0.62      0.62      0.62      1230
weighted avg       0.63      0.63      0.63      1230
```

Prediction After Second Extration

In [12]:

```
xe2_train = np.concatenate((dataes6,dataes7))
ye2_train = np.concatenate((result6,result7))

tic()
model = GaussianNB()
model.fit(xe2_train,ye2_train)
tac()

te2_train = t_used
```

```
Time Used: 0.011730670928955078 s
```

In [13]:

```
tic()
predictede2 = model.predict(dataes8)
tac()

te2_test = t_used

print(predictede2)
```

```
Time Used: 0.007820844650268555 s
[0 1 1 ... 0 1 0]
```

In [14]:

```
acce2 = accuracy_score(result8,predictede2)

print("\nAccuracy: {0:.6f}\n".format(acce2))
print("\nTime Used(Training): {0:.6f} s\n".format(te2_train))
print("\nTime Used(Prediction): {0:.6f} s\n".format(te2_test))
print("\nConfusion Matrix: \n\n {}\n".format(confusion_matrix(result8,predictede2)))
print("\nClassification Report:\n\n {}\n".format(classification_report(result8,predictede2)
```

```
Accuracy: 0.625203


Time Used(Training): 0.011731 s


Time Used(Prediction): 0.007821 s


Confusion Matrix:

 [[293 225]
 [236 476]]


Classification Report:

              precision    recall  f1-score   support

           0       0.55      0.57      0.56       518
           1       0.68      0.67      0.67       712

   micro avg       0.63      0.63      0.63      1230
   macro avg       0.62      0.62      0.62      1230
weighted avg       0.63      0.63      0.63      1230
```

Testing Purpose

In [15]:

```
xwr_train = np.concatenate((datawr6,datawr7))
ywr_train = np.concatenate((result6,result7))

tic()
model = GaussianNB()
model.fit(xwr_train,ywr_train)
tac()

twr_train = t_used
```

```
Time Used: 0.002933502197265625 s
```

In [16]:

```
tic()
predictedwr = model.predict(datawr8)
tac()

twr_test = t_used

print(predictedwr)
```

```
Time Used: 0.0019559860229492188 s
[0 0 1 ... 1 1 0]
```

In [17]:

```
accwr = accuracy_score(result8,predictedwr)

print("\nAccuracy: {0:.6f}\n".format(accwr))
print("\nTime Used(Training): {0:.6f} s\n".format(twr_train))
print("\nTime Used(Prediction): {0:.6f} s\n".format(twr_test))
print("\nConfusion Matrix: \n\n {}\n".format(confusion_matrix(result8,predictedwr)))
print("\nClassification Report:\n\n {}\n".format(classification_report(result8,predictedwr)
```

```
Accuracy: 0.702439


Time Used(Training): 0.002934 s


Time Used(Prediction): 0.001956 s


Confusion Matrix:

 [[302 216]
 [150 562]]


Classification Report:

              precision    recall  f1-score   support

           0       0.67      0.58      0.62       518
           1       0.72      0.79      0.75       712

   micro avg       0.70      0.70      0.70      1230
   macro avg       0.70      0.69      0.69      1230
weighted avg       0.70      0.70      0.70      1230
```

Prediction After Final Extration

In [18]:

```
xf_train = np.concatenate((dataf6,dataf7))
yf_train = np.concatenate((result6,result7))

tic()
model = GaussianNB()
model.fit(xf_train,yf_train)
tac()

tf_train = t_used
```

```
Time Used: 0.011732101440429688 s
```

In [19]:

```
tic()
predictedf = model.predict(dataf8)
tac()

tf_test = t_used

print(predictedf)
```

```
Time Used: 0.007821321487426758 s
[0 1 0 ... 0 1 0]
```

In [20]:

```
accf = accuracy_score(result8,predictedf)

print("\nAccuracy: {0:.6f}\n".format(accf))
print("\nTime Used(Training): {0:.6f} s\n".format(tf_train))
print("\nTime Used(Prediction): {0:.6f} s\n".format(tf_test))
print("\nConfusion Matrix: \n\n {}\n".format(confusion_matrix(result8,predictedf)))
print("\nClassification Report:\n\n {}\n".format(classification_report(result8,predictedf)))
```

```
Accuracy: 0.716260


Time Used(Training): 0.011732 s


Time Used(Prediction): 0.007821 s


Confusion Matrix:

 [[346 172]
 [177 535]]


Classification Report:

              precision    recall  f1-score   support

           0       0.66      0.67      0.66       518
           1       0.76      0.75      0.75       712

   micro avg       0.72      0.72      0.72      1230
   macro avg       0.71      0.71      0.71      1230
weighted avg       0.72      0.72      0.72      1230
```

Prediction of the Analysis

In [35]:

```
ta6 = pd.read_excel("DataMatches.xlsx", sheet_name='2016')
ta7 = pd.read_excel("DataMatches.xlsx", sheet_name='2017')
ta8 = pd.read_excel("DataMatches.xlsx", sheet_name='2018')

datata6 = ta6.iloc[:,1:109]
datata7 = ta7.iloc[:,1:109]
datata8 = ta8.iloc[:,1:109]

xta_train = np.concatenate((datata6,datata7,datata8))
yta_train = np.concatenate((result6,result7,result8))

tic()
model = GaussianNB()
model.fit(xta_train,yta_train)
tac()

tta_train = t_used
```

```
Time Used: 0.010754108428955078 s
```

Round Robin 2018

In [36]:

```
tic()
predictedr = model.predict(datar8)
tac()

tr_test = t_used

print(predictedr)
```

```
Time Used: 0.0019555091857910156 s
[1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 1 1 0 1 1 1 0 0 0 0 1 1 1 0 0 0 1 0 1 1 1
 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 1 1 1 1 1 1 0 0 1 0 1 1 1
 1 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 0 1 0 0 0 0 0 1 0 1 0 0 0 1 1 0 1
 1 0 0 0 0 1 0 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 0 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 0 0 0 1 0
 1 0 0 0 0 0 0 0 1 0 0 0 0 1 0 1 0 0 0 0 0 0 1 1 0 1 1 1 1 1 1 1 1 0 0 1 1
 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1]
```

Prediction of NBA Season 2018-19

In [37]:

```
ta9 = pd.read_excel("DataMatches.xlsx", sheet_name='2019')

datata9 = ta9.iloc[:,1:109]

result9 = ta9.iloc[:,0].astype('int')

xta2_train = np.concatenate((datata6,datata7,datata8,datata9))
yta2_train = np.concatenate((result6,result7,result8,result9))

tic()
model = GaussianNB()
model.fit(xta2_train,yta2_train)
tac()

ttp9_train = t_used
```

```
Time Used: 0.01564192771911621 s
```

In [38]:

```
p9 = pd.read_excel("NBA2019STAT.xlsx", sheet_name='PData')

datatp9 = p9.iloc[:,0:108]

tic()
predictedp9 = model.predict(datatp9)
tac()

ttp9_test = t_used

print(predictedp9)
```

```
Time Used: 0.0009777545928955078 s
[1 1 0 1 0 0 1 1 0 1 1 0 1 1 0 0 0 1 1 1 0 0 1 0 1 1 1 0 1 1 1 0 0 1 0 0 1
 1 1 1 1 0 1 0 0 1 1 0 1 0 0 1 0 1 1 0 1 1 0 1 1 1 1 1 1 0 1 1 0 1 1 0 1 1
 0 0 1 1 1 1 1 1 0 0 1 1 1 1 0 1 1 1 0 0 0 1 1 0 0 0 1 1 0 0 1 0 1 1 1]
```

In [ ]: