In [84]:

```
#   With weights 3:1
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder

from sklearn.model_selection import train_test_split
# import dataset
```

In [85]:

```
Data1 = pd.read_csv('/Users/alialghaithi/Box/Advance NL Class/Discussion3/Data1.csv')
Data10 = pd.read_csv('/Users/alialghaithi/Box/Advance NL Class/Discussion3/Data10.csv')
Data30 = pd.read_csv('/Users/alialghaithi/Box/Advance NL Class/Discussion3/Data30.csv')
```

30% Data

wrf : With weights 2:1

In [86]:

```
y = Data30['Class']
x = Data30.drop('Class', axis=1)

from sklearn.ensemble import RandomForestClassifier
w = 2 # The weight for the positive class

#RF = RandomForestClassifier(class_weight={0: 1, 1: w})
RF = RandomForestClassifier(class_weight={0: 1, 1: w})

#RF.fit(X_train, y_train,sample_weight=np.array([2 if i == 0 else 1 for
 i in y]))
RF.fit(x, y,sample_weight=None)
from sklearn.metrics import confusion_matrix


y_pred = RF.predict(x)
tn, fp, fn, tp = confusion_matrix(y, y_pred).ravel()
recall = tp / (tp + fn)
prec = tp / (tp + fp)
F1 = 2 * recall * prec / (recall + prec)
Tnr= tn/(tn+fp)
Tpr= tp/(tp+fn)
g_mean=(Tnr*Tpr) ** 0.5
beta=0.5
Wt_accuracy= beta*Tpr + (1-beta)*Tnr

print(Tpr,Tnr,prec,F1,g_mean,Wt_accuracy)

pd.crosstab(y_pred, np.array(y),
            rownames=['Predicted'], colnames=['Actual'])
```

```
1.0 1.0 1.0 1.0 1.0 1.0
```

Out[86]:

Actual	**0**	**1**
Predicted		
0	1148	0
1	0	492

10% Data

wrf : With weights 2:1

In [87]:

```
y = Data10['Class']
x = Data10.drop('Class', axis=1)

from sklearn.ensemble import RandomForestClassifier
w = 2 # The weight for the positive class

#RF = RandomForestClassifier(class_weight={0: 1, 1: w})
RF = RandomForestClassifier(class_weight={0: 1, 1: w})

#RF.fit(X_train, y_train,sample_weight=np.array([2 if i == 0 else 1 for
 i in y]))
RF.fit(x, y,sample_weight=None)
from sklearn.metrics import confusion_matrix


y_pred = RF.predict(x)
tn, fp, fn, tp = confusion_matrix(y, y_pred).ravel()
recall = tp / (tp + fn)
prec = tp / (tp + fp)
F1 = 2 * recall * prec / (recall + prec)
Tnr= tn/(tn+fp)
Tpr= tp/(tp+fn)
g_mean=(Tnr*Tpr) ** 0.5
beta=0.5
Wt_accuracy= beta*Tpr + (1-beta)*Tnr

print(Tpr,Tnr,prec,F1,g_mean,Wt_accuracy)

pd.crosstab(y_pred, np.array(y),
            rownames=['Predicted'], colnames=['Actual'])
```

```
1.0 1.0 1.0 1.0 1.0 1.0
```

Out[87]:

Actual	0	1
Predicted		
0	4428	0
1	0	492

1% Data

wrf : With weights 3:1

In [88]:

```
y = Data1['Class']
x = Data1.drop('Class', axis=1)

from sklearn.ensemble import RandomForestClassifier
w = 3 # The weight for the positive class

#RF = RandomForestClassifier(class_weight={0: 1, 1: w})
RF = RandomForestClassifier(class_weight={0: 1, 1: w})

RF.fit(x, y,sample_weight=None)
from sklearn.metrics import confusion_matrix


y_pred = RF.predict(x)
tn, fp, fn, tp = confusion_matrix(np.array(y), y_pred).ravel()
recall = tp / (tp + fn)
prec = tp / (tp + fp)
F1 = 2 * recall * prec / (recall + prec)
Tnr= tn/(tn+fp)
Tpr= tp/(tp+fn)
g_mean=(Tnr*Tpr) ** 0.5
beta=0.5
Wt_accuracy= beta*Tpr + (1-beta)*Tnr

print(Tpr,Tnr,prec,F1,g_mean,Wt_accuracy)
pd.crosstab(y_pred, np.array(y),
            rownames=['Predicted'], colnames=['Actual'])
```

```
0.9979674796747967 1.0 1.0 0.9989827060020345 0.9989832229195827 0.9989
837398373984
```

Out[88]:

Actual	**0**	**1**
Predicted		
0	48708	1
1	0	491

In [ ]: