In [1]:
```python
import numpy as np
import pandas as pd
import seaborn as sns

from sklearn.model_selection import train_test_split

from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import confusion_matrix

import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```python
twitch = pd.read_csv("twitchdata-update.csv")
twitch = twitch.loc[0:500,]
twitch.head()
```

Out[2]:

	Channel	Watch time(Minutes)	Stream time(minutes)	Peak viewers	Average viewers	Followers	Followers gained	
0	xQcOW	6196161750	215250	222720	27716	3246298	1734810	9303
1	summit1g	6091677300	211845	310998	25610	5310163	1370184	897(
2	Gaules	5644590915	515280	387315	10976	1767635	1023779	1026
3	ESL_CSGO	3970318140	517740	300575	7714	3944850	703986	1065
4	Tfue	3671000070	123660	285644	29602	8938903	2068424	7899

In [3]:
```python
twitch2 = pd.get_dummies(twitch, columns = ["Mature"], drop_first = True
)
twitch2.head(20)
```

Out[3]:

	Channel	Watch time(Minutes)	Stream time(minutes)	Peak viewers	Average viewers	Followers	Followers gained
0	xQcOW	6196161750	215250	222720	27716	3246298	1734810
1	summit1g	6091677300	211845	310998	25610	5310163	1370184
2	Gaules	5644590915	515280	387315	10976	1767635	1023779
3	ESL_CSGO	3970318140	517740	300575	7714	3944850	703986
4	Tfue	3671000070	123660	285644	29602	8938903	2068424
5	Asmongold	3668799075	82260	263720	42414	1563438	554201
6	NICKMERCS	3360675195	136275	115633	24181	4074287	1089824
7	Fextralife	3301867485	147885	68795	18985	508816	425468
8	loltyler1	2928356940	122490	89387	22381	3530767	951730
9	Anomaly	2865429915	92880	125408	12377	2607076	1532689
10	TimTheTatman	2834436990	108780	142067	25664	5265659	1244341
11	LIRIK	2832930285	128490	89170	21739	2666382	199077
12	Riot Games (riotgames)	2674646715	80820	639375	20960	4487489	497678
13	Rubius	2588632635	58275	240096	42948	5751354	3820532
14	auronplay	2410022550	40575	170115	53986	3983847	3966525
15	MontanaBlack88	2408460990	67740	181600	33514	2911316	1101093
16	sodapoppin	2329440420	115305	107833	19659	2786162	236169
17	풍월량 (hanryang1125)	2186662470	181230	26999	12201	494445	92205
18	alanzoka	2055003870	103770	89153	19560	3445134	1325075
19	CohhCarnage	2029212570	175230	43615	11343	1264808	124242

```
In [4]: twitch3 = twitch2[['Mature_True',"Watch time(Minutes)","Stream time(minu
        tes)"]]
        twitch3.head()
```

Out[4]:

	Mature_True	Watch time(Minutes)	Stream time(minutes)
0	0	6196161750	215250
1	0	6091677300	211845
2	1	5644590915	515280
3	0	3970318140	517740
4	0	3671000070	123660

```
In [5]: twitch3["Mature_True"] = twitch3["Mature_True"].astype(float)
```

```
/usr/local/lib/python3.4/site-packages/ipykernel_launcher.py:1: Setting
WithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-d
ocs/stable/indexing.html#indexing-view-versus-copy
  """Entry point for launching an IPython kernel.
```

```
In [6]: x = twitch3 [["Watch time(Minutes)", "Stream time(minutes)"]]
        x.head()
```

Out[6]:

	Watch time(Minutes)	Stream time(minutes)
0	6196161750	215250
1	6091677300	211845
2	5644590915	515280
3	3970318140	517740
4	3671000070	123660

```
In [7]: y = twitch3 ["Mature_True"]
        y.head()
```

```
Out[7]: 0     0.0
        1     0.0
        2     1.0
        3     0.0
        4     0.0
        Name: Mature_True, dtype: float64
```

```
In [8]: x_train,x_test,y_train,y_test = train_test_split(x,y, test_size = 0.2)
```

```
In [9]: knn4 = KNeighborsClassifier(n_neighbors = 4)
        knn4.fit(x_train, y_train)
```

```
Out[9]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowsk
        i',
                     metric_params=None, n_jobs=None, n_neighbors=4, p=2,
                     weights='uniform')
```

```
In [10]: y_test_preds = knn4.predict(x_test)
```

```
In [11]: confusion_matrix(y_test,y_test_preds)
```

```
Out[11]: array([[83,  1],
                [16,  1]])
```

```
In [12]: Sensitivity4 = 1/(16 + 1)
         Sensitivity4
```

```
Out[12]: 0.058823529411764705
```

```
In [13]: specificity4 = 81/(81 + 3)
         specificity4
```

```
Out[13]: 0.9642857142857143
```

DECISION TREE

```
In [14]: tree = DecisionTreeClassifier(max_depth = 6)
         tree.fit(x_train, y_train)
```

```
Out[14]: DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=
        6,
                     max_features=None, max_leaf_nodes=None,
                     min_impurity_decrease=0.0, min_impurity_split=None,
                     min_samples_leaf=1, min_samples_split=2,
                     min_weight_fraction_leaf=0.0, presort=False, random_state=N
        one,
                     splitter='best')
```

```
In [15]: y_test_preds_tree = tree.predict(x_test)
         confusion_matrix(y_test, y_test_preds_tree)
```

```
Out[15]: array([[80,  4],
                [17,  0]])
```

```
In [16]: Sensitivity6 = 2/(15 + 2)
         Sensitivity6
```

```
Out[16]: 0.11764705882352941
```

```
In [17]:  Sensitivity6 = 1/(21 + 1)
          Sensitivity6
```

Out[17]: 0.045454545454545456