test

January 3, 2017

In [10]:
```
# Necssary libraries
import pandas as pd
import statsmodels.api as sm
from sklearn.cross_validation import KFold
from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier as RF
from sklearn.neighbors import KNeighborsClassifier as KNN
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, auc
from sklearn.utils import shuffle
from sklearn.metrics import roc_curve, auc
import pylab
from sklearn import svm
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier

import re
import pylab as plt
pd.set_option('display.max_columns', 500)
%matplotlib inline
from sklearn.linear_model import LinearRegression
import numpy.random as nprnd
import random
pd.set_option('display.max_columns', 500)
%matplotlib inline
```

In [22]:
```
size = 100000

userid = nprnd.randint(0,1000,size=size)

playlistid = nprnd.randint(1,1000,size=size)

trackid = nprnd.randint(1,10,size=size)
```

```
track_duration = nprnd.randint(1000000,3000000, size=size)

listening_duration = nprnd.randint(1000000,3000000, size=size)

#sellouts_total = nprnd.randint(1,10,size=size)

#location = nprnd.randint(1,100,size=size).astype(float)/10

#rooms_left = nprnd.randint(1,500,size=size)

#account_num = nprnd.randint(1,100000,size=size)
#date='2015-06-01'
```

In [17]:
```
import random
import time

def strTimeProp(start, end, format, prop):
    """Get a time at a proportion of a range of two formatted times.

    start and end should be strings specifying times formated in the
    given format (strftime-style), giving an interval [start, end].
    prop specifies how a proportion of the interval to be taken after
    start.  The returned time will be in the specified format.
    """

    stime = time.mktime(time.strptime(start, format))
    etime = time.mktime(time.strptime(end, format))

    ptime = stime + prop * (etime - stime)

    return time.strftime(format, time.localtime(ptime))


def randomDate(start, end, prop):
    return strTimeProp(start, end, '%m/%d/%Y %I:%M %p', prop)

random_dates = [randomDate("1/1/2008 1:30 PM", "1/1/2009 4:50 AM", random.
```

In [19]:
```
df_playlists = pd.DataFrame({'user_id':userid, 'playlist_id':playlistid})
```

In [26]:
```
df_playlists=df_playlists.drop_duplicates(['user_id','playlist_id'])
```

In [31]:
```
# The SQL command to execute this query would be:
sql_command = "select sum(playlist_id) from top_playlists group by playlis

#In python, this is equivalent to:
df_top_playlists=df_playlists.groupby('playlist_id').size()
df_top_playlists.sort()
df_top_playlists[::-1].head()
```

```
/anaconda/lib/python2.7/site-packages/ipykernel/__main__.py:6: FutureWarning: sort
```

```
Out[31]: playlist_id
         546    125
         969    122
         965    121
         321    121
         515    120
         dtype: int64
```

```
In [42]: # SQL:
         df_user_top_playlists= df_playlists.groupby(['user_id'])['playlist_id'].si
         df_user_top_playlists.sort()
         df_user_top_playlists[::-1]
```

```
/anaconda/lib/python2.7/site-packages/ipykernel/__main__.py:2: FutureWarning: sort
  from ipykernel import kernelapp as app
```

```
Out[42]: user_id
         957    123
         43     123
         919    123
         846    122
         760    121
         721    120
         415    119
         386    119
         144    118
         961    118
         901    118
         285    118
         825    118
         723    117
         894    116
         87     115
         425    115
         791    115
         702    115
         413    115
         23     115
         463    114
         893    114
         388    114
         329    113
         987    113
         671    113
         173    113
```

```
510    113
941    113
       ...
840     78
579     78
33      78
562     78
384     78
717     78
473     78
673     77
841     77
76      77
351     77
776     77
654     77
96      76
454     76
244     76
643     75
839     75
51      75
264     74
707     74
772     73
303     73
320     73
399     72
749     72
61      71
705     71
613     71
618     63
dtype: int64
```

1 Bonus thing

```
In [46]: df_listings = pd.DataFrame({'user_id':userid,'playlist_id':playlistid,'tra

In [53]: #df_listings
         df_user_tracks = pd.DataFrame(index=range(0,size), columns=range(0,10))

In [65]: for i in range(0,len(df_listings)):
             df_user_tracks.ix[i] = np.zeros(10)
             df_user_tracks.ix[i][df_listings.ix[i]['track_id']]=1
             #df_listings.ix[i]['track_id']

In [63]: df_user_tracks
```

Out[63]:

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	1	0	0	0	0	0	0
1	0	0	0	0	0	1	0	0	0	0
2	0	0	0	0	0	0	0	0	0	1
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
6	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
8	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
9	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
10	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
11	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
12	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
14	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
15	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
16	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
17	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
18	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
19	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
20	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
21	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
22	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
23	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
24	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
25	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
26	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
27	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
28	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
29	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...
99970	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
99971	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
99972	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
99973	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
99974	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
99975	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
99976	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
99977	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
99978	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
99979	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
99980	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
99981	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
99982	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
99983	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
99984	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
99985	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

```
99986  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN
99987  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN
99988  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN
99989  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN
99990  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN
99991  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN
99992  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN
99993  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN
99994  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN
99995  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN
99996  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN
99997  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN
99998  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN
99999  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN

[100000 rows x 10 columns]
```