In [1]:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression
import copy
```

```
c:\users\netta\appdata\local\programs\python\python35\lib\site-packages\sk
learn\ensemble\weight_boosting.py:29: DeprecationWarning: numpy.core.umath
_tests is an internal NumPy module and should not be imported. It will be
removed in a future NumPy release.
  from numpy.core.umath_tests import inner1d
```

In [2]:

```
data_set = pd.read_csv(r'top_hits_processed_dataset.csv').dropna()
data_set.sample(3)
```

Out[2]:

	track_id	Track Name	Artist	danceability	energy	key	loudness
9816	3VdDQkNCQKPAL4nZlFKWIN	Ghetto	Benash	0.715	0.703	10.0	-6.687
1941	0hYSal2BeLJ4LSpmKHVtPf	La vache	Sadek	0.870	0.634	11.0	-5.262
13239	4knL4iPxPOZjQzTUlELGSY	Rake It Up	Yo Gotti	0.910	0.444	1.0	-8.126

3 rows × 77 columns

we removed some categorical features such as time_signature and key.

In [3]:

```
features = data_set[['acousticness','danceability','energy','instrumentalness','livenes
s','mode','speechiness',
                     'valence','duration_norm','loudness_norm','tempo_norm']]
features.head()
```

Out[3]:

	acousticness	danceability	energy	instrumentalness	liveness	mode	speechiness	valen
0	0.1310	0.748	0.627	0.000000	0.0852	1.0	0.0644	0.5
1	0.6840	0.509	0.803	0.000539	0.4630	1.0	0.0400	0.6
2	0.7680	0.838	0.703	0.000002	0.0861	0.0	0.0500	0.6
3	0.8880	0.346	0.433	0.000000	0.1010	0.0	0.0650	0.7
4	0.0412	0.598	0.950	0.000000	0.6640	0.0	0.1000	0.6

similarly to the previous data, the regression model was no good...

In [4]:

```
for country in ['Estonia','Argentina','Belgium','Taiwan: Province of China','Ireland',
'Denmark','Austria']:
    print('training model on ' + country)
    labels = data_set[country]
    train_features, test_features, train_labels, test_labels = train_test_split(feature
s, labels, test_size = 0.1)
    linear_regression = LinearRegression()
    linear_regression.fit(train_features,train_labels)
    print(linear_regression.score(test_features,test_labels))
```

```
training model on Estonia
-0.00012459293701927443
training model on Argentina
0.02287973495569018
training model on Belgium
0.006225300559613123
training model on Taiwan: Province of China
0.04872724804881423
training model on Ireland
0.023228344415777927
training model on Denmark
0.02132603530167576
training model on Austria
0.0105431700318106
```

so we tried a random forest regressor:

Random Forest

In [5]:

```
for country in ['Estonia','Argentina','Belgium','Taiwan: Province of China','Ireland',
'Denmark','Austria']:
    print('training model on ' + country)
    labels = data_set[country]
    train_features, test_features, train_labels, test_labels = train_test_split(feature
s, labels, test_size = 0.1)
    random_forest = RandomForestRegressor(n_estimators=15,max_features ='sqrt')
    random_forest.fit(train_features,train_labels)
    print('score is: ' + str(random_forest.score(test_features,test_labels)))
```

```
training model on Estonia
score is: 0.03929896290203804
training model on Argentina
score is: 0.13846190786190293
training model on Belgium
score is: 0.047572786782645315
training model on Taiwan: Province of China
score is: 0.05229990504825244
training model on Ireland
score is: 0.03054209936805363
training model on Denmark
score is: 0.032615967866960305
training model on Austria
score is: 0.05283791979534136
```

the results are still very bad, even though they are slightly better than the result for the previous dataset.... maybe it has to do with the fact that the dataset has less noise