In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from collections import defaultdict
from PIL import Image
from scipy.misc import imread
### added myself
from random import *
###
%matplotlib inline
plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')
```

Simple K-Means

In this assignment, we will walk you through an implementation of the simple K-Means algorithm.

The K-means algorithm works as follows, assuming we have inputs $x_1, x_2, x_3, ..., x_n$ and value of K

- Step 1 - Pick K random points as cluster centers called centroids.
- Step 2 - Assign each x_i to nearest cluster by calculating its distance to each centroid.
- Step 3 - Find new cluster center by taking the average of the assigned points.
- Step 4 - Repeat Step 2 and 3 until none of the cluster assignments change.

Importing the data

In [2]:

```
data = pd.read_csv('xclara.csv')
print(data.shape)
data.head()
```

```
(3000, 2)
```

Out[2]:

	V1	V2
0	2.072345	-3.241693
1	17.936710	15.784810
2	1.083576	7.319176
3	11.120670	14.406780
4	23.711550	2.557729

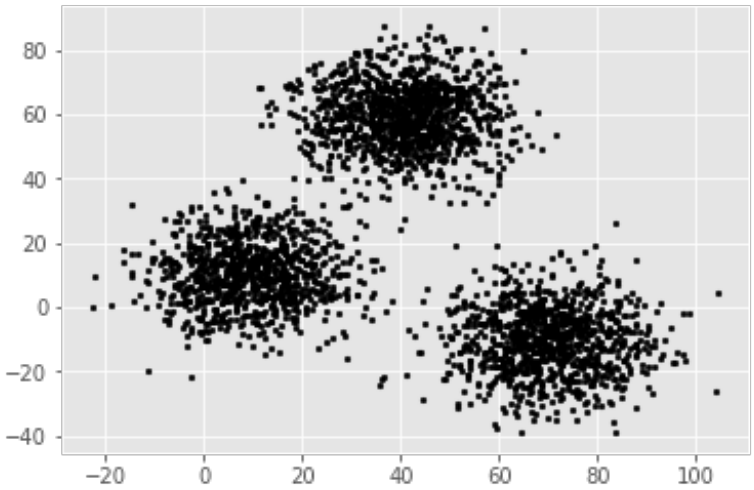
In [3]:

```
# Getting the values and plotting it
```

```
# Getting the values and plotting it
f1 = data['V1'].values
f2 = data['V2'].values
X = np.array(list(zip(f1, f2)))
plt.scatter(f1, f2, c='black', s=7)
```

Out[3]:

```
<matplotlib.collections.PathCollection at 0x10dc9f2b0>
```



In [4]:

```
#Number of clusters
k = 3
colors = ['red', 'green', 'blue']
```

Step 1: Initialize k random points as centroids

Hint: use the function np.random.randint

In [5]:

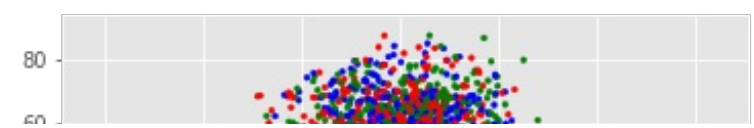
```
#TODO:
indices = sample(list(np.arange(X.shape[0])),  3)
Cs = [X[i] for i in indices]
# X coordinates of random centroids
C_x = [C[0] for C in Cs]
# Y coordinates of random centroids
C_y = [C[1] for C in Cs]
# Zip the arrays C_x and C_y into a list of tuples (C_x, C_y)
C = np.array(list(zip(C_x,C_y)), dtype=np.float32)
```

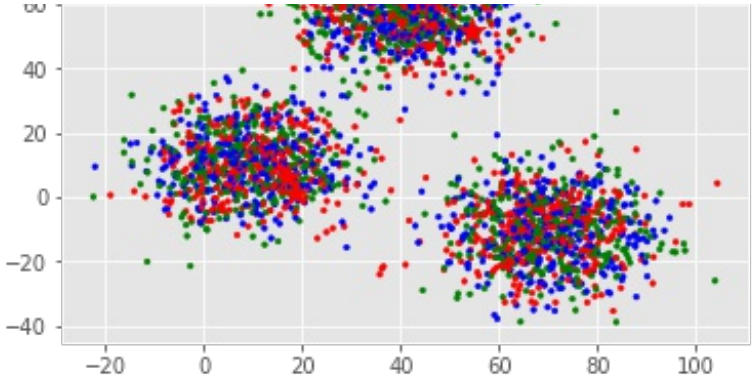
In [6]:

```
# Plotting along with the Centroids
plt.scatter(f1, f2, c=colors, s=7)
plt.scatter(C_x, C_y, marker='*', s=200, c='red')
```

Out[6]:

```
<matplotlib.collections.PathCollection at 0x10dd9c668>
```





Step 2: Assign each x_i to nearest cluster by calculating its distance to each centroid.

In [7]:

```
# TODO: Euclidean Distance Caculator
# Hint: use np.linalg.norm
def dist(a, b, ax=1):
    return np.linalg.norm(a-b,ax)
```

In [8]:

```
from copy import deepcopy

# To store the value of centroids when it updates
C_old = np.zeros(C.shape)
# Cluster Labels(0, 1, 2)
clusters = np.zeros(len(X))

# TODO: Error func. - Distance between new centroids and old centroids
# Hint: use the function dist()
error = sum([dist(a,b) for a in C for b in C_old])
```

In [9]:

```
# Loop will run till the error becomes zero
while error != 0:
    # Assigning each value to its closest cluster
    for i in range(len(X)):
        #TODO: Compute distances between each point and the centroid and
        #   assign it to the cluster of the closest centroid
        distances = np.array([dist(C[z],X[i]) for z in range(len(C))])
        cluster = np.argmin(distances)
        clusters[i] = cluster
    # Storing the old centroid values
    C_old = deepcopy(C)
    # Finding the new centroids by taking the average value
    for i in range(k):
        points = np.array([X[j] for j in range(len(X)) if clusters[j]==i])
        #TODO: compute average
        C[i] = points.mean(axis=0)
    error = 0 #REDO
```

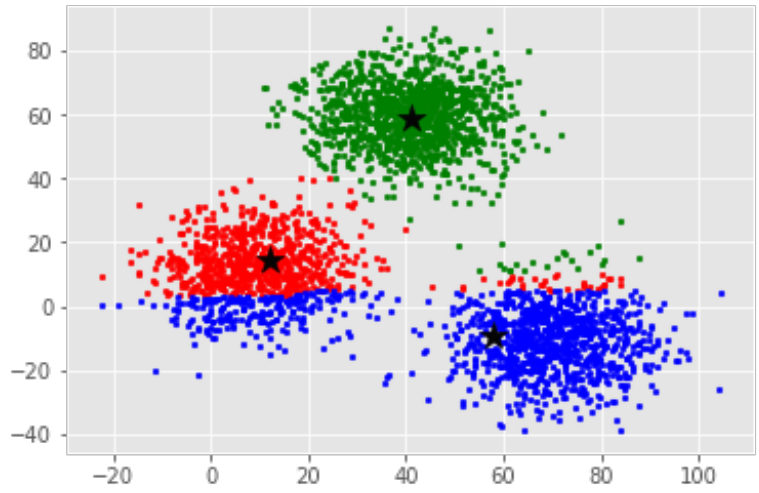
In [10]:

```
fig, ax = plt.subplots()
```

```
for i in range(k):
        points = np.array([X[j] for j in range(len(X)) if clusters[j] == i])
        ax.scatter(points[:, 0], points[:, 1], s=7, c=colors[i])
ax.scatter(C[:, 0], C[:, 1], marker='*', s=200, c='#050505')
```

Out[10]:

```
<matplotlib.collections.PathCollection at 0x1813ebbbe0>
```



K-Means Clustering on audio

This part walks you through a simple application of clustering on audio processing. The goal is to cluster the onsets of a music track in the time domain. Onset is the beginning of a musical note which is characterized by increases in spectral energy, changes in spectral energy distribution/phase, changes in detected pitch or even spectral patterns recognizable by machine learning techniques such as neural networks.

In [11]:

```
# import sys
# !{sys.executable} -m pip install librosa
# !{sys.executable} -m pip install mir_eval
!pip install librosa
!pip install mir_eval
#NOTE: RESTART NOTEBOOK FROM TERMINAL IF ERROR OCCURS

import numpy, scipy, matplotlib.pyplot as plt, sklearn, librosa, mir_eval,
IPython.display, urllib
plt.rcParams['figure.figsize'] = (14, 4)
```

```
Requirement already satisfied: librosa in /anaconda3/lib/python3.6/site-pac
kages
Requirement already satisfied: decorator>=3.0.0 in
/anaconda3/lib/python3.6/site-packages (from librosa)
Requirement already satisfied: six>=1.3 in /anaconda3/lib/python3.6/site-pa
ckages (from librosa)
Requirement already satisfied: scipy>=0.14.0 in
/anaconda3/lib/python3.6/site-packages (from librosa)
Requirement already satisfied: joblib>=0.7.0 in
/anaconda3/lib/python3.6/site-packages (from librosa)
Requirement already satisfied: resampy>=0.2.0 in
/anaconda3/lib/python3.6/site-packages (from librosa)
Requirement already satisfied: audioread>=2.0.0 in
```

```
/anaconda3/lib/python3.6/site-packages (from librosa)
Requirement already satisfied: numpy>=1.8.0 in
/anaconda3/lib/python3.6/site-packages (from librosa)
Requirement already satisfied: scikit-learn!=0.19.0,>=0.14.0 in
/anaconda3/lib/python3.6/site-packages (from librosa)
Requirement already satisfied: numba>=0.32 in
/anaconda3/lib/python3.6/site-packages (from resampy>=0.2.0->librosa)
Requirement already satisfied: llvmlite>=0.22.0.dev0 in
/anaconda3/lib/python3.6/site-packages (from numba>=0.32->resampy>=0.2.0->l
ibrosa)
You are using pip version 9.0.1, however version 9.0.3 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
Requirement already satisfied: mir_eval in /anaconda3/lib/python3.6/site-pa
ckages
Requirement already satisfied: scipy>=0.9.0 in
/anaconda3/lib/python3.6/site-packages (from mir_eval)
Requirement already satisfied: numpy>=1.7.0 in
/anaconda3/lib/python3.6/site-packages (from mir_eval)
Requirement already satisfied: future in /anaconda3/lib/python3.6/site-pack
ages (from mir_eval)
Requirement already satisfied: six in /anaconda3/lib/python3.6/site-package
s (from mir_eval)
You are using pip version 9.0.1, however version 9.0.3 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
```

In [12]:

```
filename = 'audio.mp3'
IPython.display.Audio(filename)
```

Out[12]:

Your browser does not support the audio element.

Load the audio file into an array

In [13]:

```
x, fs = librosa.load(filename)
print(fs)
```

```
22050
```

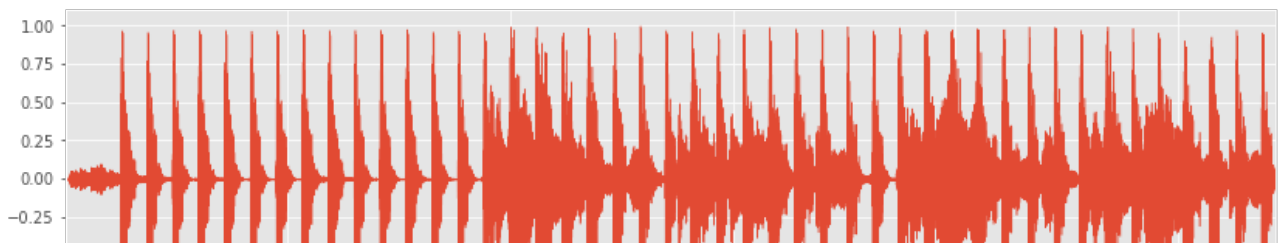
Plot audio signal

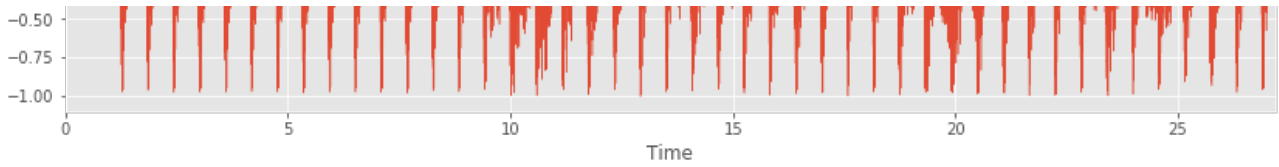
In [14]:

```
import librosa.display
librosa.display.waveplot(x, fs)
```

Out[14]:

```
<matplotlib.collections.PolyCollection at 0x1c250cc7b8>
```





Onset Detection

In [15]:

```
onset_frames = librosa.onset.onset_detect(x, sr=fs, delta=0.04, wait=4)
onset_times = librosa.frames_to_time(onset_frames, sr=fs)
onset_samples = librosa.frames_to_samples(onset_frames)
```

The detected onsets are marked as 'beeps'

In [16]:

```
x_with_beeps = mir_eval.sonify.clicks(onset_times, fs, length=len(x))
IPython.display.Audio(x + x_with_beeps, rate=fs)
```

Out[16]:

Your browser does not support the audio element.

Feature Extraction

Compute the zero crossing rate and energy for each detected onset

In [17]:

```
def extract_features(x, fs):
    zcr = librosa.zero_crossings(x).sum()
    energy = scipy.linalg.norm(x)
    return [zcr, energy]
frame_sz = int(fs*0.090)
features = numpy.array([extract_features(x[i:i+frame_sz], fs) for i in
onset_samples])
print(features.shape)
```

```
(71, 2)
```

Scale the features from -1 to 1

In [18]:

```
min_max_scaler = sklearn.preprocessing.MinMaxScaler(feature_range=(-1, 1))
features_scaled = min_max_scaler.fit_transform(features)
print(features_scaled.shape)
print(features_scaled.min(axis=0))
print(features_scaled.max(axis=0))
```

```
(71, 2)
[-1. -1.]
[1. 1.]
```

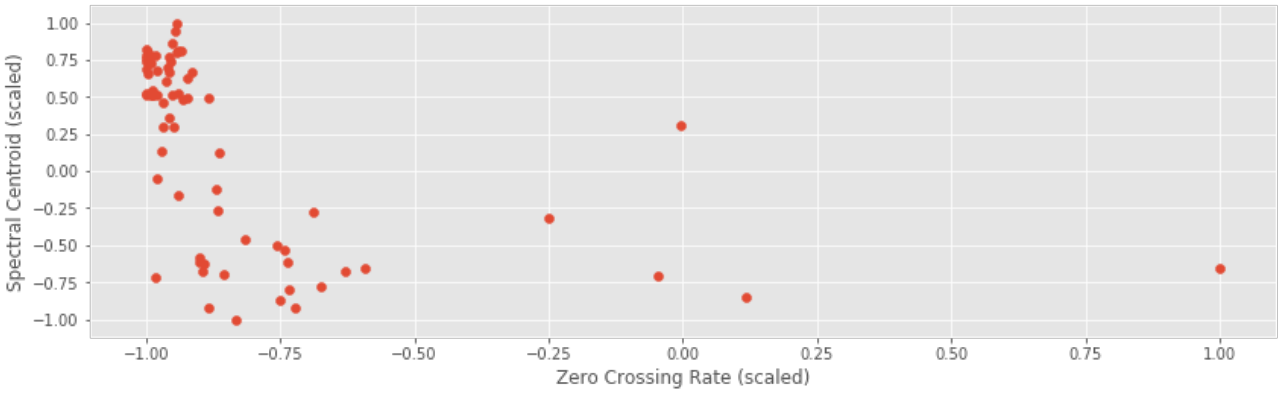
Plot the scaled spectral centroid against the scaled zero crossing rate

In [19]:

```
plt.scatter(features_scaled[:,0], features_scaled[:,1])
plt.xlabel('Zero Crossing Rate (scaled)')
plt.ylabel('Spectral Centroid (scaled)')
```

Out[19]:

```
Text(0,0.5,'Spectral Centroid (scaled)')
```



K-means clustering on onset data

Use K-means to group the onset data into 2 clusters.

In [20]:

```
#Hint: use sklearn.cluster.KMeans, fit_predict
model = sklearn.cluster.KMeans(n_clusters=2)
labels = model.fit_predict(features_scaled)
print(labels)
```

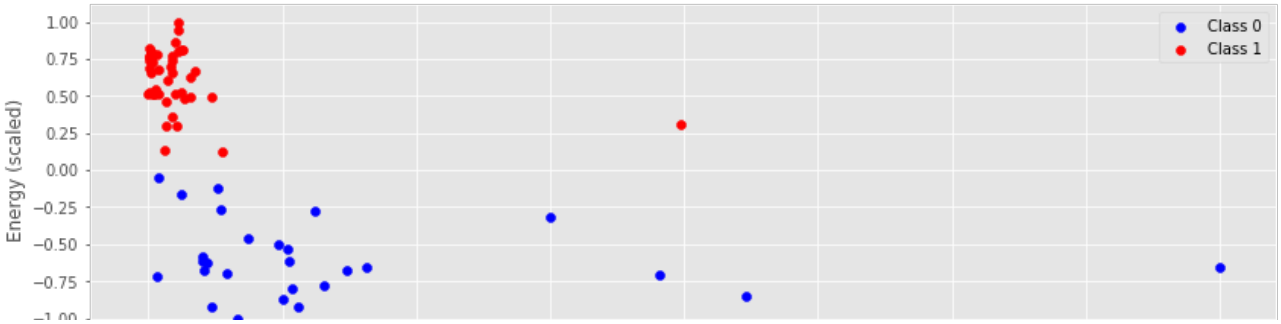
```
[0 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 0 1 0 1 0 1 0 0 1 1 0 1 1 0 1 0
 1 0 1 0 0 0 0 0 1 1 1 1 1 0 1 0 1 0 1 0 1 1 1 1 1 1 0 0 1 0 0 1 0 1]
```

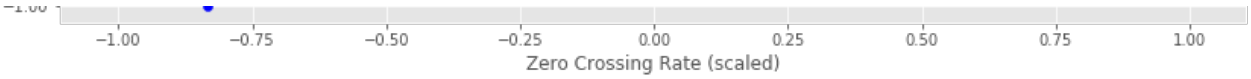
In [21]:

```
plt.scatter(features_scaled[labels==0,0], features_scaled[labels==0,1], c='
b')
plt.scatter(features_scaled[labels==1,0], features_scaled[labels==1,1], c='
r')
plt.xlabel('Zero Crossing Rate (scaled)')
plt.ylabel('Energy (scaled)')
plt.legend(('Class 0', 'Class 1'))
```

Out[21]:

```
<matplotlib.legend.Legend at 0x1c28845940>
```





Onsets assigned to Class 0

In [22]:

```
x_with_beeps = mir_eval.sonify.clicks(onset_times[labels==0], fs, length=le
n(x))
IPython.display.Audio(x + x_with_beeps, rate=fs)
```

Out[22]:

Your browser does not support the audio element.

Onsets assigned to Class 1

In [23]:

```
x_with_beeps = mir_eval.sonify.clicks(onset_times[labels==1], fs, length=le
n(x))
IPython.display.Audio(x + x_with_beeps, rate=fs)
```

Out[23]:

Your browser does not support the audio element.

Affinity Propagation

Try clustering with other clustering algorithms in scikit-learn such as affinity propagation which can cluster without defining the number of clusters beforehand

In [24]:

```
model = sklearn.cluster.AffinityPropagation()
labels = model.fit_predict(features_scaled)
print(labels)
```

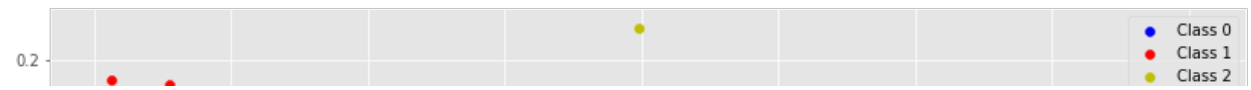
```
[3 3 4 4 4 4 4 4 4 4 5 3 5 5 4 4 4 4 1 4 4 1 5 0 1 3 5 6 3 5 5 1 4 5 1 5 3
 2 3 4 3 3 1 3 3 5 5 4 4 5 3 4 3 4 3 4 3 4 1 4 5 5 5 3 6 5 6 3 5 3 5]
```

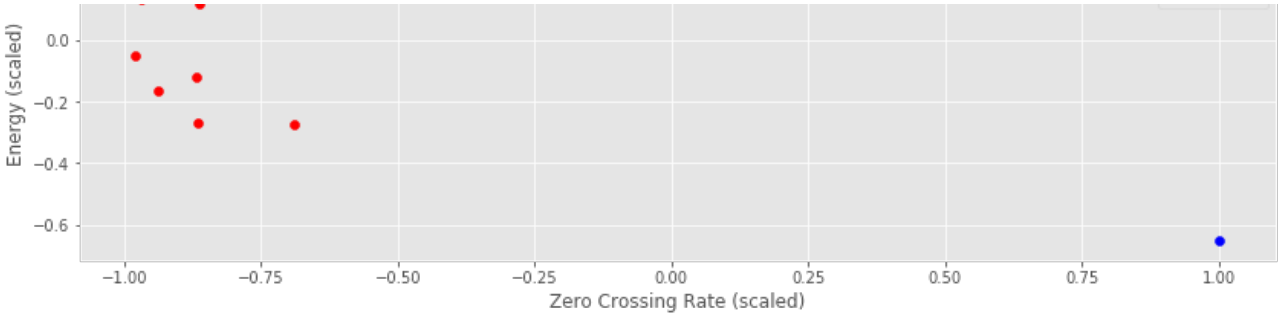
In [25]:

```
plt.scatter(features_scaled[labels==0,0], features_scaled[labels==0,1], c='
b')
plt.scatter(features_scaled[labels==1,0], features_scaled[labels==1,1], c='
r')
plt.scatter(features_scaled[labels==2,0], features_scaled[labels==2,1], c='
y')
plt.xlabel('Zero Crossing Rate (scaled)')
plt.ylabel('Energy (scaled)')
plt.legend(('Class 0', 'Class 1', 'Class 2'))
```

Out[25]:

```
<matplotlib.legend.Legend at 0x1c2569ba20>
```





Beeps are played for each onset assigned to cluster 0

In [26]:

```
x_with_beeps = mir_eval.sonify.clicks(onset_times[labels==0], fs, length=le
n(x))
IPython.display.Audio(x + x_with_beeps, rate=fs)
```

Out[26]:

Your browser does not support the audio element.

Beeps are played for each onset assigned to cluster 1

In [27]:

```
x_with_beeps = mir_eval.sonify.clicks(onset_times[labels==1], fs, length=le
n(x))
IPython.display.Audio(x + x_with_beeps, rate=fs)
```

Out[27]:

Your browser does not support the audio element.

Beeps are played for each onset assigned to cluster 2

In [28]:

```
x_with_beeps = mir_eval.sonify.clicks(onset_times[labels==2], fs, length=le
n(x))
IPython.display.Audio(x + x_with_beeps, rate=fs)
```

Out[28]:

Your browser does not support the audio element.