Music Genre Classification using K Means

In [2]:

```
#importing useful modules of python
import numpy as np
import pandas as pd
import scipy as sp
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.metrics import confusion_matrix
```

In [3]:

```
#loading the data from .csv file
dataframe=pd.read_csv('Genre_data.csv')
print("Coloumns of the data file....\n")

for col in dataframe.columns:
    print(col)
```

```
Coloumns of the data file....

filename
length
chroma_stft_mean
chroma_stft_var
rms_mean
rms_var
spectral_centroid_mean
spectral_centroid_var
spectral_bandwidth_mean
spectral_bandwidth_var
rolloff_mean
rolloff_var
zero_crossing_rate_mean
zero_crossing_rate_var
harmony_mean
harmony_var
perceptr_mean
perceptr_var
tempo
mfcc1_mean
mfcc1_var
mfcc2_mean
mfcc2_var
mfcc3_mean
mfcc3_var
mfcc4_mean
mfcc4_var
mfcc5_mean
mfcc5_var
mfcc6_mean
mfcc6_var
mfcc7_mean
mfcc7_var
mfcc8_mean
mfcc8_var
mfcc9_mean
mfcc9_var
mfcc10_mean
mfcc10_var
mfcc11_mean
mfcc11_var
mfcc12_mean
mfcc12_var
mfcc13_mean
mfcc13_var
mfcc14_mean
mfcc14_var
mfcc15_mean
mfcc15_var
mfcc16_mean
mfcc16_var
mfcc17_mean
mfcc17_var
mfcc18_mean
mfcc18_var
mfcc19_mean
mfcc19_var
mfcc20_mean
mfcc20_var
label
```

In [4]:

```
#dictionary which maps lable name to a positive integer
genre_to_number= {
    'blues':0,
    'classical':1,
    'country':2,
    'disco':3,
    'hiphop':4,
    'jazz':5,
    'metal':6,
    'pop':7,
    'reggae':8,
    'rock':9,
}
```

In [5]:

```
#removing unnecessary columns
dataframe.label=[genre_to_number[item] for item in dataframe.label]
labels=dataframe['label']
dataframe=dataframe.drop(['filename', 'length', 'label'], axis = 1)
```

In [6]:

```
#Converting into numpy array
X=dataframe.to_numpy()
y=labels.to_numpy()
```

In [7]:

```
#Verifying the shape
print(X.shape)
print(y.shape)
```

```
(9990, 57)
(9990,)
```

In [8]:

```
print("Number of datapoints: ",X.shape[0])
print("Number of features: ",X.shape[1])
```

```
Number of datapoints:  9990
Number of features:  57
```

In [9]:

```
#Performing the K-means clustering
km = KMeans(
    n_clusters=10, init='random',
    n_init=200, max_iter=500,
    tol=1e-04, random_state=0
)
y_pred = km.fit_predict(X)
```

In [11]:

```
#Calculating the density of each cluster
freq_denominator=np.zeros(10)
for i in range(9990):
    freq_denominator[y_pred[i]]+=1
print("ith element represents the number of data points belonging to cluster i")
print(freq_denominator)
```

```
ith element represents the number of data points belonging to cluster i
[ 269. 2343. 1674.  498.   71. 1255. 2538.  821.  186.  335.]
```

Observation: Here we can see that each cluster contains different amount of data points. So K-means is not able to form appropriate clusters.

In [13]:

```
#giving proper lables
m=y_pred.shape[0]
cluster_to_label=np.zeros(10)
freq=np.zeros([10,10])
for i in range(m):
    freq[y_pred[i]][y[i]]+=1
for i in range(10):
    cluster_to_label[i]=np.argmax(freq[i])
for i in range(m):
    y_pred[i]=cluster_to_label[y_pred[i]]
```

In [14]:

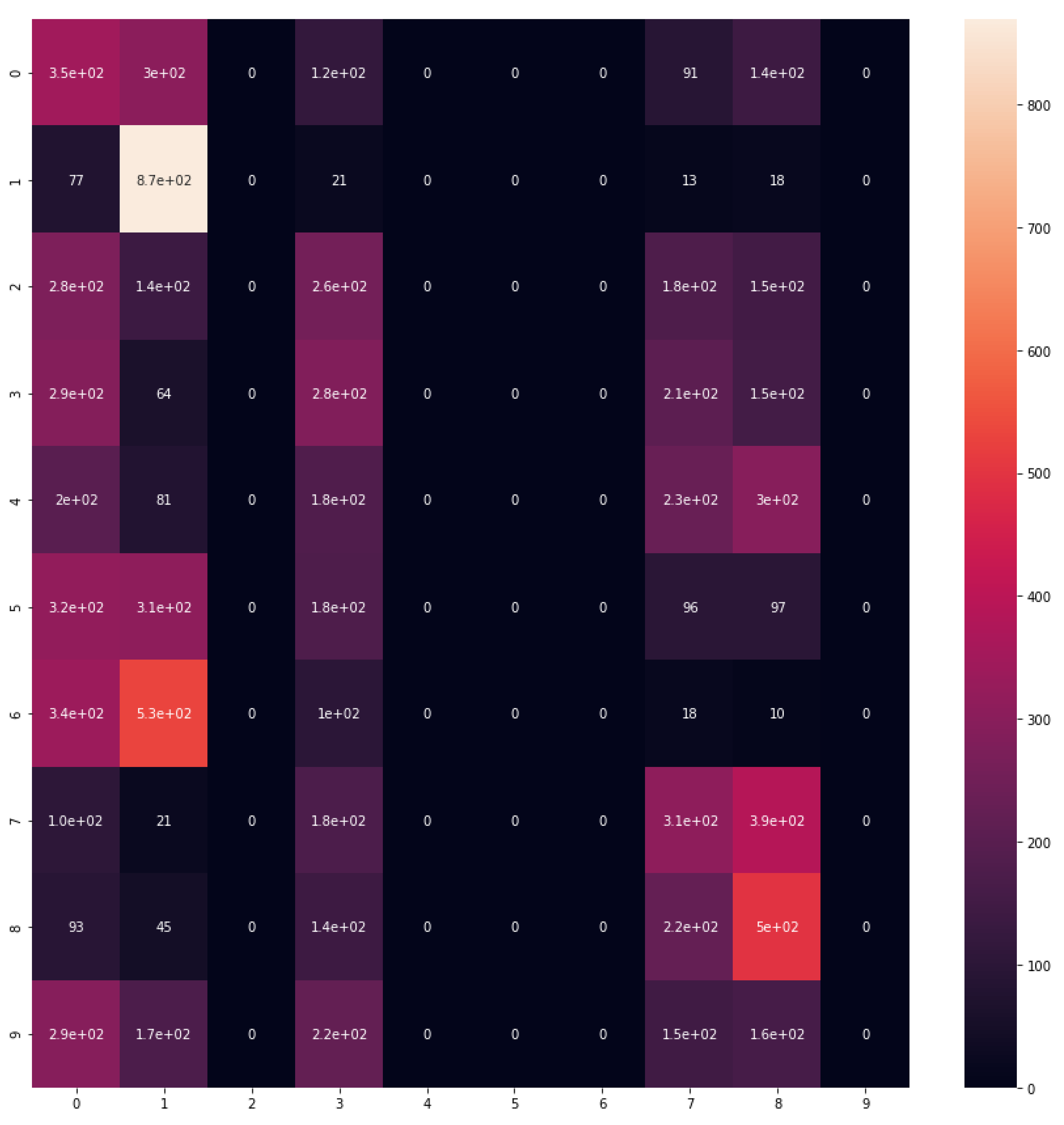
```
print("(i,j) element represents the number of datapoints belonging to cluster i and having j as original label: ")
print(freq)
print("ith element represents which cluster represents which label: ")
print(cluster_to_label)
```

```
(i,j) element represents the number of datapoints belonging to cluster i and having j as original label:
[[  2.   0.  11.  13.  59.   1.   2. 113.  53.  15.]
 [349.  77. 277. 289. 203. 318. 339. 105.  93. 293.]
 [116.  21. 255. 282. 183. 177. 101. 175. 140. 224.]
 [ 44.   6.  43.  39.  71.  15.   2.  99. 148.  31.]
 [  0.   1.   0.   4.  14.   2.   0.  17.  19.  14.]
 [ 89.  13. 168. 197. 174.  95.  16. 199. 172. 132.]
 [304. 869. 136.  64.  81. 312. 532.  21.  45. 174.]
 [ 70.   9.  85.  80.  98.  66.   7. 139. 183.  84.]
 [  7.   2.   3.   9.  50.   6.   0.  41.  55.  13.]
 [ 19.   0.  19.  22.  65.   8.   1.  91.  92.  18.]]
ith element represents which cluster represents which label:
[7. 0. 3. 8. 8. 7. 1. 8. 8. 8.]
```

In [15]:

```
#plotting confusion matrix
print("Confusion matrix is visualized below. Where X axis has ground truth and Y axis has predicted values.\n\n\n")
plt.figure(figsize=(15,15))
sns.heatmap(confusion_matrix(y,y_pred),annot=True)
plt.show()
```

```
Confusion matrix is visualized below. Where X axis has ground truth and Y axis has predicted values.
```



Observation: On a final note K-means is not sufficient to classify the music genre and we need to come up with another ML model.