

電子商務技術\_HW4  
108403201 資管三 A 黃名揚

一. 請用 python 實作以下問題

1. 載入資料並刪除除了"energy", "speechiness", "acousticness", "instrumentalness", "loudness", "tempo", "danceability", "valence", "liveness" 以外之欄位(使用 pandas dataframe)

```
In [1]: """
1. 載入資料並刪除除了"energy", "speechiness", "acousticness",
"instrumentalness", "loudness", "tempo", "danceability", "valence", "liveness"
以外之欄位(使用 pandas dataframe)
"""

import pandas as pd
import numpy as np
from itertools import combinations
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.preprocessing import StandardScaler

data = pd.read_csv("wu_songs.csv")
wu_song = pd.DataFrame(data)
wu_song = wu_song.drop(['time_signature', 'key', 'duration_ms', 'mode', 'type', 'uri'], axis=1)
wu_song.head()
```

```
Out[1]:
```

|   | energy | liveness | tempo   | speechiness | acousticness | instrumentalness | danceability | loudness | valence |
|---|--------|----------|---------|-------------|--------------|------------------|--------------|----------|---------|
| 0 | 0.979  | 0.2720   | 128.876 | 0.1220      | 0.007620     | 0.015200         | 0.410        | -3.481   | 0.080   |
| 1 | 0.861  | 0.0747   | 100.080 | 0.0493      | 0.001890     | 0.000539         | 0.518        | -6.998   | 0.317   |
| 2 | 0.963  | 0.2030   | 195.979 | 0.1590      | 0.000090     | 0.000304         | 0.356        | -4.385   | 0.379   |
| 3 | 0.968  | 0.1060   | 158.089 | 0.1370      | 0.000047     | 0.000006         | 0.365        | -4.267   | 0.386   |
| 4 | 0.327  | 0.0922   | 75.122  | 0.0489      | 0.605000     | 0.000012         | 0.434        | -10.161  | 0.260   |

2. 將剩下的欄位做特徵篩選的動作，並使用 kmeans silhouette analysis 的方法找出在哪三個欄位的情況下(需考慮所有組合)，分 X 群會有最高的 silhouette score。請找出 X 與 silhouette score 還有是哪三個欄位。(20%)  
請解釋 silhouette 分析法 與 elbow 轉折判斷法的差別(3%)

```
In [2]: """
2. 將剩下的欄位做特徵篩選的動作，並使用 kmeans silhouette analysis 的方法
找出在哪三個欄位的情況下(需考慮所有組合)，分 X 群會有最高的 silhouette
score。請找出 X 與 silhouette score 還有是哪三個欄位。(20%)
請解釋 silhouette 分析法 與 elbow 轉折判斷法的差別(3%)

"""

#將剩下的欄位做特徵篩選的動作，找出三個欄位的所有組合
wu_song_feature=list(combinations(wu_song.columns,3))
wu_song_feature = np.array(wu_song_feature)

#使用陣列儲存
silhouette_avg = []
silhouette_max_bycolumn = []
z = 0

#使用StandardScaler在每次fit時將資料標準化
#random_state=15, x 範圍落在2~12
for idx, val in enumerate(wu_song_feature):
    wu_song_test=wu_song[val]
    scaler = StandardScaler()
    scaler.fit(wu_song_test)
    wu_song_scaled = scaler.transform(wu_song_test)
    silhouette_avg.append([])

    for i in range(2,12):
        kmeans_fit = KMeans(n_clusters = i,random_state=15).fit(wu_song_scaled)
        silhouette_avg[z].append(silhouette_score(wu_song_scaled, kmeans_fit.labels_))

    silhouette_max_bycolumn.append(max(silhouette_avg[z]))
    z+=1
```

```
In [3]: #三個欄位·X·silhouette score執行結果

maxindex=silhouette_max_bycolumn.index(max(silhouette_max_bycolumn))
conclusion = {
    "分群數目":["X=2","X=3","X=4","X=5","X=6","X=7","X=8","X=9","X=10","X=11"],
    "silhouette score":silhouette_avg[maxindex]
}
df = pd.DataFrame(conclusion)

print("欄位:", wu_song_feature[maxindex])
print("X=2")
print("silhouette score=", max(silhouette_avg[maxindex]))
print("-----")
print(df)
```

```
欄位: ['energy' 'acousticness' 'loudness']
X=2
silhouette score= 0.6322586386208744
-----
分群數目 silhouette score
0 X=2 0.632259
1 X=3 0.497777
2 X=4 0.484137
3 X=5 0.392655
4 X=6 0.394427
5 X=7 0.352258
6 X=8 0.347309
7 X=9 0.328941
8 X=10 0.328518
9 X=11 0.327294
```

(欄位:energy、acousticness、loudness)

(X=2 silhouette score=0.6322586386208744)

**silhouette 分析法:** 衡量物件和所屬群之間的相似度。Silhouette 值接近 1，說明物件與所屬群之間有密切聯繫；反之則接近-1

**elbow 轉折判斷法:** 適用於 K 值相對較小的情況。選擇的 k 值小於真正的時，k 每增加 1，cost 值就會大幅減小；選擇的 k 值大於真正的 K 時，k 每增加 1，cost 值的變化不明顯。正確的 k 值會在轉捩點上。

3. 使用剛剛找出來的欄位用 k-means 做分群。超參數設定為 n\_cluster=4 , random\_state=15 。 並使用 plotly 繪製出 3d 圖形如以下所示(15%)： 注意要有欄位名稱，也就是剛剛找出來的那三個

```
In [4]: #3. 使用剛剛找出來的欄位用 k-means 做分群。 n_cluster=4 ,random_state=15 。
#使用 plotly 繪製出 3d 圖形如以下所示

import plotly.express as px

wu_song_new=wu_song[wu_song_feature[maxindex]]

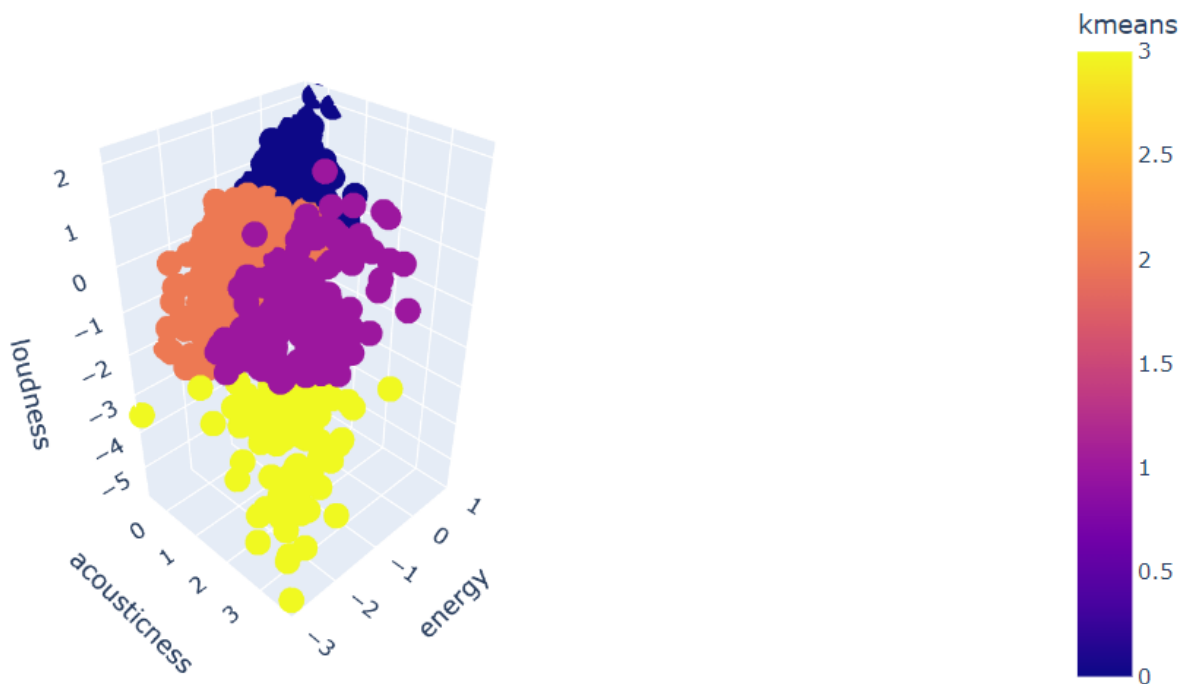
#資料標準化
scaler = StandardScaler()
scaler.fit(wu_song_new)
wu_song_scaled = scaler.transform(wu_song_new)

#利用kmeans分群
kmeans = KMeans(n_clusters = 4,random_state=15).fit(wu_song_scaled)

#將資料轉為dataframe格式
wu_song_scaled_df=pd.DataFrame(wu_song_scaled)
wu_song_scaled_df['kmeans'] = kmeans.labels_
wu_song_scaled_df.columns = ['energy' , 'acousticness' , 'loudness','kmeans']

#畫出3D圖形
fig = px.scatter_3d(wu_song_scaled_df, x='energy', y='acousticness', z='loudness',
                    color='kmeans')

fig.update_layout(
    scene = dict(
        xaxis = dict(nticks=5,range=[1,-3])
    )
)
fig.show()
```



4. 使用剛剛找出來的欄位用 **Meanshift** 做分群(15%) 請找出最佳的 **estimate\_bandwidth**.超參數設定為 **random\_state=15, quantile=0.32, n\_samples=1000** 使用剛剛找出的 **estimate\_bandwidth** 做分群並繪製如第三題的圖

```
In [5]: """
4. 使用剛剛找出來的欄位用 Meanshift 做分群
請找出最佳的 estimate_bandwidth. random_state=15,
quantile=0.32, n_samples=1000
"""

from sklearn.cluster import MeanShift, estimate_bandwidth
wu_song_new1=wu_song[wu_song_feature[maxindex]]

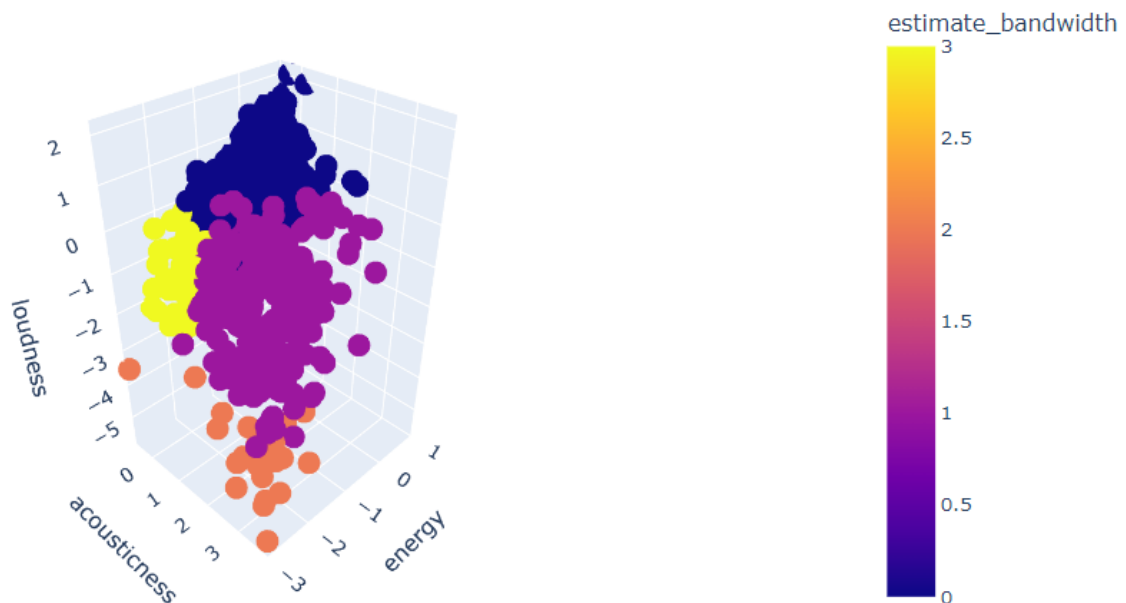
#資料標準化
scaler = StandardScaler()
scaler.fit(wu_song_new1)
wu_song_scaled = scaler.transform(wu_song_new1)

#使用Meanshift分群
bandwidth = estimate_bandwidth(wu_song_scaled,random_state=15,quantile=0.32, n_samples=1000)
ms = MeanShift(bandwidth=bandwidth, bin_seeding=True)
ms = ms.fit(wu_song_scaled)
labels = ms.labels_
cluster_centers = ms.cluster_centers_

#將資料轉為dataframe格式
wu_song_scaled_df1=pd.DataFrame(wu_song_scaled)
wu_song_scaled_df1['estimate_bandwidth'] = ms.labels_
wu_song_scaled_df1.columns = ['energy', 'acousticness', 'loudness', 'estimate_bandwidth']

#畫出3D圖形
fig = px.scatter_3d(wu_song_scaled_df1, x='energy', y='acousticness', z='loudness',
                    color='estimate_bandwidth')

fig.update_layout(
    scene = dict(
        xaxis = dict(nticks=5,range=[1,-3])
    )
)
fig.show()
```



5. 使用剛剛找出來的欄位用 k-prototypes 做分群。超參數設定為 n\_cluster=4 random\_state=15,init='Huang',verbose=0。並使用 plotly 繪製出 3d 圖形如第三題的圖

```
In [6]: """
5. 使用剛剛找出來的欄位用 k-prototypes 做分群。 n_cluster=4
random_state=15,init='Huang',verbose=0。
並使用 plotly 繪製出 3d 圖形
"""
from kmodes.kprototypes import KPrototypes

#資料標準化
scaler = StandardScaler()
scaler.fit(wu_song)
wu_song_scaled = scaler.transform(wu_song)

#將資料轉為dataframe
wu_song_scaled_df2=pd.DataFrame(wu_song_scaled)
wu_song_scaled_df2.columns = ['energy', 'liveness', 'tempo', 'speechiness', 'acousticness', 'instrumentalness',
                              'danceability', 'loudness', 'valence']

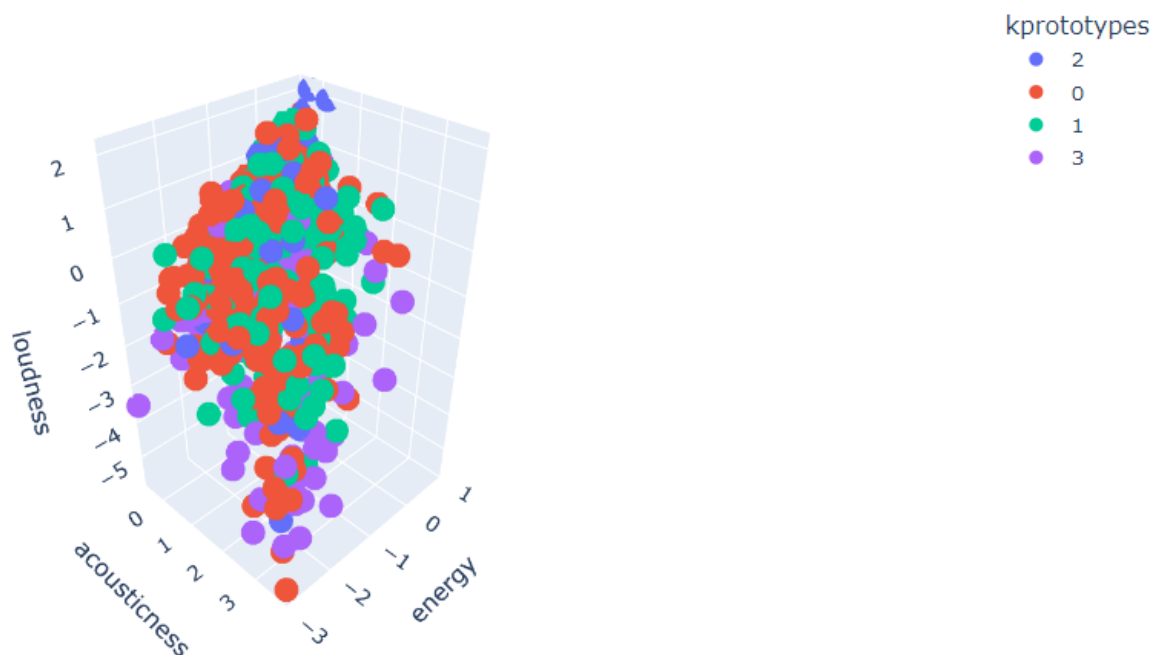
#使用KPrototypes分群
kproto = KPrototypes(n_clusters= 4,random_state=15, init='Huang', verbose=0)
kproto.fit_predict(wu_song_scaled_df2,categorical=[0,4,7])

wu_song_scaled_df2['kprototypes'] = kproto.labels_
wu_song_scaled_df2=wu_song_scaled_df2[['energy', 'acousticness', 'loudness', 'kprototypes']]

#畫出3D圖形
fig = px.scatter_3d(wu_song_scaled_df2, x='energy', y='acousticness', z='loudness',
                    color='kprototypes')

fig.update_layout(
    scene = dict(
        xaxis = dict(nticks=5,range=[1,-3])
    )
)

fig.show()
```



6. 使用剛剛找出來的欄位用 k-modes 做分群。超參數設定為 `n_cluster=4` `random_state=15,init='Huang',verbose=0`。並使用 `plotly` 繪製出 3d 圖形如第三題的圖

```
In [7]: """
6. 使用剛剛找出來的欄位用 k-modes 做分群。 n_cluster=4
random_state=15,init='Huang',verbose=0。
並使用 plotly 繪製出 3d 圖形如第三題的圖
"""

from kmodes import kmodes
wu_song_new3=wu_song[wu_song_feature[maxindex]]

#資料標準化
scaler = StandardScaler()
scaler.fit(wu_song_new3)
wu_song_scaled = scaler.transform(wu_song_new3)

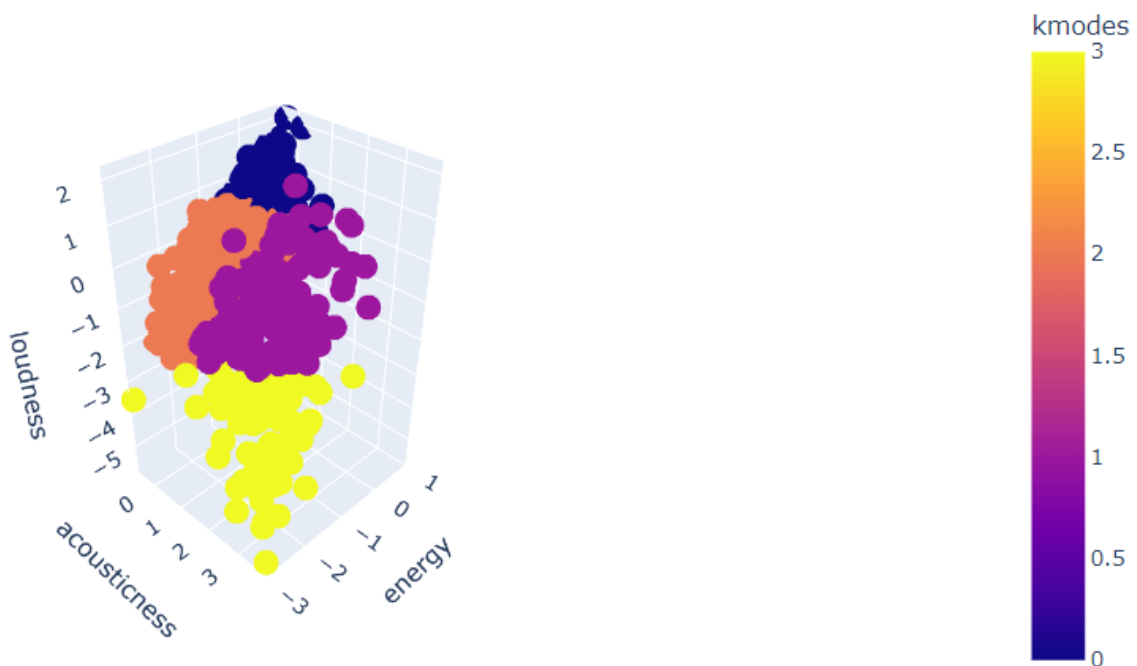
#使用kmodes分群
kmode = kmodes.KModes(n_clusters=4, random_state=15,init='Huang', verbose=0)
kmode.fit_predict(wu_song_scaled)

#將資料轉為dataframe格式
wu_song_scaled_df3=pd.DataFrame(wu_song_scaled)
wu_song_scaled_df3['kmodes'] = kmode.labels_
wu_song_scaled_df3.columns = ['energy', 'acousticness', 'loudness', 'kmodes']

#畫出3D圖形
fig = px.scatter_3d(wu_song_scaled_df3, x='energy', y='acousticness', z='loudness',
                    color='kmodes')

fig.update_layout(
    scene = dict(
        xaxis = dict(nticks=5,range=[1,-3])
    )
)

fig.show()
```



## 7. 請比較說明上述四種分群法的差異

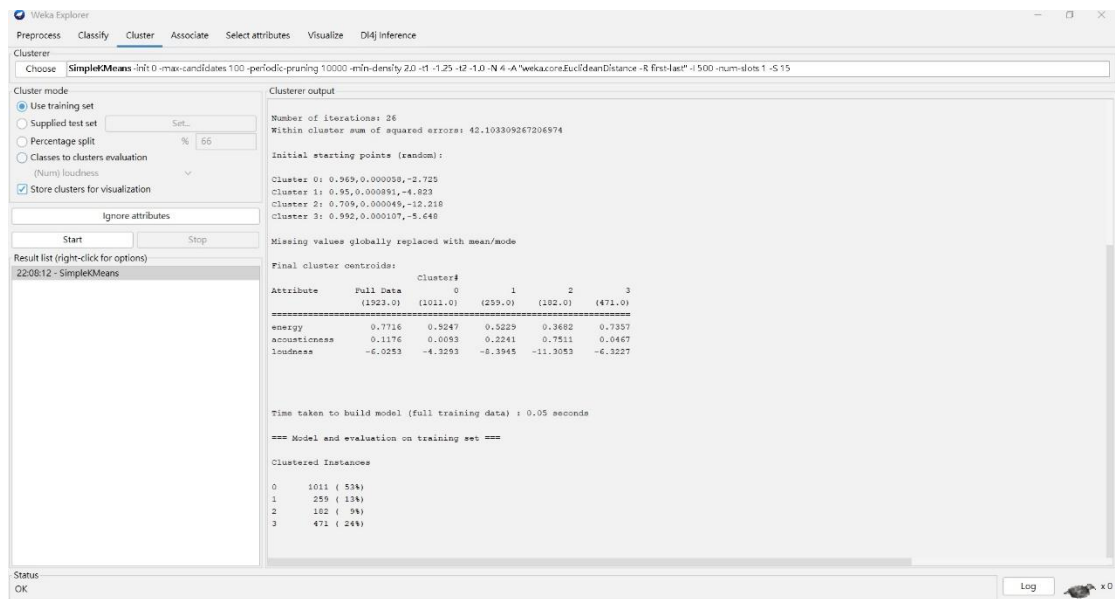
k-means:適用於數字資料

k-modes:適用於類別資料

k-prototype: k-means 與 k-modes 的結合，可用於數字與類別的混合資料

Meanshift: 對類別個數未知的數據，k-means 較難求出解。基於密度的非參數聚類算法，但與 k-means 不同的地方在於不須事先制定 k 值

## 二. 使用 weka 做分群



Weka Explorer

Preprocess Classify **Cluster** Associate Select attributes Visualize D4J inference

Cluster

Choose **SimpleKMeans** -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t 1 -1.25 -k2 -1.0 -N 4 -A "weka.core.EuclideanDistance -R first-last" -i 500 -num-dots 1 -S 15

Cluster mode

☒ Use training set

☐ Supplied test set Set...

☐ Percentage split % 66

☐ Classes to clusters evaluation (Num) loadview

☒ Store clusters for visualization

Ignore attributes

Start Stop

Result list (right-click for options)

220812 - SimpleKMeans

Clusterer output

Number of iterations: 26  
Within cluster sum of squared errors: 42.103309267206974

Initial starting points (random):

Cluster 0: 0.969,0.000059,-2.725  
Cluster 1: 0.95,0.000891,-4.023  
Cluster 2: 0.709,0.000049,-12.218  
Cluster 3: 0.992,0.000107,-5.648

Missing values globally replaced with mean/mode

Final cluster centroids:

| Attribute    | Full Data | Cluster# 0 | Cluster# 1 | Cluster# 2 | Cluster# 3 |
|--------------|-----------|------------|------------|------------|------------|
|              | (1923.0)  | (1011.0)   | (259.0)    | (102.0)    | (471.0)    |
| energy       | 0.7716    | 0.5247     | 0.5229     | 0.3682     | 0.7357     |
| acousticness | 0.1176    | 0.0093     | 0.2241     | 0.7511     | 0.0467     |
| loudness     | -6.0253   | -4.3293    | -8.3945    | -11.3053   | -6.3227    |

Time taken to build model (full training data) : 0.05 seconds

=== Model and evaluation on training set ===

Clustered Instances

| Cluster | Count | Percentage |
|---------|-------|------------|
| 0       | 1011  | ( 53%)     |
| 1       | 259   | ( 13%)     |
| 2       | 102   | ( 5%)      |
| 3       | 471   | ( 24%)     |

Status: OK

Log