Clustering & Analysis of Spotify's top chart songs

Details are discussed [here (https://reikakfujimura.wixsite.com/reikafujimura/post/spotify-music-analysis)](https://reikakfujimura.wixsite.com/reikafujimura/post/spotify-music-analysis)

Import libraries

In [1]:

```
import os

import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_samples
from sklearn.preprocessing import MinMaxScaler
import plotly.express as px
import plotly.graph_objects as go
```

Load data

In [2]:

```
# load us data
data_dir_us = '../preprocess/data/con/us'
df_us = pd.DataFrame()

for dirname, _, filenames in os.walk(data_dir_us):
    for filename in filenames:
        if filename == 'all.csv':
#         if (filename == 'all.csv') & (dirname.split('/')[-1] not in ['2018','2019']): # us vs jp ver.
            tmp = pd.read_csv(os.path.join(dirname, filename))
            df_us = pd.concat([df_us, tmp],axis=0).reset_index(drop=True)
            print(os.path.join(dirname, filename), tmp.shape, df_us.shape)

df_us['date'] = pd.to_datetime(df_us['date'])

# load jp data
data_dir_jp = '../preprocess/data/con/jp'
df_jp = pd.DataFrame()

for dirname, _, filenames in os.walk(data_dir_jp):
    for filename in filenames:
        if filename == 'all.csv':
            tmp = pd.read_csv(os.path.join(dirname, filename))
            df_jp = pd.concat([df_jp, tmp],axis=0).reset_index(drop=True)
            print(os.path.join(dirname, filename), tmp.shape, df_jp.shape)

df_jp['date'] = pd.to_datetime(df_jp['date'])
```

```
../preprocess/data/con/us/2019/all.csv (73000, 23) (73000, 23)
../preprocess/data/con/us/2021/all.csv (62386, 23) (135386, 23)
../preprocess/data/con/us/2020/all.csv (73200, 23) (208586, 23)
../preprocess/data/con/us/2018/all.csv (73000, 23) (281586, 23)
../preprocess/data/con/jp/2021/all.csv (66800, 23) (66800, 23)
../preprocess/data/con/jp/2020/all.csv (73200, 23) (140000, 23)
```

Preparation

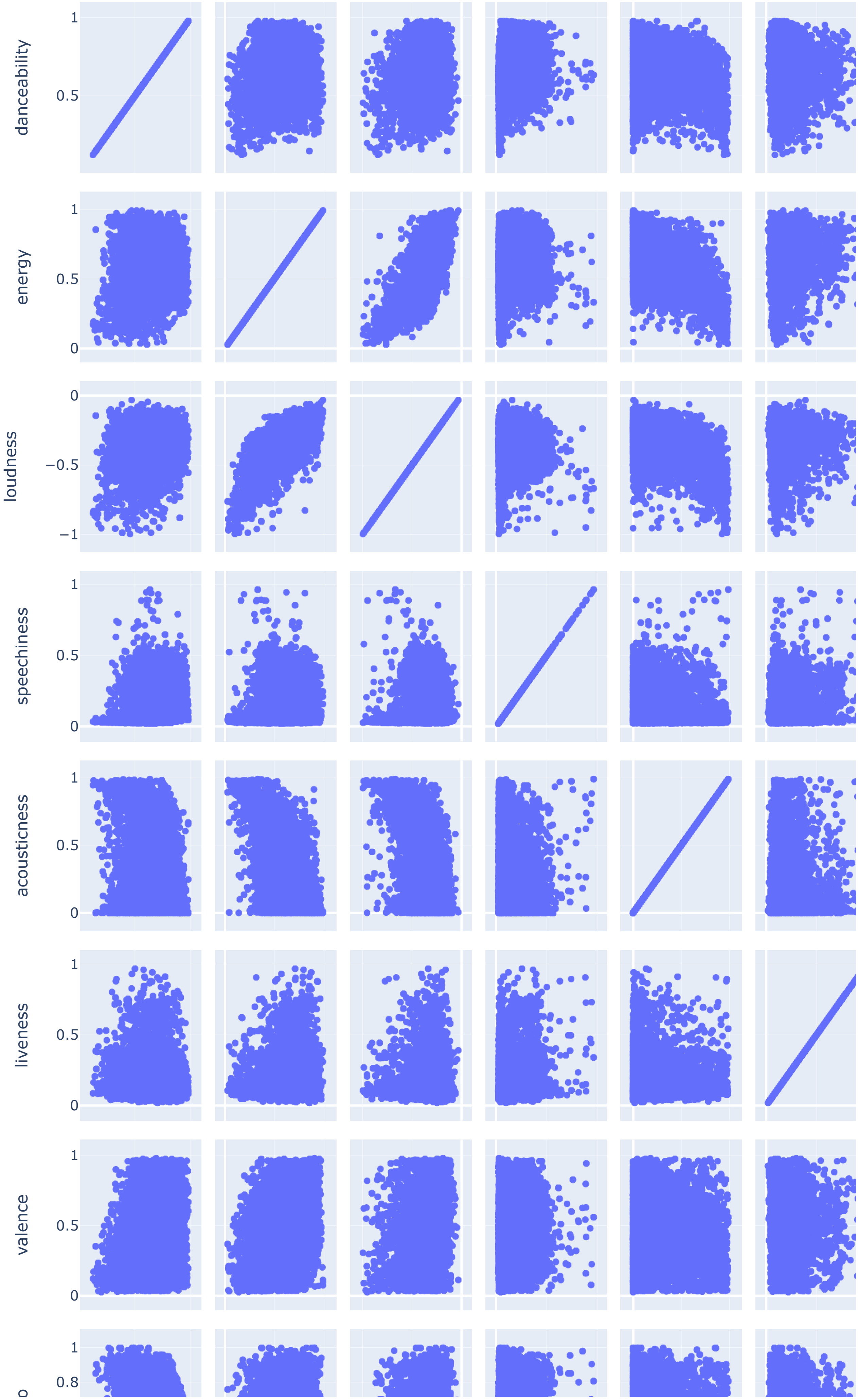
In [3]:

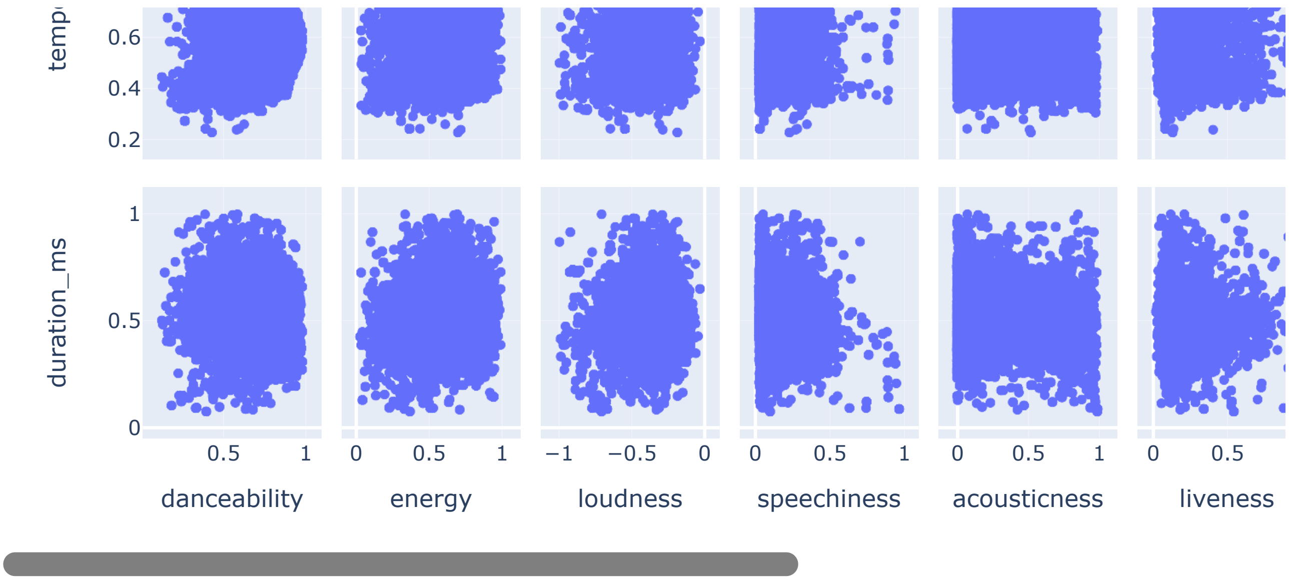
```python
# normalization, data preprocessing

d_xlim = {
    'danceability': [0,1],
    'energy': [0,1],
    'loudness': [-20,0],
    'speechiness': [0,1],
    'acousticness': [0,1],
    'liveness': [0,1],
    'valence': [0,1],
    'tempo': [0,200],
    'duration_ms': [0,4e+5]
}

def normalize(df):
    print(df.shape)
    for feature in d_xlim.keys():
        df = df[(df[feature]<=d_xlim[feature][1]) & (df[feature]>=d_xlim[feature][0]) ]
    print(df.shape)
    for feature in d_xlim.keys():
        df[feature] = df[feature] / np.absolute(d_xlim[feature][1]  - d_xlim[feature][0])
    return df

df_norm_us = df_us.copy()
df_norm_us = normalize(df_norm_us)

df_norm = df_norm_us.copy()

remove_cols = ['title', 'rank', 'date', 'artist', 'url', 'region', 'chart', 'streams','yea
r', 'month', 'mode', 'key', 'instrumentalness','time_signature']
use_cols = [col for col in df_norm.columns if col not in remove_cols]

X = df_norm[use_cols].drop_duplicates().to_numpy()
X_fit = df_norm[use_cols].to_numpy()
```

```
(281586, 23)
(279304, 23)
```

In [4]:

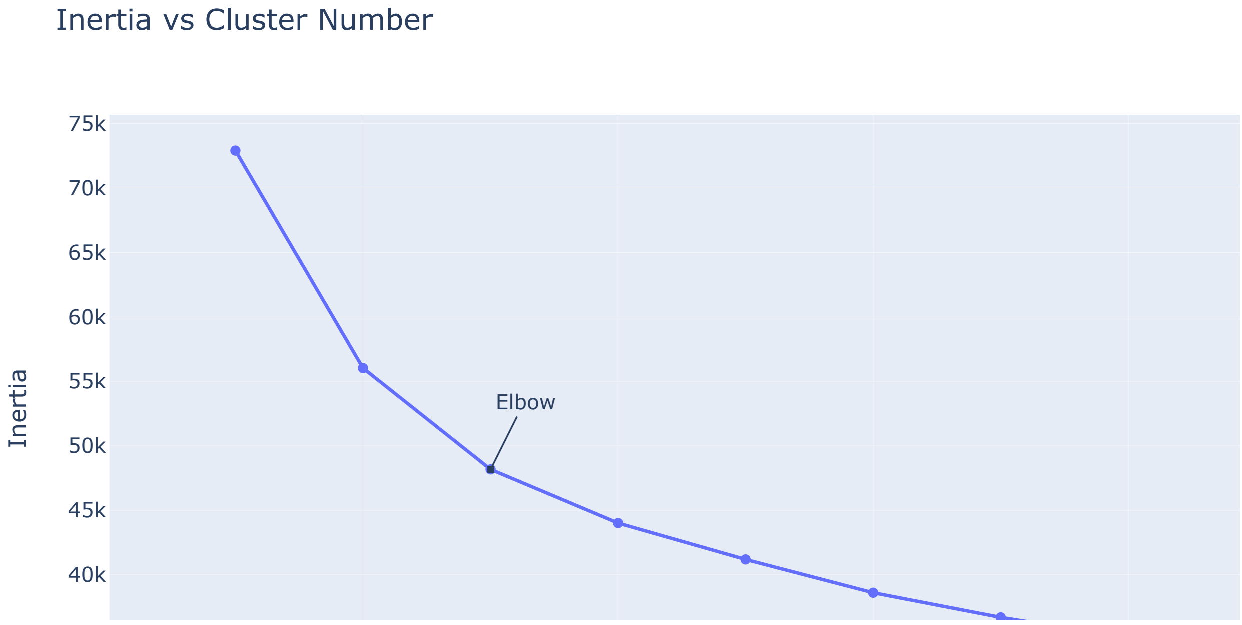
```
# features vizualization (check if there are highly correlated features)

fig = px.scatter_matrix(df_norm[use_cols],
width=1200, height=1600)
fig.show()
```


danceability
1
0.5
energy
1
0.5
0
loudness
0
−0.5
−1
speechiness
1
0.5
0
acousticness
1
0.5
0
liveness
1
0.5
0
valence
1
0.5
0
1
0.8


temp
0.6
0.4
0.2
duration_ms
1
0.5
0
0.5
1
0
0.5
1
−1
−0.5
0
0
0.5
1
0
0.5
1
0
0.5
danceability
energy
loudness
speechiness
acousticness
liveness

In [5]:

```
# elbow method to decide the number of clusters

scaler = MinMaxScaler()
scaler.fit(X_fit)
X_scaler=scaler.transform(X_fit)
inertia = []
for i in range(1,11):
    kmeans = KMeans(
        n_clusters=i, init="k-means++",
        n_init=10,
        tol=1e-04, random_state=42
    )
    kmeans.fit(X_scaler)
    inertia.append(kmeans.inertia_)
fig = go.Figure(data=go.Scatter(x=np.arange(1,11),y=inertia))
fig.update_layout(title="Inertia vs Cluster Number",xaxis=dict(range=[0,11],title="Cluster Number"),
                  yaxis={'title':'Inertia'},
                 annotations=[
        dict(
            x=3,
            y=inertia[2],
            xref="x",
            yref="y",
            text="Elbow",
            showarrow=True,
            arrowhead=7,
            ax=20,
            ay=-40
        )
    ])
```



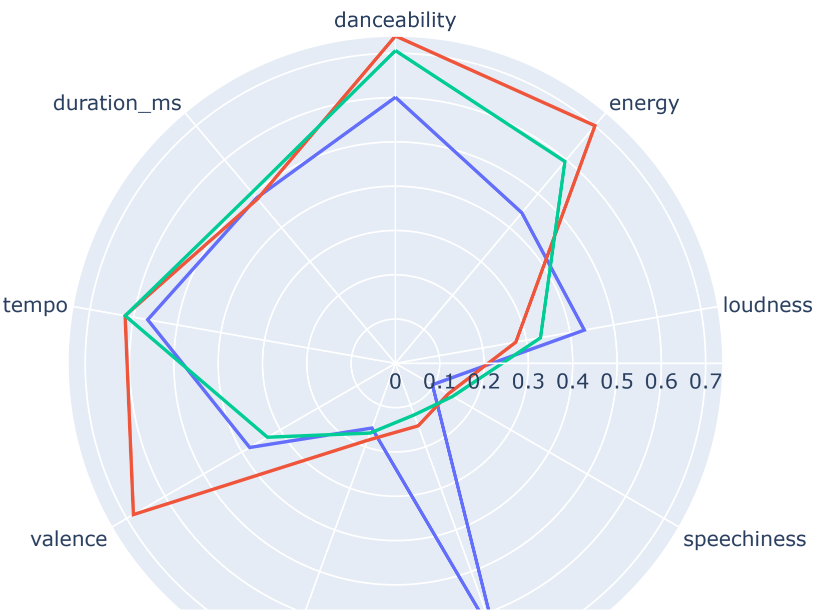
Clustering

```
In [6]: # k-means clustering

        n_clusters=3
        kmeans = KMeans(n_clusters = n_clusters,
                        init = 'k-means++',
                        n_init= 10,
                        max_iter=350,
                        tol=1e-04,
                        random_state = 42
                       )
        pred_y = kmeans.fit_predict(X_fit)

        df_norm['class_pred'] = pred_y
        df_norm['class_pred'].value_counts()
```

```
Out[6]: 2    131051
        1     95037
        0     53216
        Name: class_pred, dtype: int64
```

Analysis

```
In [7]: # visualize the average of features of three classes
        use_cols = use_cols + ['class_pred']
        df_norm['loudness'] = - df_norm['loudness']
        polar=df_norm[use_cols].groupby(["class_pred"]).mean().reset_index()
        polar=pd.melt(polar,id_vars=["class_pred"])
        fig = px.line_polar(polar, r="value", theta="variable", color="class_pred", line_close=Tru
        e)
        fig.show()

        df_norm['loudness'] = - df_norm['loudness']
```



In [8]:

```
# calculate silhouette constant

cluster_labels = np.unique(pred_y)
n_clusters = cluster_labels.shape[0]

silhouette = silhouette_samples(X_fit,pred_y, metric="euclidean")
df_norm['silhouette'] = silhouette
```

In [16]:

```
# class 0
# -> ballads, folk
df_0 = df_norm[df_norm['class_pred']==0][['artist','title','url','class_pred','silhouette'
]].drop_duplicates()
df_0.sort_values(by=['silhouette'], ascending=False).reset_index(drop=True)[:10][['artist'
,'title']]
```

Out[16]:

	artist	title
0	Taylor Swift	illicit affairs
1	Ed Sheeran	First Times
2	Taylor Swift	gold rush
3	Michael Bublé	Home
4	Taylor Swift	seven
5	Chan Se Park	Thinkin of You
6	A$AP Rocky	CALLDROPS (feat. Kodak Black)
7	Shawn Mendes	Perfectly Wrong
8	Kaash Paige	Love Songs - Bonus
9	Billie Eilish	Male Fantasy

In [17]:

```
# class 1
# -> happy-vibe pop
df_1 = df_norm[df_norm['class_pred']==1][['artist','title','url','class_pred','silhouette'
]].drop_duplicates()
df_1.sort_values(by=['silhouette'], ascending=False).reset_index(drop=True)[:10][['artist'
,'title']]
```

Out[17]:

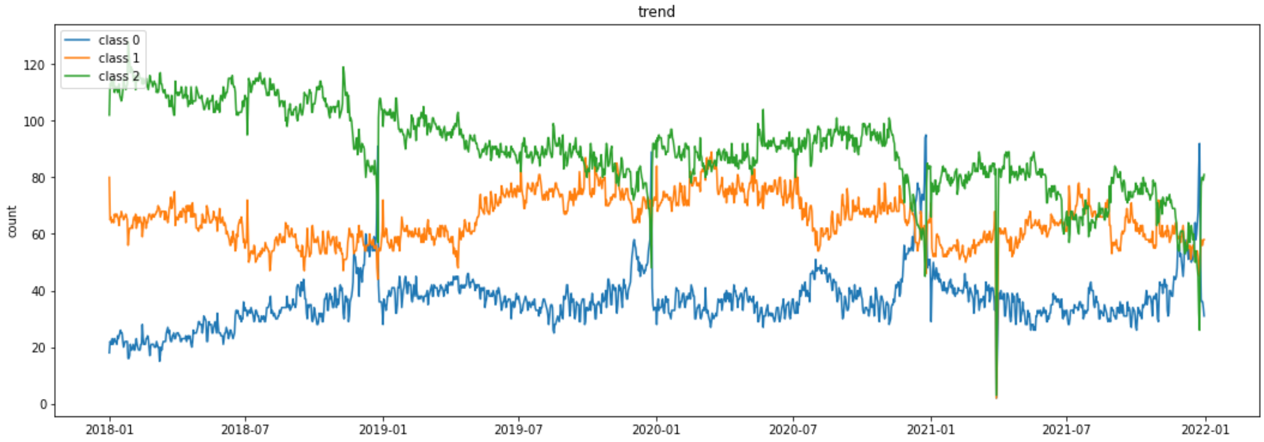
	artist	title
0	Avril Lavigne	Dumb Blonde (feat. Nicki Minaj)
1	Doja Cat	Woman
2	Lil Uzi Vert	20 Min
3	Shakira	Whenever
4	Lil Peep	I've Been Waiting (w/ ILoveMakonnen & Fall Ou...
5	Lil Uzi Vert	Celebration Station
6	Regard	Ride It
7	Shakira	Chantaje (feat. Maluma)
8	BTS	Filter
9	Thomas Rhett	Beer Can’t Fix

In [18]:
```
# class 2
# -> hiphop
df_2 = df_norm[df_norm['class_pred']==2][['artist','title','url','class_pred','silhouette'
]].drop_duplicates()
df_2.sort_values(by=['silhouette'], ascending=False).reset_index(drop=True)[:10][['artist'
,'title']]
```

Out[18]:

	artist	title
0	Twenty One Pilots	No Chances
1	J. Cole	m y . l i f e (with 21 Savage & Morray)
2	Billie Eilish	Oxytocin
3	NAV	Just Happened
4	Future	HATE THE REAL ME
5	Andy Grammer	Don't Give Up On Me - (From "Five Feet Apart")
6	Don Toliver	Way Bigger
7	Kanye West	Remote Control
8	Trippie Redd	Weeeeee
9	Future	Stick to the Models

In [12]:
```
# time plot, classes

# 0 -> ballads, folk
# 1 -> happy-vibe pop
# 2 -> hiphop

fig, ax = plt.subplots(figsize=(18,6))
for k in range(n_clusters):
    tmp = df_norm[df_norm.class_pred==k]
    t = list(tmp.groupby('date').title.count().index)
    y = list(tmp.groupby('date').title.count())
    ax.plot(t, y, label='class ' + str(k))

ax.set_ylabel('count')
ax.legend(loc = 'upper left')
ax.title.set_text('trend')

plt.show()
```


trend
class 0
class 1
class 2
count