MAJORProject : Music Mood Prediction

Krishi Agrahari,Rajsi Kesharwani,Nikhil Kamale,Shazia Khan,Kirti Mohitkar

In [1]:
```
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
import requests
import json
warnings.filterwarnings("ignore")
```

Part 1: Dataset Preparation :

We will use Million Song Data Subset, containing 10000 songs' data as the Dataset for our Mood Prediction Classifier

In [2]:
```
import os, sys
dir_tree = 'MillionSongSubset'
for dir_path, dir_names, file_names in os.walk(dir_tree):
    for file_name in file_names:
        try:
            os.rename(os.path.join(dir_path, file_name), os.path.join(di
r_tree, file_name))
        except OSError:
            print ("Could not move %s " % os.join(dir_path, file_name))
```

Create Pandas Dataframe with following Columns :

1. File name
2. Artist Name
3. Song Title
4. Lyrics (Empty for Now)

In [3]:
```
def make_artist_table(base):
    files = [os.path.join(base,fn) for fn in os.listdir(base) if fn.ends
with('.h5')]
    data = {'file':[], 'artist':[], 'title':[]}
    for f in files:
        store = pd.HDFStore(f)
        title = store.root.metadata.songs.cols.title[0]
        artist = store.root.metadata.songs.cols.artist_name[0]
        data['file'].append(os.path.basename(f))
        data['title'].append(title.decode("utf-8"))
        data['artist'].append(artist.decode("utf-8"))
        store.close()
    df = pd.DataFrame.from_dict(data, orient='columns')
    df = df[['file', 'artist', 'title']]
    return df
```

In [4]:
```
base = 'MillionSongSubset'
df = make_artist_table(base)
df['lyrics'] = pd.Series('', index=df.index)
df.tail()
```

Out[4]:

	file	artist	title	lyrics
9996	TRAZSEY128F424C12B.h5	Ivan Parker	Close To The Well (Live)	
9997	TRASTIS128F92FA998.h5	Eri Esittäjiä	Dawn	
9998	TRAFKDG12903CAB85C.h5	Webb Wilder	Too Cool for Love	
9999	TRATJTH128F92EF33C.h5	The Bureau De Change	Seasick	
10000	TRACCVN128F92DAE3D.h5	String Trio Of New York	Ju Ju	

In [ ]:
```
df.to_csv('lyrics_ml.csv', index=False,encoding='utf-8')
```

In [0]:
```
import pandas as pd
df=pd.read_csv("lyrics_ml.csv")
```

In [2]:
```
df.head()
```

Out[2]:

	file	artist	title	lyrics
0	TRAWLGG128F42884CA.h5	Usher	Pianolude	NaN
1	TRAIQDI128F92E8E1F.h5	Seal	Still Love Remains (Album Version)	NaN
2	TRAHQNV12903CECAB0.h5	Terminal 11 feat. b.b0Nd	In Place Of Love [edit]	NaN
3	TRAUVAQ128F92DC677.h5	Fake Problems	Tabernacle Song	NaN
4	TRAOOQN12903CEB763.h5	Dottie Rambo_ The Whites	New Shoes	NaN

Using the Artist Name and Song Title, We are fetching lyrics for all the songs using PyLyrics package, which uses LyricWikia.com API to get lyrics for songs

In [ ]:
```
from PyLyrics import *
for row_id in df.index:
    #print(str(row_id),end=" ")
    try:
        lyr = PyLyrics.getLyrics(df.loc[row_id]['artist'],df.loc[row_id]
['title'])
        df.loc[row_id,'lyrics'] = lyr
    except:
        continue
```

Out of the 10000 songs, we could only fetch 3449 song lyrics from LyricWikia.com API

In [9]:
```
print('downloaded Lyrics for %s songs' %sum(df.lyrics!=))
df.head()
```

```
downloaded Lyrics for 10001 songs
```

Out[9]:

	file	artist	title	lyrics
0	TRAWLGG128F42884CA.h5	Usher	Pianolude	Can I love you darling,\nCan I love you darlin...
1	TRAIQDI128F92E8E1F.h5	Seal	Still Love Remains (Album Version)	How will I stand if you turn out the light\nTh...
2	TRAHQNV12903CECAB0.h5	Terminal 11 feat. b.b0Nd	In Place Of Love [edit]	NaN
3	TRAUVAQ128F92DC677.h5	Fake Problems	Tabernacle Song	We've almost got job security,\nAt least there...
4	TRAOOQN12903CEB763.h5	Dottie Rambo_ The Whites	New Shoes	NaN

In [0]:
```
df1 = df[pd.notnull(df['lyrics'])]
df1.head()
df1.shape
df2=df
df=df1
```

In [0]:
```
df.to_csv('df_lyr_backup.csv')
```

In [19]:
```
df = df[df.lyrics!='']
df.shape
```

Out[19]:
```
(3449, 4)
```

We are creating our model using english lyrics. So all the song lyrics containing any other language are removed from the Dataset. Total 495 songs are removed.

In [0]:
```
import nltk

def eng_ratio(text):
    english_vocab = set(w.lower() for w in nltk.corpus.words.words())
    text_vocab = set(w.lower() for w in text.split() if w.lower().isalph
a())
    unusual = text_vocab.difference(english_vocab)
    diff = len(unusual)/len(text_vocab)
    return diff
```

In [22]:
```
nltk.download('words')
```

```
[nltk_data] Downloading package words to /root/nltk_data...
[nltk_data]   Unzipping corpora/words.zip.
```

Out[22]:
```
True
```

In [23]:
```
before = df.shape[0]
for row_id in df.index:
    text = df.loc[row_id]['lyrics']
    diff = eng_ratio(text)
    if diff >= 0.5:
        df = df[df.index != row_id]
after = df.shape[0]
rem = before - after
print('%s have been removed.' %rem)
print('%s songs remain in the dataset.' %after)
```

```
495 have been removed.
2954 songs remain in the dataset.
```

Now we will use Last.FM API to extract Tags for the remaining ~3000 songs in our dataset. Tags can be based on Genre, Mood, Artist Type etc.

In [0]:
```
def getSongTags(artist,track):
    url = "http://ws.audioscrobbler.com/2.0/?method=track.getTopTags&api
_key=0f6916aff634cb3e768baa9d5ee89341&artist="+artist+"&track="+track+"&
format=json"
    results = requests.get(url).json()
    tagList = []
    if 'toptags' in results:
        toptags = results['toptags']
        if 'tag' in toptags:
            taglistss = toptags['tag']
            for tagItem in taglistss:
                tagList.append(tagItem['name'])
    return tagList
```

In [30]:
```
import pyprind
df['tags'] = ''
pbar = pyprind.ProgBar(df.shape[0])
for row_id in df.index:
    #print(row_id,end=" ")
    try:
      tags = getSongTags(df.loc[row_id]['artist'],df.loc[row_id]['title'
])
      df.loc[row_id,'tags'] = tags
      pbar.update()
    except:
      continue
```

```
0% [############################# ] 100% | ETA: 00:00:13
```

In [0]:
```
df.head()
df.to_csv('df_tag_backup.csv')
```

In [0]:
```
for row_id in df.index:
    if len(df.loc[row_id,'tags'])==0:
        df = df.drop(row_id)
```

Out of 3000, We extracted tags from Last.FM API for 2289 songs. Rest of them are removed

In [34]:
```
df.shape
```

Out[34]:
```
(2289, 5)
```

In [36]:
```
df.head()
```

Out[36]:

	file	artist	title	lyrics	tags
0	TRAWLGG128F42884CA.h5	Usher	Pianolude	Can I love you darling,\nCan I love you darlin...	[p, soul, rnb, Usher, r'n'b]
17	TRBDLRT128F4296B2F.h5	Alias	Dying To Stay	Initial thought: turn it up a notch from the r...	[trip-hop, rap, test, Hip-Hop, experimental, u...
23	TRBEKNV128F9326A8A.h5	Killswitch Engage	Never Again (Album Version)	I want you to suffer as I have suffered\nThis ...	[metalcore]
24	TRAUTCA128F429B624.h5	Rickie Lee Jones	Danny's All-Star Joint (LP Version)	Downstairs at danny's all-star joint\nThey got...	[hopuke42, pivudo45, singer-songwriter, classi...
26	TRAIVNL128F422CFE2.h5	Buzzcocks	Get On Our Own	Honey, I saw you yesterday\nOn my way home\nBa...	[punk, punk rock, british i like, 70s, british...

In [0]:
```
df.to_csv('df_tag_backup.csv')
```

In the paper "Lyric Text Mining in Music Mood Classification" - Hu et.al, tags are being grouped into 18 categories according to different moods. We have taken 10 groups for our Mood Categories - Happy, Sad, Angry, Relax.

In [0]:
```
relaxTags="calm, comfort, quiet, serene, mellow, chill out, calm down, c
alming, chillout, comforting, content, cool down, mellow music,  mellow
 rock, peace of mind, quietness, relaxation, serenity, solace, soothe, s
oothing, still, tranquil, tranquility, tranquility,brooding, contemplati
ve, meditative, reflective, broody, pensive, pondering, wistful,desire,
 hope, hopeful"
relaxTags = relaxTags.replace(" ","").split(",")

happyTags = "cheerful, cheer up, festive, jolly, jovial, merry, cheer, c
heering,\
cheery, get happy, rejoice, songs that are cheerful, sunny,happy, happin
ess, happy songs, happy music, glad, mood: happy,\
upbeat, gleeful, high spirits, zest, enthusiastic, buoyancy, elation, mo
od: upbeat,excitement, exciting, exhilarating, thrill,\
ardor, stimulating, thrilling, titillating"
happyTags = happyTags.replace(" ","").split(",")

sadTags = "sad, sadness, unhappy, melancholic, melancholy, feeling sad,
 mood: sad - slightly, sad song,\
depressed, blue, dark, depressive, dreary, gloom, darkness, depress, dep
ression, depressing, gloomy,\
anger, angry, choleric, fury, outraged, rage, angry music,grief, heartbr
eak, mournful, sorrow, sorry, doleful, heartache, heartbreaking, heartsi
ck, lachrymose, mourning,\
plaintive, regret, sorrowful"
sadTags = sadTags.replace(" ","").split(",")

angryTags="anger, angry, choleric, fury, outraged, rage, angry music,agg
ression, aggressive,\
angst, anxiety, anxious, jumpy, nervous, angsty,pessimism, cynical, pess
imistic, weltschmerz, cynical/sarcastic"
angryTags = angryTags.replace(" ","").split(",")
```

By correlating the tags that we found from Last.FM and the tag groups generated by us, we are creating the Class Label for Moods in our Dataset. The Class Labels are -

1. Happy - 1
2. Sad - 2
3. Angry - 3
4. Relaxed -4

In [41]:
```
import numpy as np
df['mood']=""
pbar = pyprind.ProgBar(df.shape[0])
for row_id in df.index:
    tags = df.loc[row_id,'tags']
    sad_tags = np.intersect1d(tags,sadTags)
    happy_tags = np.intersect1d(tags,happyTags)
    relax_tags = np.intersect1d(tags,relaxTags)
    angry_tags = np.intersect1d(tags,angryTags)
    mmax=max(len(sad_tags),len(happy_tags),len(relax_tags),len(angry_tag
s))
    if len(sad_tags)>0 or len(happy_tags)>0 or len(angry_tags)>0 or len(
relax_tags)>0:# having mood tag
        if len(happy_tags)==mmax:
          df.loc[row_id,'mood'] = "1"
        elif len(sad_tags)==mmax:
          df.loc[row_id,'mood'] = "2"
        elif len(angry_tags)==mmax:
          df.loc[row_id,'mood'] = "3"
        elif len(relax_tags)==mmax:
          df.loc[row_id,'mood'] = "4"
    else:
        df = df.drop(row_id)# remove songs that does not have tag
    pbar.update()
```

```
0% [##############################] 100% | ETA: 00:00:00
Total time elapsed: 00:00:02
```

Out of 2300 songs, tags are correlated for only 605 songs. Our new dataset carries only these songs.

In [42]:
```
df.shape
```

Out[42]:
```
(605, 6)
```

In [43]:
```
df.head()
```

Out[43]:

	file	artist	title	lyrics	tags	mood
41	TRABNEX128F92C9DEA.h5	Kings Of Leon	Knocked Up	I don't care what nobody says, we're gonna hav...	[rock, indie rock, indie, Southern Rock, kings...	2
77	TRAESWB128F149CB2C.h5	Dexys Midnight Runners	Until I Believe In My Soul	I'll need tonight to sit and think about this\...	[80s, soul, british, Blue-Eyed Soul, new wave,...	3
86	TRAKQXJ128F147A028.h5	AFI	Summer Shudder	Listen when I say, when I say it's real\nReal ...	[punk rock, rock, punk, emo, AFI, alternative,...	2
125	TRAYYIN12903CAD8C6.h5	K-OS	Man I Used To Be	I tried it, I couldn't find it\nNow I just wan...	[Hip-Hop, Canadian, hip hop, k-os, rap, pop, h...	2
126	TRAKHYP128E0792F07.h5	Patty Griffin	Moses	Diamonds, roses I need Moses\nTo cross this se...	[female vocalists, folk, acoustic, Alt-country...	2

In [0]:
```
df.to_csv('df_mood_backup.csv')
```

From the paper "Multimodal Music Mood Classification by Fusion of Audio and Lyrics" - Hao et.al , we get to know about 777 other songs, already categorized into Happy, Sad, Angry, Relaxed. We append this dataset with our previous dataset.

In [163]:
```
import os
import re
import itertools
import pickle
from collections import Counter,defaultdict
try:
        import pandas as pd
        import numpy as np
        from nltk.corpus import stopwords
        from nltk.stem import PorterStemmer
        import gensim
        from gensim.models import Word2Vec
        from nltk.stem import WordNetLemmatizer
        from keras.utils import np_utils
except:
        print("require modules: keras,gensim,nltk.stem,nltk.corpus,nltk.
stem, please install it.")
        exit()
```

In [164]:
```
reg1 = re.compile("\.txt$")
reg2 = re.compile("([0-9]+)\.txt")
reg3 = re.compile(".*_([0-9])\.txt")
reg4 = re.compile("\[.+\]")
reg5 = re.compile("info\.txt")
lyrics_dic = {}
for i in os.listdir():
    if os.path.isdir(i):
        for path,sub,items in os.walk(i):
            if any([reg1.findall(item) for item in items]):
                for item in items:
                    if reg5.findall(item):
                        continue
                    if reg3.findall(item):
                        num = ["0"+reg3.findall(item)[0]]
                        name = "_".join(path.split("/") + num)
                    else:
                        name = "_".join(path.split("/") + reg2.findall(i
tem))
                    with open(os.path.join(path,item),"r",encoding="utf
8", errors='ignore') as f:
                        lyrics = "".join(f.readlines())
                        lyrics = reg4.subn("",lyrics)[0]
                        lyrics_dic[name] = lyrics
```

In [165]:
```
len(lyrics_dic.keys())
```

Out[165]:
```
777
```

Creating the dataframe with lyrics and mood

In [166]:
```
df=pd.DataFrame(columns=["lyrics","mood"])
```

In [167]:
```
for key in lyrics_dic.keys():
    #print(key)
    if "Happy" in key:

        df=df.append({"lyrics":lyrics_dic[key],"mood":"1"},ignore_index=
True)
    elif "Sad" in key:
        df=df.append({"lyrics":lyrics_dic[key],"mood":"2"},ignore_index=
True)
    elif "Angry" in key:
        df=df.append({"lyrics":lyrics_dic[key],"mood":"3"},ignore_index=
True)
    elif "Relaxed" in key:
        df=df.append({"lyrics":lyrics_dic[key],"mood":"4"},ignore_index=
True)
```

In [168]:
```
df.head()
```

Out[168]:

	lyrics	mood
0	Put your lips close to mine\nAs long as they d...	1
1	My lullaby,hung out to dry\nWhat's up with tha...	1
2	Though you've played at love and lost\nAnd sor...	1
3	we know we are the ones\nwho do it numb again\...	1
4	Another day has come and gone\nThey fade away ...	1

In [169]:
```
df = df.sample(frac=1).reset_index(drop=True)
```

In [170]:
```
df_new=pd.read_csv("df_mood_backup.csv")
```

In [171]:
```
df_new.tail()
```

Out[171]:

	Unnamed: 0	file	artist	title	lyrics	tags	m
600	9913	TRAREDZ12903CFB6E3.h5	Pelle Carlberg	Clever Girls Like Clever Boys Much More Than C...	You should have listened to what mama said\nAn...	['handclaps', 'title is a full sentence', 'ind...	
601	9943	TRATTMT128F149167B.h5	Michael Jackson	Ain't No Sunshine	Spoken Intro:\nYou ever want something\nThat y...	['soul', 'michael jackson', '70s', 'pop', 'cov...	
602	9966	TRAWFGF128E0792FE0.h5	Extreme	Stop The World	All the world's a masquerade\nMade up of fools...	['hard rock', 'rock', 'funk metal', 'Power bal...	
603	9967	TRAWFVE128F42912CA.h5	Dimmu Borgir	Sympozium	Chains of despair cloacked by darkness\nThe th...	['Symphonic Black Metal', 'black metal', 'melo...	
604	9969	TRBDMIN128F147FCBB.h5	Phil Collins	One More Night	One more night\nOne more night\n\nI've been tr...	['80s', 'pop', 'Phil Collins', 'soft rock', 'r...	

In [172]:
```
df_new=df_new[["lyrics","mood"]]
```

Append the previous Dataset of 605 songs with this new dataset, containing 777 songs. Our new dataset contains 1382 song lyrics and corresponding mood categories

In [173]:
```
df3=df.append(df_new,ignore_index=True)
```

In [174]:
```
df3["mood"]=df3["mood"].apply(str)
```

In [175]:
```
df3.shape
```

Out[175]:
```
(1382, 2)
```

In [176]:
```
set(df3["mood"].values)
df=df3
```

For the Test Dataset, we have taken lyrics from hindi bollywood songs, translated them into english using Google Translate API, annotated them with Happy, Sad, Angry, Relaxed (for checking accuracy on the hindi songs)

In [77]:
```
df_new=pd.read_excel("test.xls")
```

In [78]:
```
df_new.shape
```

Out[78]:
```
(235, 5)
```

In [79]:
```
df_new=df_new[['lyrics','mood']]
df_new=df_new.dropna()
```

In [80]:
```
df_new['mood']=df_new['mood'].astype(int)
```

In [81]:
```
df_new['mood'].value_counts()
df_new["mood"]=df_new["mood"].apply(str)
```

As our test dataset is labelled manually and training dataset is auto-labelled from the tags of Last.FM and Rao's paper, we add a fraction of translated-to-english songs to our training dataset to level the trueness of the testing dataset.

In [42]:
```
df=df.append(df_new.sample(a),ignore_index=True)
df = df.sample(frac=1).reset_index(drop=True)
self.df['mood'].values()
```

Out[42]:
```
['1', '2', '3', '4']
```

df contains the initial training dataset for our prediction model.

In [ ]:
```
df_to_csv('training_backup.csv',index=False)
```

df_new contains the initial testing dataset for our prediction model.

In [ ]:
```
df_new.to_csv('testing_backup.csv',index=False)
```

In [ ]: