In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from IPython import get_ipython
import warnings
warnings.filterwarnings("ignore")
```

In [2]:

```
data1 = pd.read_csv('music_sample.csv')
data2 = pd.read_csv('music_train.csv')
data3 = pd.read_csv('music_test.csv')
```

In [3]:

```
data1.head()
```

Out[3]:

	id	class
0	0	0
1	1	0
2	2	0
3	3	0
4	4	0

In [4]:

```
data1.tail()
```

Out[4]:

	id	class
2671	2671	0
2672	2672	0
2673	2673	0
2674	2674	0
2675	2675	0

In [5]:

```
data1.shape
```

Out[5]:

```
(2676, 2)
```

In [6]:

```
data1.columns
```

Out[6]:

```
Index(['id', 'class'], dtype='object')
```

In [7]:

```
data1.isnull().sum()
```

Out[7]:

```
id       0
class    0
dtype: int64
```

In [8]:

```
data1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2676 entries, 0 to 2675
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   id      2676 non-null   int64
 1   class   2676 non-null   int64
dtypes: int64(2)
memory usage: 41.9 KB
```

In [9]:

```
data1.describe()
```

Out[9]:

	id	class
count	2676.000000	2676.0
mean	1337.500000	0.0
std	772.638984	0.0
min	0.000000	0.0
25%	668.750000	0.0
50%	1337.500000	0.0
75%	2006.250000	0.0
max	2675.000000	0.0

In [10]:

```
data2.head()
```

Out[10]:

	lyric	class
0	Can't drink without thinkin' about you	1
1	Now Lil Pump flyin' private jet (Yuh)	0
2	No, matter fact, you ain't help me when I had ...	0
3	And you could find me, I ain't hidin'	0
4	From the way you talk to the way you move	1

In [11]:

```
data2.tail()
```

Out[11]:

	lyric	class
51049	I told her pour me some more, then she went ri...	0
51050	Hit the ground and crawl to the dresser	0
51051	Just keep breathin' and breathin' and breathin...	1
51052	Down go the system, long live the king (King)	0
51053	If your mother knew all the things we do (From...	1

In [12]:

```
data2.shape
```

Out[12]:

```
(51054, 2)
```

In [13]:

```
data2.columns
```

Out[13]:

```
Index(['lyric', 'class'], dtype='object')
```

In [14]:

```
data2.duplicated().sum()
```

Out[14]:

```
15318
```

In [15]:

```
data2.isnull().sum()
```

Out[15]:

```
lyric    0
class    0
dtype: int64
```

In [16]:

```
data2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51054 entries, 0 to 51053
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   lyric   51054 non-null  object
 1   class   51054 non-null  int64 
dtypes: int64(1), object(1)
memory usage: 797.8+ KB
```

In [17]:

```
data2.describe()
```

Out[17]:

	class
count	51054.000000
mean	0.434227
std	0.495660
min	0.000000
25%	0.000000
50%	0.000000
75%	1.000000
max	1.000000

In [18]:

```
data3.head()
```

Out[18]:

	id	lyric
0	0	Now they know my name wherever I go
1	1	If your girl don't get it poppin', put me on y...
2	2	P1 cleaner than your church shoes, ah
3	3	Bodies start to drop, ayy (Hit the floor)
4	4	I don't look to the sky no mo'

In [19]:

```
data3.tail()
```

Out[19]:

	id	lyric
2671	2671	So tell me, how deep is your love?
2672	2672	If this is all we're living for
2673	2673	I'll never let up on the pedal, might as well ...
2674	2674	Turned my temple down into a prison, s***
2675	2675	Yeah that's a fact, but I never been p****

In [20]:

```
data3.shape
```

Out[20]:

```
(2676, 2)
```

In [21]:

```
data3.columns
```

Out[21]:

```
Index(['id', 'lyric'], dtype='object')
```

In [22]:

```
data3.duplicated().sum()
```

Out[22]:

```
0
```

In [23]:

```
data3.isnull().sum()
```

Out[23]:

```
id       0
lyric    0
dtype: int64
```

In [24]:

```
data2['class'].unique()
```

Out[24]:

```
array([1, 0], dtype=int64)
```

In [25]:

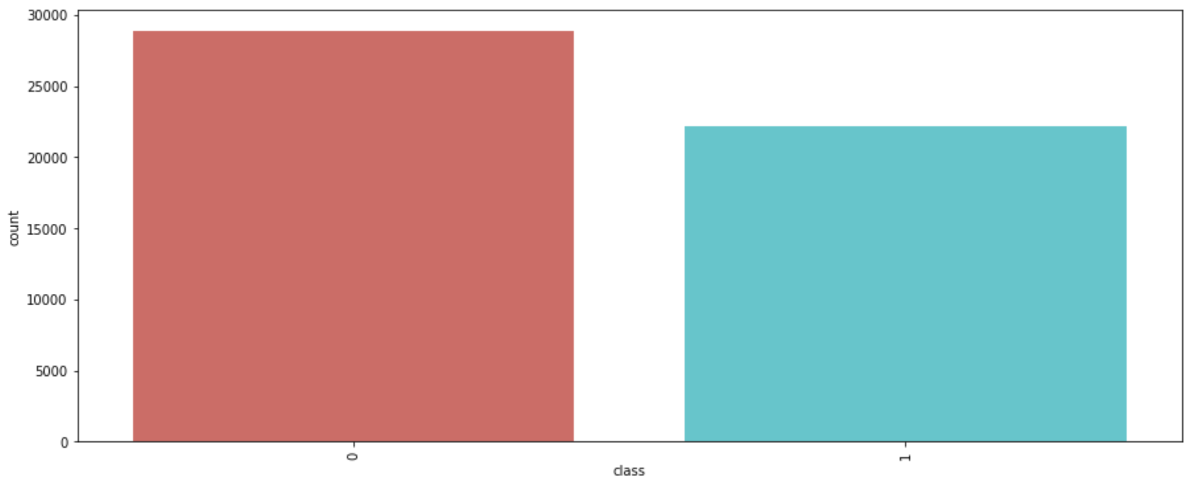
```
data2['class'].value_counts()
```

Out[25]:

```
0    28885
1    22169
Name: class, dtype: int64
```

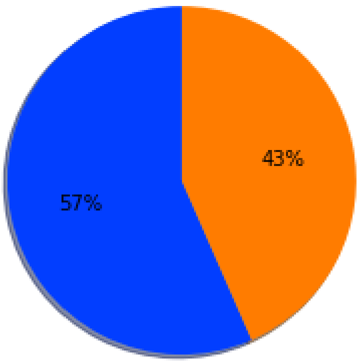
In [26]:

```
plt.figure(figsize=(15,6))
sns.countplot('class', data = data2, palette='hls')
plt.xticks(rotation = 90)
plt.show()
```



In [28]:

```
colors = sns.color_palette('bright')
plt.pie(data2['class'].value_counts(),
        colors = colors,
        autopct = '%0.0f%%', shadow = 'True',
        startangle = 90)
plt.show()
```



In [29]:

```
data4 = data2.copy()
```

In [30]:

```
data4['total_length_characters'] = data4['lyric'].str.len()
print(data4['total_length_characters'])
total_length_characters = data4['total_length_characters'].sum()
print(total_length_characters)
count = 0
for y in data4["lyric"]:
    count = count + 1
print(count)
average_length = total_length_characters / count
print (average_length)
```

```
0        38
1        37
2        54
3        37
4        41
         ..
51049    73
51050    39
51051    61
51052    45
51053    55
Name: total_length_characters, Length: 51054, dtype: int64
2032812
51054
39.81689975320249
```

In [32]:

```
data4['total_count_words'] = data4['lyric'].str.split().str.len()
print(data4['total_count_words'])
total_words = data4['total_count_words'].sum()
print(total_words)
count = 0
for y in data4["lyric"]:
    count = count + 1
print(count)
average_words = total_words / count
print (average_words)
```

```
0         6
1         7
2        12
3         8
4        10
         ..
51049    16
51050     8
51051     9
51052     9
51053    12
Name: total_count_words, Length: 51054, dtype: int64
418280
51054
8.19289379872292
```

In [35]:

```
import string
import re
import nltk
from nltk.util import pr
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from nltk.corpus import stopwords
stemmer = nltk.SnowballStemmer("english")
nltk.download('stopwords')
stopword=set(stopwords.words('english'))
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\pc\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\stopwords.zip.
```

In [36]:

```
def clean(text):
    text = str(text).lower()
    text = re.sub('\[.*?\]', '', text)
    text = re.sub('https?://\S+|www\.\S+', '', text)
    text = re.sub('<.*?>+', '', text)
    text = re.sub('[%s]' % re.escape(string.punctuation),
                  '', text)
    text = re.sub('\n', '', text)
    text = re.sub('\w*\d\w*', '', text)
    text = [word for word in text.split(' ') if word not in stopword]
    text=" ".join(text)
    text = [stemmer.stem(word) for word in text.split(' ')]
    text=" ".join(text)
    return text
data4["lyric"] = data4["lyric"].apply(clean)
```

In [37]:

```
data4['total_length_characters'] = data4['lyric'].str.len()
print(data4['total_length_characters'])
total_length_characters = data4['total_length_characters'].sum()
print(total_length_characters)
count = 0
for y in data4["lyric"]:
    count = count + 1
print(count)
average_length = total_length_characters / count
print (average_length)
```

```
0        26
1        29
2        27
3        21
4        17
         ..
51049    30
51050    24
51051    40
51052    29
51053    20
Name: total_length_characters, Length: 51054, dtype: int64
1155941
51054
22.64153641242606
```

In [38]:

```
data4['total_count_words'] = data4['lyric'].str.split().str.len()
print(data4['total_count_words'])
total_words = data4['total_count_words'].sum()
print(total_words)
count = 0
for y in data4["lyric"]:
    count = count + 1
print(count)
average_words = total_words / count
print (average_words)
```

```
0        4
1        6
2        5
3        4
4        4
        ..
51049    6
51050    4
51051    5
51052    6
51053    4
Name: total_count_words, Length: 51054, dtype: int64
229901
51054
4.5030947624084305
```

In [39]:

```
def clean(text):
    text = str(text).lower()
    text = re.sub('\[.*?\]', '', text)
    text = re.sub('https?://\S+|www\.\S+', '', text)
    text = re.sub('<.*?>+', '', text)
    text = re.sub('[%s]' % re.escape(string.punctuation),
                  '', text)
    text = re.sub('\n', '', text)
    text = re.sub('\w*\d\w*', '', text)
    text = [word for word in text.split(' ') if word not in stopword]
    text=" ".join(text)
    text = [stemmer.stem(word) for word in text.split(' ')]
    text=" ".join(text)
    return text
data3["lyric"] = data3["lyric"].apply(clean)
```

In [41]:

```
x = np.array(data4["lyric"])
y = np.array(data4["class"])
```

In [42]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(x)
vectorizer.get_feature_names_out()
print(X.shape)
```

```
(51054, 11136)
```

In [43]:

```
first_vector = X[0]
dataframe = pd.DataFrame(first_vector.T.todense(),
                         index = vectorizer.get_feature_names(),
                         columns = ["tfidf"])
dataframe.sort_values(by = ["tfidf"],ascending=False)
```

Out[43]:

```
            tfidf
  thinkin   0.549645
    drink   0.539251
  without   0.520521
     cant   0.368994
       aa   0.000000
      ...        ...
   fragil   0.000000
 fragranc   0.000000
    frame   0.000000
    franc   0.000000
    zzzzt   0.000000

11136 rows × 1 columns
```

In [44]:

```
from sklearn.cluster import KMeans
```

In [45]:

```
wcss = []
for i in range(1,11):
    km = KMeans(n_clusters=i)
    km.fit_predict(X)
    wcss.append(km.inertia_)
```

In [46]:

```
print(km.cluster_centers_)
```

```
[[8.14320124e-06 1.67517257e-05 2.18649829e-05 ... 1.13025248e-05
  1.07740573e-05 1.83056385e-05]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
  0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
  0.00000000e+00 0.00000000e+00]
 ...
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
  0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
  0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
  0.00000000e+00 0.00000000e+00]]
```

In [47]:

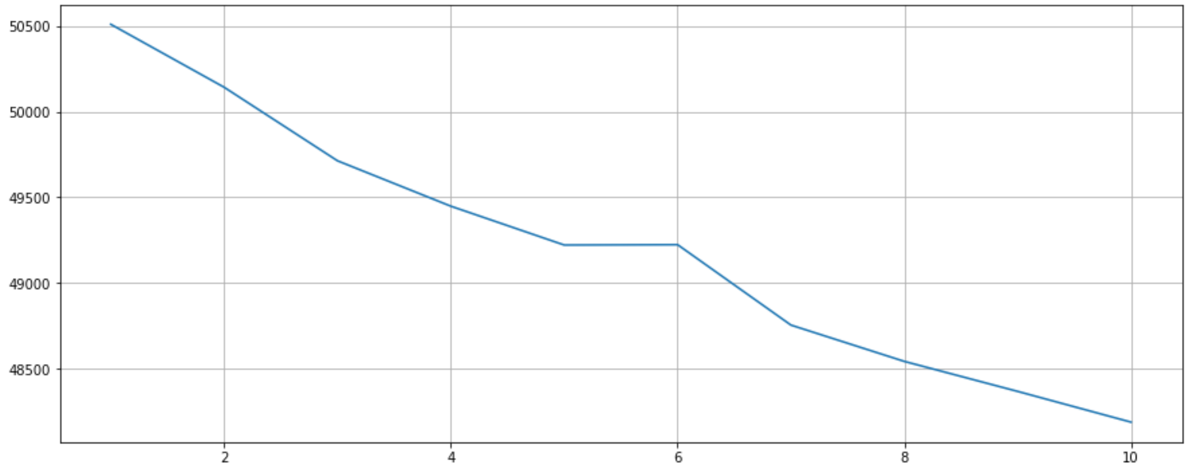
```
wcss
```

Out[47]:

```
[50509.854628753885,
 50142.31736892516,
 49713.04879461128,
 49448.00929112783,
 49221.40839457566,
 49223.37247176912,
 48754.03453853609,
 48542.10655473893,
 48366.90544721671,
 48187.46843658385]
```

In [48]:

```
plt.figure(figsize=(15,6))
plt.plot(range(1,11),wcss)
plt.grid()
plt.xticks(rotation = 0)
plt.show()
```



In [49]:

```
X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y,
                                                    test_size=0.33,
                                                    random_state=42)
```

In [50]:

```
clf = DecisionTreeClassifier()
clf.fit(X_train,y_train)
```

Out[50]:

```
DecisionTreeClassifier()
```

In [51]:

```
y_pred = clf.predict(X_test)
```

In [52]:

```
print("Training Accuracy :", clf.score(X_train, y_train))
print("Testing Accuracy :", clf.score(X_test, y_test))
```

```
Training Accuracy : 0.9926036367888674
Testing Accuracy : 0.8073361823361823
```

In [53]:

```
from sklearn import metrics
from sklearn.metrics import accuracy_score
```

In [57]:

```
print("\n Classification report for classifier %s:\n%s\n" % (clf,
                                                              metrics.classification
```

```

 Classification report for classifier DecisionTreeClassifier():
              precision    recall  f1-score   support

           0       0.83      0.83      0.83      9533
           1       0.78      0.78      0.78      7315

    accuracy                           0.81     16848
   macro avg       0.80      0.80      0.80     16848
weighted avg       0.81      0.81      0.81     16848
```

In [58]:

```python
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
```

In [59]:

```python
matrix = confusion_matrix(y_test,y_pred)
print('Confusion matrix : \n',matrix)
```

```
Confusion matrix :
 [[7877 1656]
 [1590 5725]]
```

In [60]:

```python
from sklearn.linear_model import LogisticRegression
```

In [61]:

```python
model = LogisticRegression()
model.fit(X_train, y_train)
```

Out[61]:

```
LogisticRegression()
```

In [62]:

```python
y_pred = model.predict(X_test)
```

In [63]:

```python
print("Training Accuracy :", model.score(X_train, y_train))
print("Testing Accuracy :", model.score(X_test, y_test))
```

```
Training Accuracy : 0.809477869379641
Testing Accuracy : 0.7647792022792023
```

In [64]:

```
print("\n Classification report for classifier %s:\n%s\n" % (model,
                                                              metrics.classification_
```

```

 Classification report for classifier LogisticRegression():
              precision    recall  f1-score   support

           0       0.78      0.82      0.80      9533
           1       0.74      0.70      0.72      7315

    accuracy                           0.76     16848
   macro avg       0.76      0.76      0.76     16848
weighted avg       0.76      0.76      0.76     16848

```

In [65]:

```
matrix = confusion_matrix(y_test,y_pred)
print('Confusion matrix : \n',matrix)
```

```
Confusion matrix :
 [[7781 1752]
 [2211 5104]]
```

In [66]:

```
from sklearn.svm import LinearSVC
```

In [67]:

```
LSVCClf = LinearSVC(dual = False, random_state = 0,
                    penalty = 'l1',tol = 1e-5)
LSVCClf.fit(X_train, y_train)
```

Out[67]:

```
LinearSVC(dual=False, penalty='l1', random_state=0, tol=1e-05)
```

In [68]:

```
y_pred = LSVCClf.predict(X_test)
```

In [69]:

```
print("Training Accuracy :", LSVCClf.score(X_train, y_train))
print("Testing Accuracy :", LSVCClf.score(X_test, y_test))
```

```
Training Accuracy : 0.8418113781207975
Testing Accuracy : 0.7680436847103513
```

In [70]:

```
print("\n Classification report for classifier %s:\n%s\n" % (LSVCClf,
                                                              metrics.classification
```

```
 Classification report for classifier LinearSVC(dual=False, penalty='l1',
random_state=0, tol=1e-05):
              precision    recall  f1-score   support

           0       0.81      0.77      0.79      9533
           1       0.72      0.77      0.74      7315

    accuracy                           0.77     16848
   macro avg       0.76      0.77      0.77     16848
weighted avg       0.77      0.77      0.77     16848
```

In [71]:

```
matrix = confusion_matrix(y_test,y_pred)
print('Confusion matrix : \n',matrix)
```

```
Confusion matrix :
 [[7330 2203]
 [1705 5610]]
```

In [72]:

```
from sklearn.ensemble import GradientBoostingClassifier
```

In [73]:

```
GB=GradientBoostingClassifier(n_estimators=2)
GB.fit(X_train, y_train)
```

Out[73]:

```
GradientBoostingClassifier(n_estimators=2)
```

In [74]:

```
y_pred = GB.predict(X_test)
```

In [75]:

```
print("Training Accuracy :", GB.score(X_train, y_train))
print("Testing Accuracy :", GB.score(X_test, y_test))
```

```
Training Accuracy : 0.5660995147050225
Testing Accuracy : 0.5661206077872745
```

In [76]:

```
print("\n Classification report for classifier %s:\n%s\n" % (GB,
                                                               metrics.classification
```

```
 Classification report for classifier GradientBoostingClassifier(n_estimat
ors=2):
              precision    recall  f1-score   support

           0       0.57      1.00      0.72      9533
           1       1.00      0.00      0.00      7315

    accuracy                           0.57     16848
   macro avg       0.78      0.50      0.36     16848
weighted avg       0.75      0.57      0.41     16848
```

In [77]:

```
matrix = confusion_matrix(y_test,y_pred)
print('Confusion matrix : \n',matrix)
```

```
Confusion matrix :
 [[9533    0]
 [7310    5]]
```

In [78]:

```
from sklearn.ensemble import RandomForestClassifier
```

In [79]:

```
rf_classifier= RandomForestClassifier(n_estimators= 10,
                                      criterion="entropy")
rf_classifier.fit(X_train, y_train)
```

Out[79]:

```
RandomForestClassifier(criterion='entropy', n_estimators=10)
```

In [80]:

```
y_pred = rf_classifier.predict(X_test)
```

In [81]:

```
print("Training Accuracy :", rf_classifier.score(X_train, y_train))
print("Testing Accuracy :", rf_classifier.score(X_test, y_test))
```

```
Training Accuracy : 0.9826930947786938
Testing Accuracy : 0.8360636277302944
```

In [84]:

```
print("\n Classification report for classifier %s:\n%s\n" % (rf_classifier,
                                                             metrics.classification_
```

```
 Classification report for classifier RandomForestClassifier(criterion='en
tropy', n_estimators=10):
              precision    recall  f1-score   support

           0       0.84      0.88      0.86      9533
           1       0.84      0.77      0.80      7315

    accuracy                           0.84     16848
   macro avg       0.84      0.83      0.83     16848
weighted avg       0.84      0.84      0.84     16848
```

In [85]:

```
matrix = confusion_matrix(y_test,y_pred)
print('Confusion matrix : \n',matrix)
```

```
Confusion matrix :
 [[8435 1098]
 [1664 5651]]
```