

In [1]:

```
1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 %matplotlib inline
5 import seaborn as sns
6 from IPython import get_ipython
7 import warnings
8 warnings.filterwarnings("ignore")
```

In [2]:

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

In [3]:

```
1 data1.head()
```

Out[3]:

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

In [4]:

```
1 data1.tail()
```

Out[4]:

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

In [5]:



```
1 data1.shape
```

Out[5]:

```
(2676, 2)
```

In [6]:



```
1 data1.columns
```

Out[6]:

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

In [7]:



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

Out[7]:

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

In [8]:



```
1 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]:



```
1 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]:



```
1 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]:



```
1 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]:



```
1 data2.shape
```

Out[12]:

```
(51054, 2)
```

In [13]:



```
1 data2.columns
```

Out[13]:

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

In [14]:



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

Out[14]:

```
15318
```

In [15]:



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

Out[15]:

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

In [16]:



```
1 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]:



```
1 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]:



```
1 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]:



```
1 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]:



```
1 data3.shape
```

Out[20]:

```
(2676, 2)
```

In [21]:



```
1 data3.columns
```

Out[21]:

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

In [22]:



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

Out[22]:

```
0
```

In [23]:



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

Out[23]:

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

In [24]:



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

Out[24]:

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

In [25]:



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

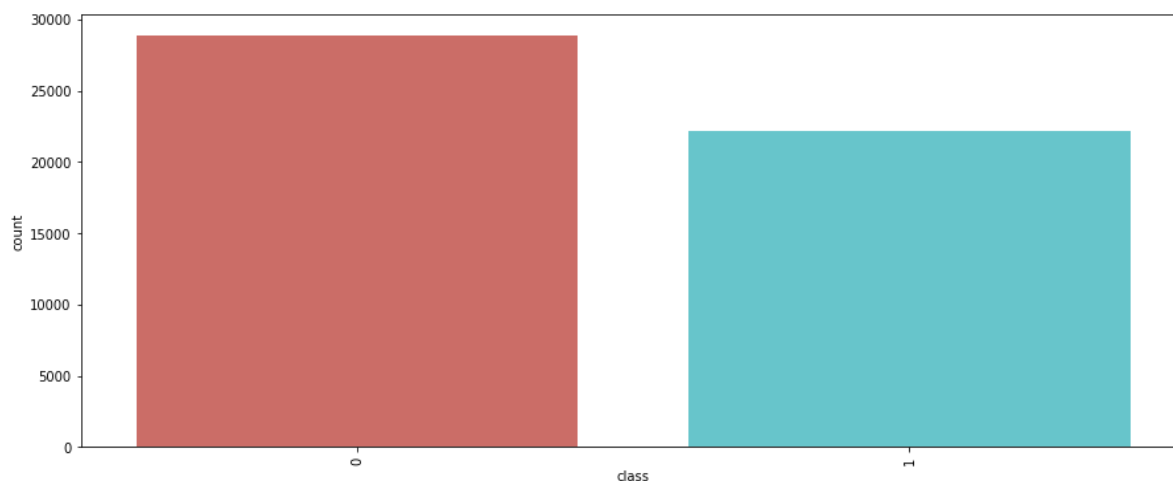
Out[25]:

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

In [26]:



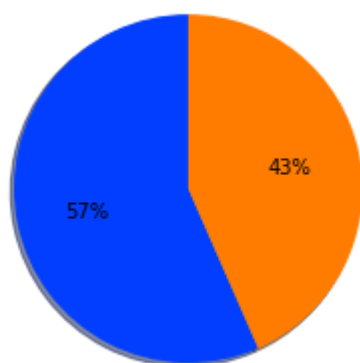
```
1 plt.figure(figsize=(15,6))
2 sns.countplot('class', data = data2, palette='hls')
3 plt.xticks(rotation = 90)
4 plt.show()
```



In [28]:



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



In [29]:



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

In [30]:



```

1 data4['total_length_characters'] = data4['lyric'].str.len()
2 print(data4['total_length_characters'])
3 total_length_characters = data4['total_length_characters'].sum()
4 print(total_length_characters)
5 count = 0
6 for y in data4["lyric"]:
7     count = count + 1
8 print(count)
9 average_length = total_length_characters / count
10 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]:



```

1 data4['total_count_words'] = data4['lyric'].str.split().str.len()
2 print(data4['total_count_words'])
3 total_words = data4['total_count_words'].sum()
4 print(total_words)
5 count = 0
6 for y in data4["lyric"]:
7     count = count + 1
8 print(count)
9 average_words = total_words / count
10 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]:



```
1 import string
2 import re
3 import nltk
4 from nltk.util import pr
5 from sklearn.feature_extraction.text import CountVectorizer
6 from sklearn.model_selection import train_test_split
7 from sklearn.tree import DecisionTreeClassifier
8 from nltk.corpus import stopwords
9 stemmer = nltk.SnowballStemmer("english")
10 nltk.download('stopwords')
11 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]:



```
1 def clean(text):
2     text = str(text).lower()
3     text = re.sub('[\.\*\?\]', '', text)
4     text = re.sub('https?://\S+|www\.\S+', '', text)
5     text = re.sub('<.*?>+', '', text)
6     text = re.sub('[%s]' % re.escape(string.punctuation),
7                 '', text)
8     text = re.sub('\n', '', text)
9     text = re.sub('\w*\d\w*', '', text)
10    text = [word for word in text.split(' ') if word not in stopword]
11    text=" ".join(text)
12    text = [stemmer.stem(word) for word in text.split(' ')]
13    text=" ".join(text)
14    return text
15    data4["lyric"] = data4["lyric"].apply(clean)
```

In [37]:



```

1 data4['total_length_characters'] = data4['lyric'].str.len()
2 print(data4['total_length_characters'])
3 total_length_characters = data4['total_length_characters'].sum()
4 print(total_length_characters)
5 count = 0
6 for y in data4["lyric"]:
7     count = count + 1
8 print(count)
9 average_length = total_length_characters / count
10 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]:



```

1 data4['total_count_words'] = data4['lyric'].str.split().str.len()
2 print(data4['total_count_words'])
3 total_words = data4['total_count_words'].sum()
4 print(total_words)
5 count = 0
6 for y in data4["lyric"]:
7     count = count + 1
8 print(count)
9 average_words = total_words / count
10 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]:



```
1 def clean(text):
2     text = str(text).lower()
3     text = re.sub('[\.\*\?\']', '', text)
4     text = re.sub('https?://\S+|www\.\S+', '', text)
5     text = re.sub('<.*?>+', '', text)
6     text = re.sub('[%s]' % re.escape(string.punctuation),
7                 '', text)
8     text = re.sub('\n', '', text)
9     text = re.sub('\w*\d\w*', '', text)
10    text = [word for word in text.split(' ') if word not in stopwords]
11    text=" ".join(text)
12    text = [stemmer.stem(word) for word in text.split(' ')]
13    text=" ".join(text)
14    return text
15 data3["lyric"] = data3["lyric"].apply(clean)
```

In [41]:



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

In [42]:



```
1 from sklearn.feature_extraction.text import TfidfVectorizer
2 vectorizer = TfidfVectorizer()
3 X = vectorizer.fit_transform(x)
4 vectorizer.get_feature_names_out()
5 print(X.shape)
```

(51054, 11136)

In [43]:



```

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

```

Out[43]:

	tfidf
<b>thinkin</b>	0.549645
<b>drink</b>	0.539251
<b>without</b>	0.520521
<b>cant</b>	0.368994
<b>aa</b>	0.000000
...	...
<b>fragil</b>	0.000000
<b>fragranc</b>	0.000000
<b>frame</b>	0.000000
<b>franc</b>	0.000000
<b>zzzzt</b>	0.000000

11136 rows × 1 columns

In [44]:



```

1 from sklearn.cluster import KMeans

```

In [45]:



```

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

```

In [46]:



```
1 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]:



```
1 wcss
```

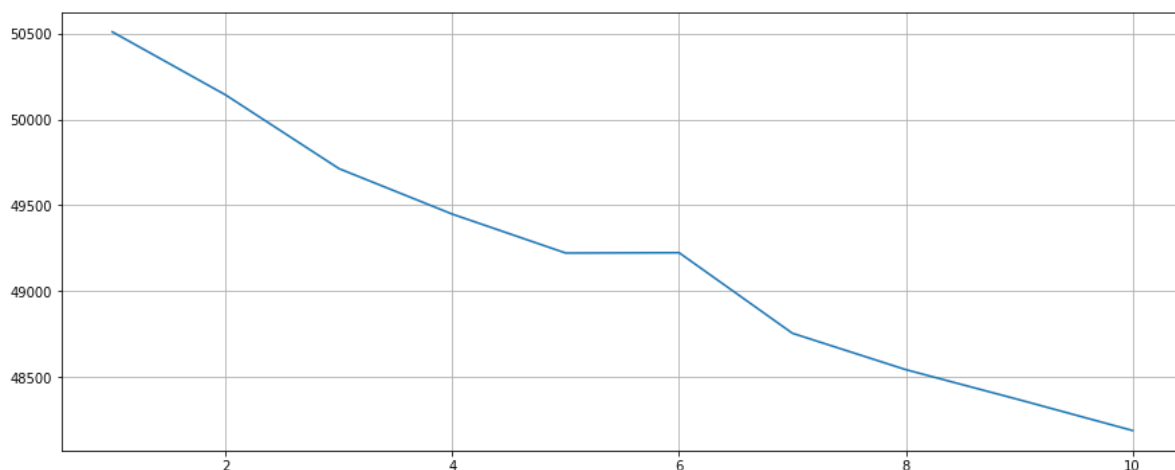
Out[47]:

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

In [48]:



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



In [49]:

```

1 X_train, X_test, y_train, y_test = train_test_split(X,
2                                                    y,
3                                                    test_size=0.33,
4                                                    random_state=42)

```

In [50]:

```

1 clf = DecisionTreeClassifier()
2 clf.fit(X_train,y_train)

```

Out[50]:

DecisionTreeClassifier()

In [51]:

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

In [52]:

```

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

```

Training Accuracy : 0.9926036367888674

Testing Accuracy : 0.8073361823361823

In [53]:

```

1 from sklearn import metrics
2 from sklearn.metrics import accuracy_score

```

In [57]:

```

1 print("\n Classification report for classifier %s:\n%s\n" % (clf,
2                                                            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]:



```
1 from sklearn.metrics import confusion_matrix
2 from sklearn.metrics import classification_report
```

In [59]:



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

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

In [60]:



```
1 from sklearn.linear_model import LogisticRegression
```

In [61]:



```
1 model = LogisticRegression()
2 model.fit(X_train, y_train)
```

Out[61]:

```
LogisticRegression()
```

In [62]:



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

In [63]:



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

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

In [64]:

```
1 print("\n Classification report for classifier %s:\n%s\n" % (model,
2                                     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]:

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

Confusion matrix :

```
[[7781 1752]
 [2211 5104]]
```

In [66]:

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

In [67]:

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

Out[67]:

LinearSVC(dual=False, penalty='l1', random\_state=0, tol=1e-05)

In [68]:

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

In [69]:

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

Training Accuracy : 0.8418113781207975

Testing Accuracy : 0.7680436847103513



In [70]:

```
1 print("\n Classification report for classifier %s:\n%s\n" % (LSVC1f,
2                                     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]:

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

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

In [72]:

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

In [73]:

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

Out[73]:

GradientBoostingClassifier(n\_estimators=2)

In [74]:

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

In [75]:

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

Training Accuracy : 0.5660995147050225  
Testing Accuracy : 0.5661206077872745

In [76]:

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

Classification report for classifier GradientBoostingClassifier(n\_estimators=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]:

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

Confusion matrix :

```
[[9533  0]
 [7310  5]]
```

In [78]:

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

In [79]:

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

Out[79]:

RandomForestClassifier(criterion='entropy', n\_estimators=10)

In [80]:

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

In [81]:

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

Training Accuracy : 0.9826930947786938

Testing Accuracy : 0.8360636277302944

In [84]:



```
1 print("\n Classification report for classifier %s:\n%s\n" % (rf_classifier,
2 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]:



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

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