I have structured the project into 5 key sections. Below, I will share with you the steps I took together with insights for each section.

Note: Click on the links to go to the respective section

1. Data Understanding

```
#import all modules
import pandas as pd

import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

from sklearn.preprocessing import LabelEncoder
from scipy import stats
# Tensorflow and Keras are two packages for creating neural network
models.
import tensorflow as tf
from tensorflow import keras
from sklearn.model_selection import train_test_split

# import NN layers and other componenets.
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Activation, Dense,
BatchNormalization, Dropout
from tensorflow.keras import optimizers
```

```python
import matplotlib.pyplot as plt # for plotting data and creating
different charts.
import numpy as np # for math and arrays
import pandas as pd # data from for the data.
import seaborn as sns # for plotting.
from sklearn import preprocessing
from sklearn import linear_model
from sklearn import datasets

df=pd.read_csv('Data2021.csv')
df
```

```
                 Artist Name                              Track Name  \
0                 Bruno Mars  That's What I Like (feat. Gucci Mane)
1                     Boston                           Hitch a Ride
2              The Raincoats                        No Side to Fall In
3                       Deno                Lingo (feat. J.I & Chunkz)
4      Red Hot Chili Peppers      Nobody Weird Like Me - Remastered
...                      ...                                     ...
17991            Green-House                               Find Home
17992               Micatone                                All Gone
17993         Smash Hit Combo                           Peine perdue
17994                Beherit                          Salomon's Gate
17995         The Raconteurs                       Broken Boy Soldier

       Popularity  danceability  energy   key  loudness  mode
speechiness  \
0            60.0         0.854   0.564   1.0    -4.964     1
0.0485
1            54.0         0.382   0.814   3.0    -7.230     1
0.0406
2            35.0         0.434   0.614   6.0    -8.334     1
0.0525
3            66.0         0.853   0.597  10.0    -6.528     0
0.0555
4            53.0         0.167   0.975   2.0    -4.279     1
0.2160
...           ...           ...     ...   ...       ...   ...
...
17991        35.0         0.166   0.109   7.0   -17.100     0
0.0413
17992        27.0         0.638   0.223  11.0   -10.174     0
0.0329
17993        34.0         0.558   0.981   4.0    -4.683     0
0.0712
17994        29.0         0.215   0.805   6.0   -12.757     0
0.1340
17995        43.0         0.400   0.853   4.0    -5.320     0
0.0591
```

```
       acousticness  instrumentalness  liveness  valence    tempo  \
0          0.017100               NaN    0.0849   0.8990  134.071
1          0.001100          0.004010    0.1010   0.5690  116.454
2          0.486000          0.000196    0.3940   0.7870  147.681
3          0.021200               NaN    0.1220   0.5690  107.033
4          0.000169          0.016100    0.1720   0.0918  199.060
...             ...               ...       ...      ...      ...
17991      0.993000          0.824000    0.0984   0.1770  171.587
17992      0.858000          0.000016    0.0705   0.3350   73.016
17993      0.000030          0.000136    0.6660   0.2620  105.000
17994      0.001290          0.916000    0.2560   0.3550  131.363
17995      0.006040          0.212000    0.3340   0.3770  138.102

       duration_in min/ms  time_signature  Class
0                234596.0               4      5
1                251733.0               4     10
2                109667.0               4      6
3                173968.0               4      5
4                229960.0               4     10
...                   ...             ...    ...
17991            193450.0               3      6
17992            257067.0               4      2
17993            216222.0               4      8
17994            219693.0               4      8
17995            182227.0               4     10

[17996 rows x 17 columns]
```

```
df.dtypes
```

```
Artist Name            object
Track Name             object
Popularity            float64
danceability          float64
energy                float64
key                   float64
loudness              float64
mode                    int64
speechiness           float64
acousticness          float64
instrumentalness      float64
liveness              float64
valence               float64
tempo                 float64
duration_in min/ms    float64
time_signature          int64
Class                   int64
dtype: object
```

```
df.shape
```

```
(17996, 17)
```

This dataset consists of 13 music charcteristics that is used to predict the target variable:

As for the target variable we are going to predict:

```
df["Class"].unique()

array([ 5, 10,  6,  2,  4,  8,  9,  3,  7,  1,  0], dtype=int64)
```

There are in total 11 unique classes that represents different type of music such as Folk, Blue, HipHop, Metal Po

There are 2 other rows - Artist Name and Track Name that is a unique identifier of each row.

2. Data Preparation

```
print("Rows, columns: " + str(df.shape))

Rows, columns: (17996, 17)

df.isnull().sum()

Artist Name              0
Track Name               0
Popularity             428
danceability             0
energy                   0
key                   2014
loudness                 0
mode                     0
speechiness              0
acousticness             0
instrumentalness      4377
liveness                 0
valence                  0
tempo                    0
duration_in min/ms       0
time_signature           0
Class                    0
dtype: int64
```

2.1 Replacing Missing Values

I chose not to drop the null values and instead replace it with KNN imputation as total number of missing values for popularity,instrumentalness and key were ranging from 400 to 4300. If I were to drop all it will affect our dataset as we do not have much information on these 3 music characteristics to predict the class

```python
# Import the INNImputer
from sklearn.impute import KNNImputer

# The dfKNN data frame will only have the 3 columns with missing
values
dfKNN = df[['Popularity',"instrumentalness","key"]]

# Create a kNNImputer object and set k=1
imputer = KNNImputer(n_neighbors=5)

# We use the fit_transform() method to perform the imputation
# We also create another DataFrame from the results returned
# by the fit_transform function
dfKNN = pd.DataFrame(imputer.fit_transform(dfKNN))

# We merge back with the rest of the columns (Year and Method)
df = pd.concat([df[["Artist Name","Track
Name","danceability","energy","loudness","mode",

"speechiness","acousticness","liveness","valence","tempo","duration_in
min/ms",
                    "time_signature","Class"]], dfKNN], axis=1,
join="inner")
df
```

```
                 Artist Name                               Track Name  \
0                 Bruno Mars  That's What I Like (feat. Gucci Mane)
1                     Boston                             Hitch a Ride
2              The Raincoats                          No Side to Fall In
3                       Deno                Lingo (feat. J.I & Chunkz)
4      Red Hot Chili Peppers          Nobody Weird Like Me - Remastered
...                      ...                                      ...
17991            Green-House                                 Find Home
17992               Micatone                                  All Gone
17993        Smash Hit Combo                              Peine perdue
17994                Beherit                            Salomon's Gate
17995         The Raconteurs                         Broken Boy Soldier

       danceability  energy  loudness  mode  speechiness  acousticness
\
0             0.854   0.564    -4.964     1       0.0485      0.017100

1             0.382   0.814    -7.230     1       0.0406      0.001100

2             0.434   0.614    -8.334     1       0.0525      0.486000

3             0.853   0.597    -6.528     0       0.0555      0.021200

4             0.167   0.975    -4.279     1       0.2160      0.000169
```

```
...                ...     ...        ...   ...          ...           ...
17991            0.166   0.109    -17.100     0       0.0413      0.993000
17992            0.638   0.223    -10.174     0       0.0329      0.858000
17993            0.558   0.981     -4.683     0       0.0712      0.000030
17994            0.215   0.805    -12.757     0       0.1340      0.001290
17995            0.400   0.853     -5.320     0       0.0591      0.006040

       liveness  valence    tempo  duration_in min/ms  time_signature
Class  \
0        0.0849   0.8990  134.071            234596.0               4
5
1        0.1010   0.5690  116.454            251733.0               4
10
2        0.3940   0.7870  147.681            109667.0               4
6
3        0.1220   0.5690  107.033            173968.0               4
5
4        0.1720   0.0918  199.060            229960.0               4
10
...         ...      ...      ...                 ...             ...
...
17991    0.0984   0.1770  171.587            193450.0               3
6
17992    0.0705   0.3350   73.016            257067.0               4
2
17993    0.6660   0.2620  105.000            216222.0               4
8
17994    0.2560   0.3550  131.363            219693.0               4
8
17995    0.3340   0.3770  138.102            182227.0               4
10

          0         1     2
0      60.0  0.251237   1.0
1      54.0  0.004010   3.0
2      35.0  0.000196   6.0
3      66.0  0.057497  10.0
4      53.0  0.016100   2.0
...     ...       ...   ...
17991  35.0  0.824000   7.0
17992  27.0  0.000016  11.0
17993  34.0  0.000136   4.0
17994  29.0  0.916000   6.0
17995  43.0  0.212000   4.0
```

```
[17996 rows x 17 columns]

df.rename(columns={0: "Popularity", 1:
"instrumentalness",2:"key"},inplace=True)

df.isnull().sum()

Artist Name           0
Track Name            0
danceability          0
energy                0
loudness              0
mode                  0
speechiness           0
acousticness          0
liveness              0
valence               0
tempo                 0
duration_in min/ms    0
time_signature        0
Class                 0
Popularity            0
instrumentalness      0
key                   0
dtype: int64
```

2.2 Removing duplicates

```
df.shape

(17996, 17)

print("Number of duplicates in whole
dataset:",df.duplicated(keep='first').sum())

Number of duplicates in whole dataset: 0
```

Looks like there are no duplicates. However, as I scanned through the whole dataset, I realised that there are rows that have duplicated values across all columns except for Class. This would mean that there are 2 exactly songs having the same music characteristic but they of 2 different classes. Hence, let us drop data with similar values across all columns except for class

```
print("Number of duplicates across all columns(except for
class):",df[['Artist Name', 'Track Name', 'danceability', 'energy',
'loudness',
       'mode', 'speechiness', 'acousticness', 'liveness', 'valence',
'tempo',
       'duration_in min/ms', 'time_signature', 'Popularity',
       'instrumentalness', 'key']].duplicated(keep='first').sum())
```

```
Number of duplicates across all columns(except for class): 1673

df_no_class=df[['Artist Name', 'Track Name', 'danceability', 'energy',
'loudness',
       'mode', 'speechiness', 'acousticness', 'liveness', 'valence',
'tempo',
       'duration_in min/ms', 'time_signature', 'Popularity',
       'instrumentalness', 'key']].drop_duplicates(keep=False)

 df=df[df.index.isin(df_no_class.index)]

df.shape

(14838, 17)
```

Now, that we have dropped duplicates, we are left with 14838 rows.

2.3 Converting duration from ms to minutes

After scanning through the dataset, I realised that the longest duration is minutes is 30. Hence, I will be converting the values above 30 from miliseconds to minutes.

```
def convert_to_min(time):
    if time>30:
        return time/(1000*60)
    else:
        return time

df['duration_in_min']=df['duration_in min/ms'].apply(convert_to_min)

<ipython-input-17-09a8aa8de0ae>:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  df['duration_in_min']=df['duration_in min/ms'].apply(convert_to_min)

df.drop(['duration_in min/ms'],axis=1,inplace=True)

C:\Users\JiaYi\anaconda3\lib\site-packages\pandas\core\frame.py:4308:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  return super().drop(
```

2.4 Removing outliers

```
###Use a new dataframe for numerical columns
df1=df[["danceability","energy","loudness","mode","speechiness","acous
ticness","liveness","valence","tempo","duration_in_min","time_signatur
e","instrumentalness","key","Popularity","Class"]]
z_scores = stats.zscore(df1)
abs_z_scores = np.abs(z_scores)
filtered_entries = (abs_z_scores < 3).all(axis=1)
print("Number of Outliers:",df.shape[0]-
np.count_nonzero(filtered_entries))
```

```
Number of Outliers: 1354
```

```
df= df[filtered_entries]
df
```

```
                 Artist Name                                  Track Name  \
0                 Bruno Mars  That's What I Like (feat. Gucci Mane)
1                     Boston                                Hitch a Ride
2              The Raincoats                          No Side to Fall In
3                       Deno                Lingo (feat. J.I & Chunkz)
4       Red Hot Chili Peppers          Nobody Weird Like Me - Remastered
...                      ...                                         ...
17991            Green-House                                   Find Home
17992               Micatone                                    All Gone
17993         Smash Hit Combo                               Peine perdue
17994                Beherit                              Salomon's Gate
17995          The Raconteurs                         Broken Boy Soldier

       danceability  energy  loudness  mode  speechiness  acousticness
\
0             0.854   0.564    -4.964     1       0.0485      0.017100

1             0.382   0.814    -7.230     1       0.0406      0.001100

2             0.434   0.614    -8.334     1       0.0525      0.486000

3             0.853   0.597    -6.528     0       0.0555      0.021200

4             0.167   0.975    -4.279     1       0.2160      0.000169

...             ...     ...       ...   ...          ...           ...

17991         0.166   0.109   -17.100     0       0.0413      0.993000

17992         0.638   0.223   -10.174     0       0.0329      0.858000

17993         0.558   0.981    -4.683     0       0.0712      0.000030

17994         0.215   0.805   -12.757     0       0.1340      0.001290
```

```
17995          0.400   0.853    -5.320     0       0.0591      0.006040


       liveness  valence    tempo  time_signature  Class
Popularity  \
0        0.0849   0.8990  134.071               4      5          60.0

1        0.1010   0.5690  116.454               4     10          54.0

2        0.3940   0.7870  147.681               4      6          35.0

3        0.1220   0.5690  107.033               4      5          66.0

4        0.1720   0.0918  199.060               4     10          53.0

...         ...      ...      ...             ...    ...           ...

17991    0.0984   0.1770  171.587               3      6          35.0

17992    0.0705   0.3350   73.016               4      2          27.0

17993    0.6660   0.2620  105.000               4      8          34.0

17994    0.2560   0.3550  131.363               4      8          29.0

17995    0.3340   0.3770  138.102               4     10          43.0


       instrumentalness   key  duration_in_min
0              0.251237   1.0         3.909933
1              0.004010   3.0         4.195550
2              0.000196   6.0         1.827783
3              0.057497  10.0         2.899467
4              0.016100   2.0         3.832667
...                 ...   ...              ...
17991          0.824000   7.0         3.224167
17992          0.000016  11.0         4.284450
17993          0.000136   4.0         3.603700
17994          0.916000   6.0         3.661550
17995          0.212000   4.0         3.037117

[13484 rows x 17 columns]
```

```
df.reset_index(inplace=True)
df.drop(columns=['index'],inplace=True)
df
```

```
                 Artist Name                              Track Name  \
0                 Bruno Mars  That's What I Like (feat. Gucci Mane)
1                     Boston                            Hitch a Ride
2              The Raincoats                      No Side to Fall In
```

```
3                        Deno            Lingo (feat. J.I & Chunkz)
4      Red Hot Chili Peppers      Nobody Weird Like Me - Remastered
...                       ...                                   ...
13479             Green-House                             Find Home
13480                Micatone                              All Gone
13481         Smash Hit Combo                          Peine perdue
13482                 Beherit                        Salomon's Gate
13483         The Raconteurs                    Broken Boy Soldier

       danceability  energy  loudness  mode  speechiness  acousticness
\
0             0.854   0.564    -4.964     1       0.0485      0.017100

1             0.382   0.814    -7.230     1       0.0406      0.001100

2             0.434   0.614    -8.334     1       0.0525      0.486000

3             0.853   0.597    -6.528     0       0.0555      0.021200

4             0.167   0.975    -4.279     1       0.2160      0.000169

...             ...     ...       ...   ...          ...           ...

13479         0.166   0.109   -17.100     0       0.0413      0.993000

13480         0.638   0.223   -10.174     0       0.0329      0.858000

13481         0.558   0.981    -4.683     0       0.0712      0.000030

13482         0.215   0.805   -12.757     0       0.1340      0.001290

13483         0.400   0.853    -5.320     0       0.0591      0.006040


       liveness  valence    tempo  time_signature  Class
Popularity  \
0        0.0849   0.8990  134.071               4      5          60.0

1        0.1010   0.5690  116.454               4     10          54.0

2        0.3940   0.7870  147.681               4      6          35.0

3        0.1220   0.5690  107.033               4      5          66.0

4        0.1720   0.0918  199.060               4     10          53.0

...         ...      ...      ...             ...    ...           ...

13479    0.0984   0.1770  171.587               3      6          35.0

13480    0.0705   0.3350   73.016               4      2          27.0
```

```
13481    0.6660   0.2620  105.000                   4      8         34.0

13482    0.2560   0.3550  131.363                   4      8         29.0

13483    0.3340   0.3770  138.102                   4     10         43.0


       instrumentalness   key  duration_in_min
0              0.251237   1.0         3.909933
1              0.004010   3.0         4.195550
2              0.000196   6.0         1.827783
3              0.057497  10.0         2.899467
4              0.016100   2.0         3.832667
...                 ...   ...              ...
13479          0.824000   7.0         3.224167
13480          0.000016  11.0         4.284450
13481          0.000136   4.0         3.603700
13482          0.916000   6.0         3.661550
13483          0.212000   4.0         3.037117

[13484 rows x 17 columns]
```

Now that we have removed outliers, it will allow for a better model performance in the later stage

3.Exploratory Data Analysis

3.1 Descriptive Statistical Analysis After cleaning the data, we used the describe() function once again to look at the statistical summary of the numerical variables.

```
describe_df=df.describe()
describe_df

       danceability        energy      loudness          mode
speechiness  \
count  13484.000000  13484.000000  13484.000000  13484.000000
13484.000000
mean       0.551126      0.652734     -7.830060      0.635494
0.071960
std        0.163310      0.233155      3.594211      0.481309
0.063338
min        0.059900      0.016300    -20.892000      0.000000
0.022500
25%        0.441000      0.488000     -9.677500      0.000000
0.034300
50%        0.552000      0.684500     -7.084000      1.000000
0.045900
75%        0.666000      0.851000     -5.205000      1.000000
0.079800
```

```
max       0.989000      1.000000      0.943000      1.000000
0.344000

       acousticness      liveness       valence         tempo
time_signature  \
count  13484.000000  13484.000000  13484.000000  13484.000000
13484.000000
mean       0.265530      0.178511      0.484164    122.106857
3.938891
std        0.314791      0.122955      0.240459     29.584804
0.283498
min        0.000000      0.011900      0.018300     42.956000
3.000000
25%        0.005940      0.096900      0.292000     98.812750
4.000000
50%        0.105000      0.125000      0.475000    119.990500
4.000000
75%        0.480000      0.235000      0.670000    141.014000
4.000000
max        0.996000      0.680000      0.986000    209.905000
5.000000

              Class    Popularity  instrumentalness           key  \
count  13484.000000  13484.000000      13484.000000  13484.000000
mean       6.857980     43.952996          0.167114      5.934295
std        3.155246     17.341515          0.266135      3.074491
min        0.000000      1.000000          0.000001      1.000000
25%        5.000000     32.000000          0.000227      4.000000
50%        8.000000     43.000000          0.015972      6.000000
75%       10.000000     56.000000          0.215106      9.000000
max       10.000000     95.000000          0.996000     11.000000

       duration_in_min
count     13484.000000
mean          3.855722
std           1.080495
min           0.388667
25%           3.136667
50%           3.673175
75%           4.364504
max           8.429550
```

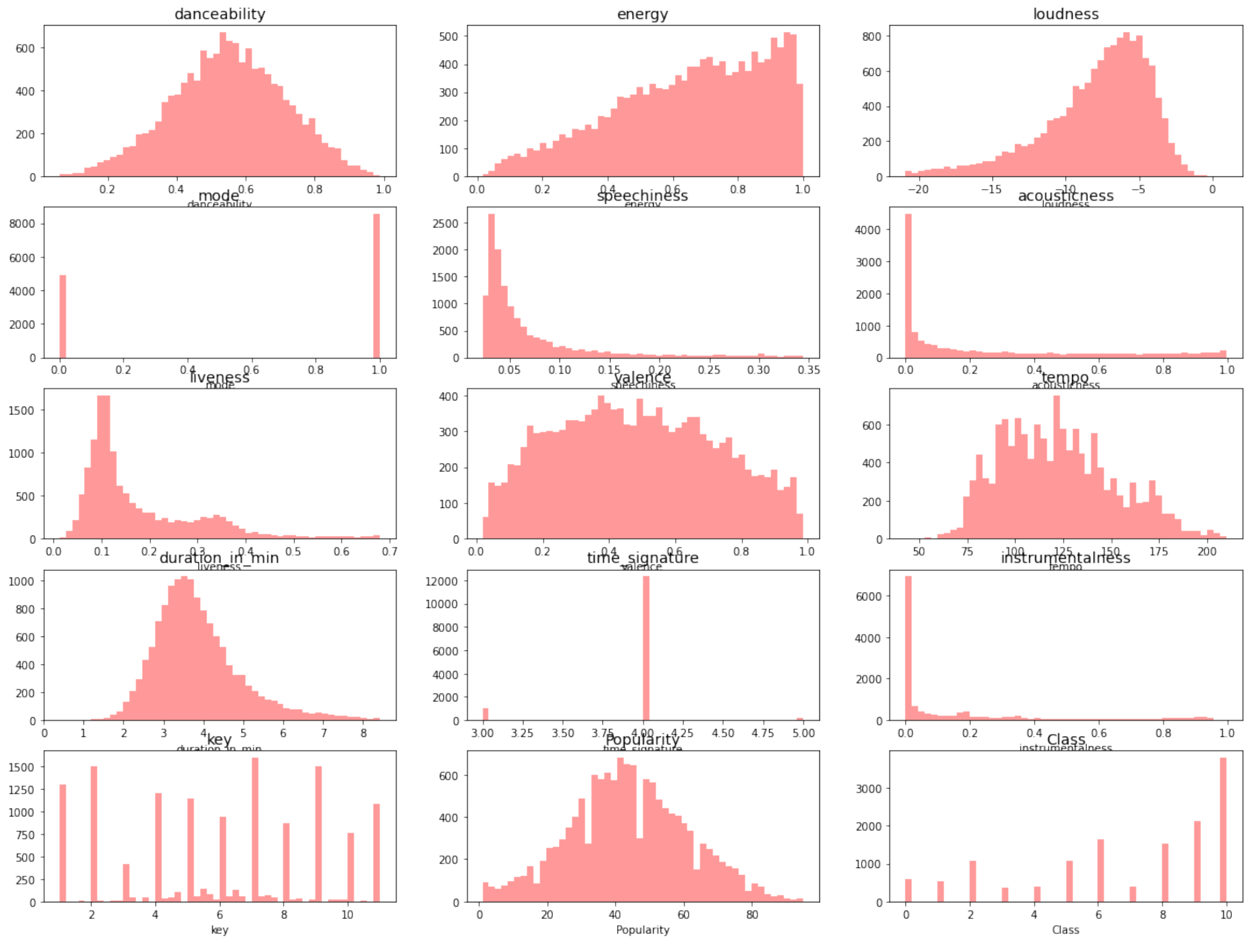
From this dataframe, we may not be able to visualize at one glance the distrbution of all chracteristics. Hence, I have decided to plot the distribution of each characteristics below

Distribution of each music characteristics

```
labels
=["danceability","energy","loudness","mode","speechiness","acousticnes
```

```
s","liveness","valence","tempo","duration_in_min","time_signature","in
strumentalness","key","Popularity","Class"]
row, col = 5, 3

fig, ax = plt.subplots(5, 3, figsize=(20,15))
for i in range(5):
    for j in range(3):
        c_type = labels[i*col+j]
        c_ax = ax[i][j]

dplot=sns.distplot(df[c_type],kde=False,color='red',norm_hist=False,bi
ns=50,ax=c_ax)
        c_ax.set_title(label=c_type,fontdict = {'fontsize': 14})
```

```
C:\Users\JiaYi\anaconda3\lib\site-packages\seaborn\
distributions.py:2557: FutureWarning: `distplot` is a deprecated
function and will be removed in a future version. Please adapt your
code to use either `displot` (a figure-level function with similar
flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

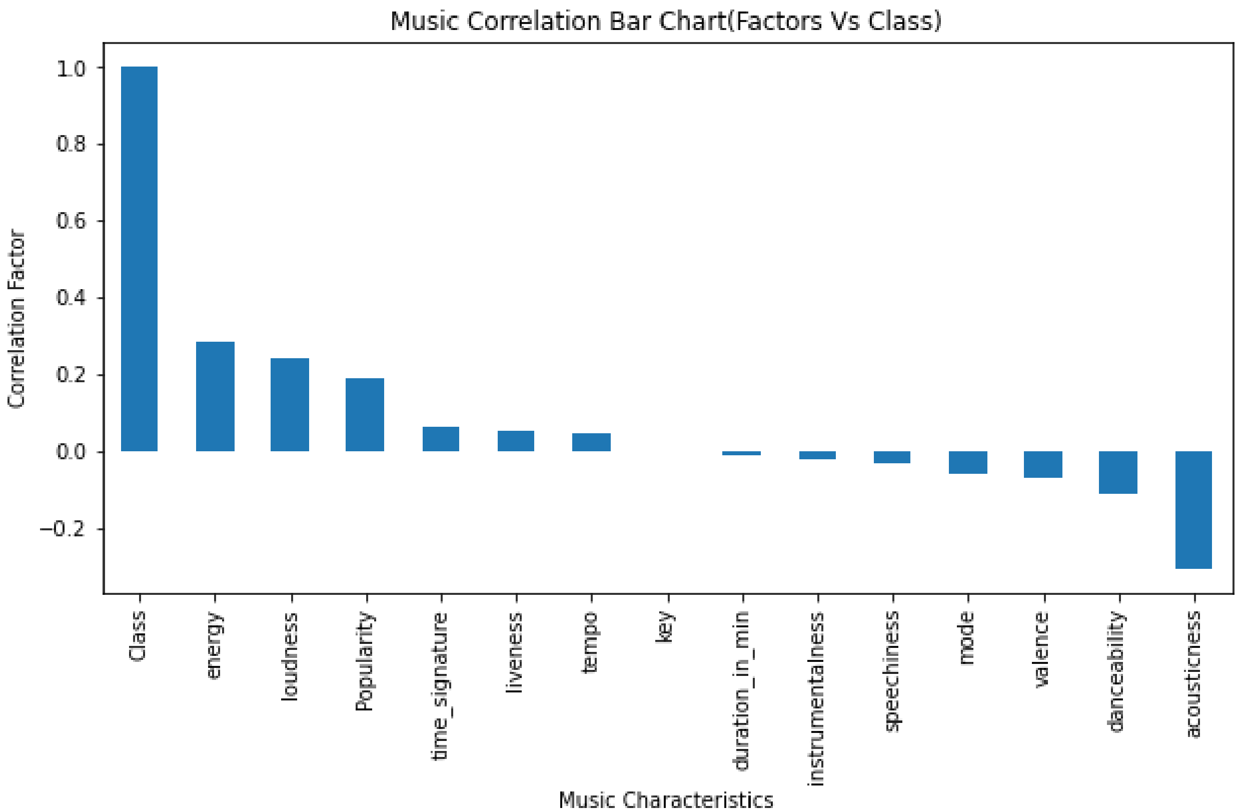


1. Danceability
2. Valence
1. Energy
2. Loudness
1. Speechiness
2. Liveness
3. Acousticness
4. Instrumentalness
5. Duration in min
1. Key
2. Tempo
1. Mode
2. Time Signature

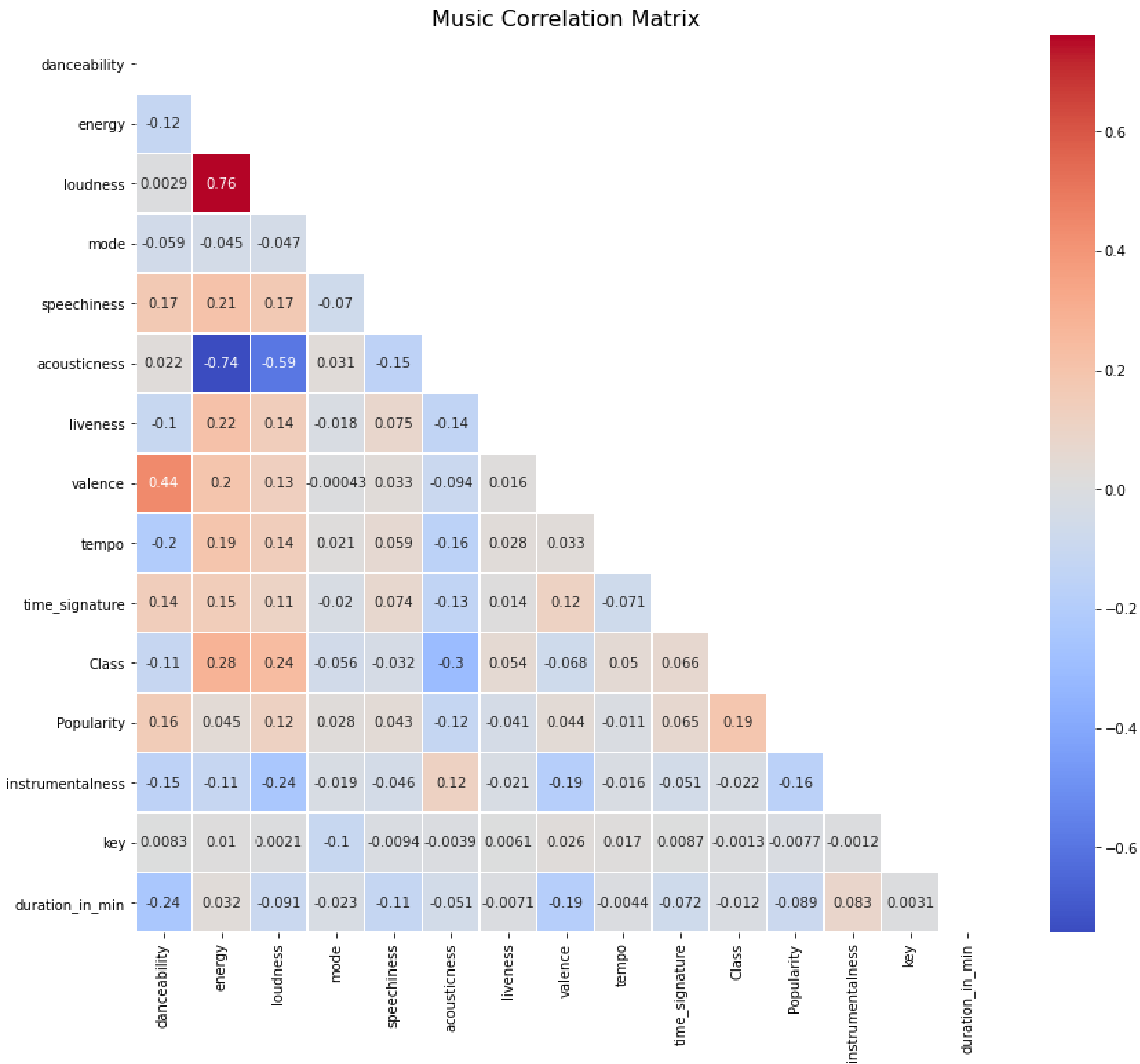
3.2 Correlation Analysis Next, we conducted a Correlation Analysis to better understand the relationship between the 13 key factors affecting the class of the music.

```
corr_mat = df.corr()
plt.figure(figsize = (10,5))
```

```
corr_mat['Class'].sort_values(ascending = False).plot(kind = 'bar');
plt.title('Music Correlation Bar Chart(Factors Vs Class)')
plt.xlabel('Music Characteristics')
plt.ylabel('Correlation Factor')
plt.show()
```




```
#data_df_corr = data_df.corr()
#data_df_corr.style.background_gradient(cmap =
'coolwarm').set_precision(2)
import numpy as np
rs = np.random.RandomState(33)

# Generate a mask for the upper triangle
mask = np.triu(np.ones_like(df.corr(), dtype=bool))

plt.figure(figsize=(14,12))
plt.title('Music Correlation Matrix', fontsize = 16)
p=sns.heatmap(df.corr(), linewidth = 0.5, annot=True,
cmap='coolwarm',mask=mask)
```



Correlation between music characteristics and class Overall correlation between the 13 variables and the Class is not very strong. However, I have identified four variables which had significant correlation with the Class

1. Acousticness Vs Class(-0.3)
2. Energy Vs Class(0.28)
3. Loudness Vs Class(0.24)

Correlation among music characteristics High Correlation exist among these music characteristics

1. Loudness Vs Energy(0.76)
2. Acousticness Vs Energy (-0.74)
3. Acousticness Vs Loudness(-0.59)

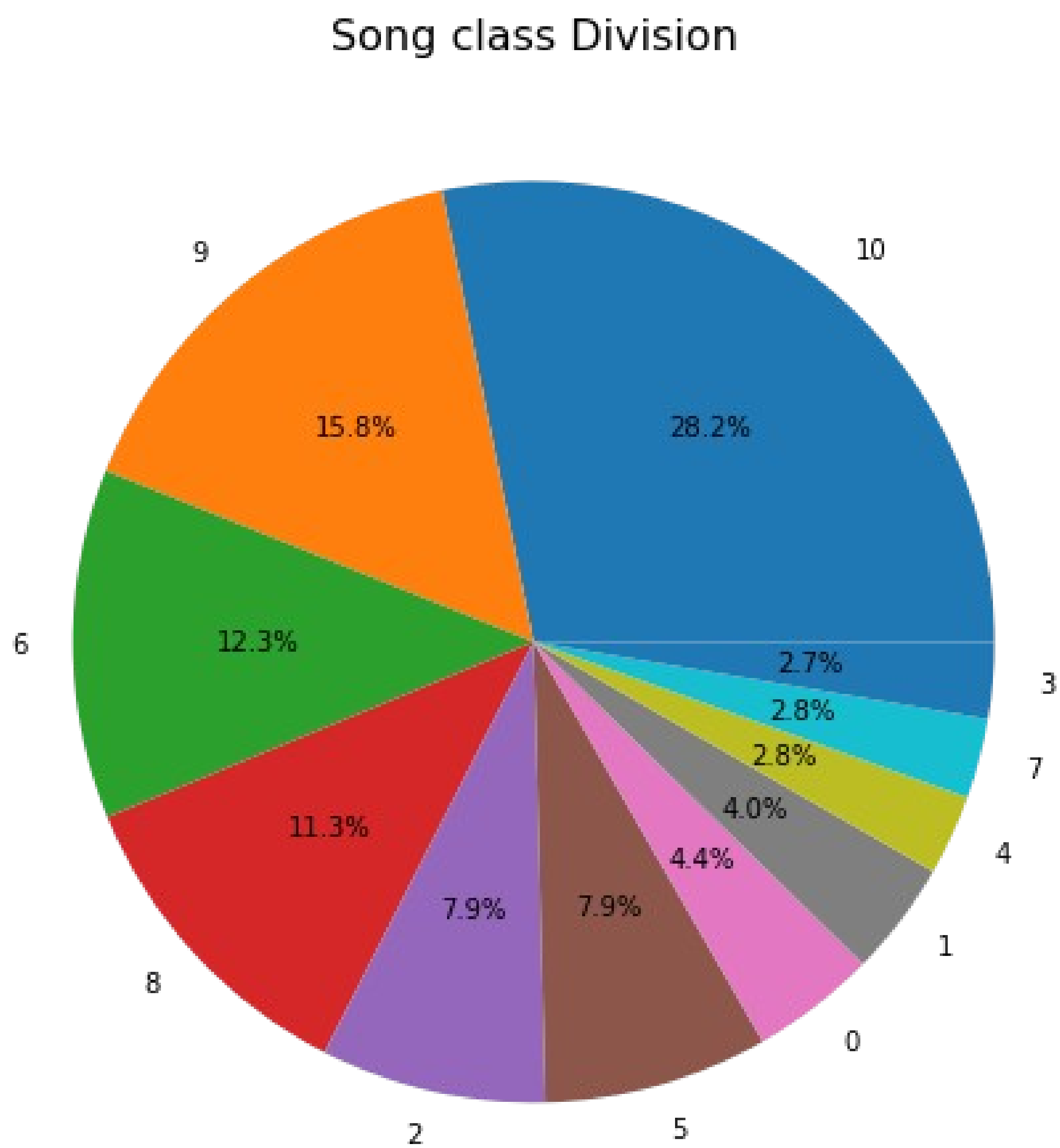
3.3 Class Analysis Next, I will be analysing the class of the music to get a better understanding of it.

Distribution and Porportion of class

```
class_count= df.Class.value_counts()
class_count
```

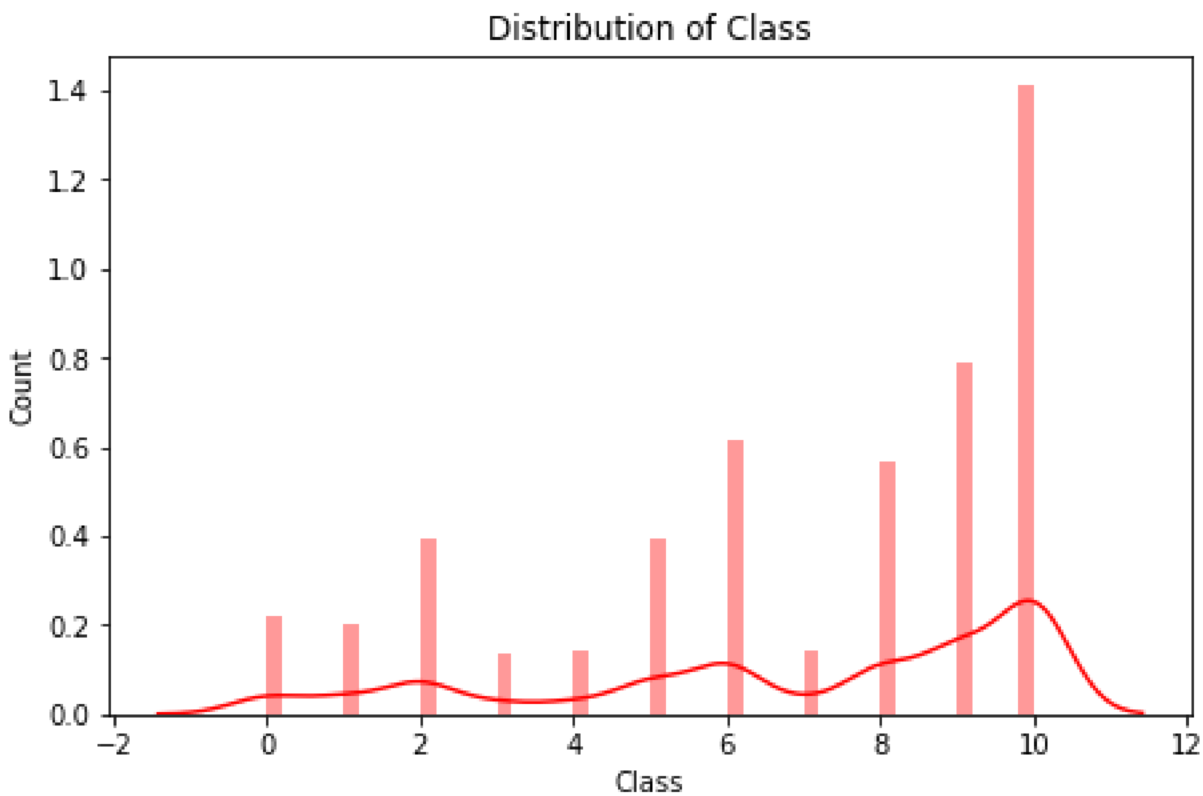
```
10    3800
9     2125
6     1654
8     1524
2     1065
5     1062
0      592
1      543
4      380
7      378
3      361
Name: Class, dtype: int64
```

```
plt.figure(figsize=(10,8))
plt.title("Song class Division",fontsize=16)
plt.pie(class_count, labels = class_count.index, autopct='%1.1f%%');
```




```
figure = plt.figure(figsize=(16,10))

# Distribution of Volatile acidity
ax1 = figure.add_subplot(2, 2, 1)
sns.distplot(df['Class'],
             norm_hist=False,
             bins=50,
             color='red',
             ax=ax1).set_title('Distribution of Class')
ax1.set_ylabel("Count")
```

```
C:\Users\JiaYi\anaconda3\lib\site-packages\seaborn\
distributions.py:2557: FutureWarning: `distplot` is a deprecated
function and will be removed in a future version. Please adapt your
code to use either `displot` (a figure-level function with similar
flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)

Text(0, 0.5, 'Count')
```



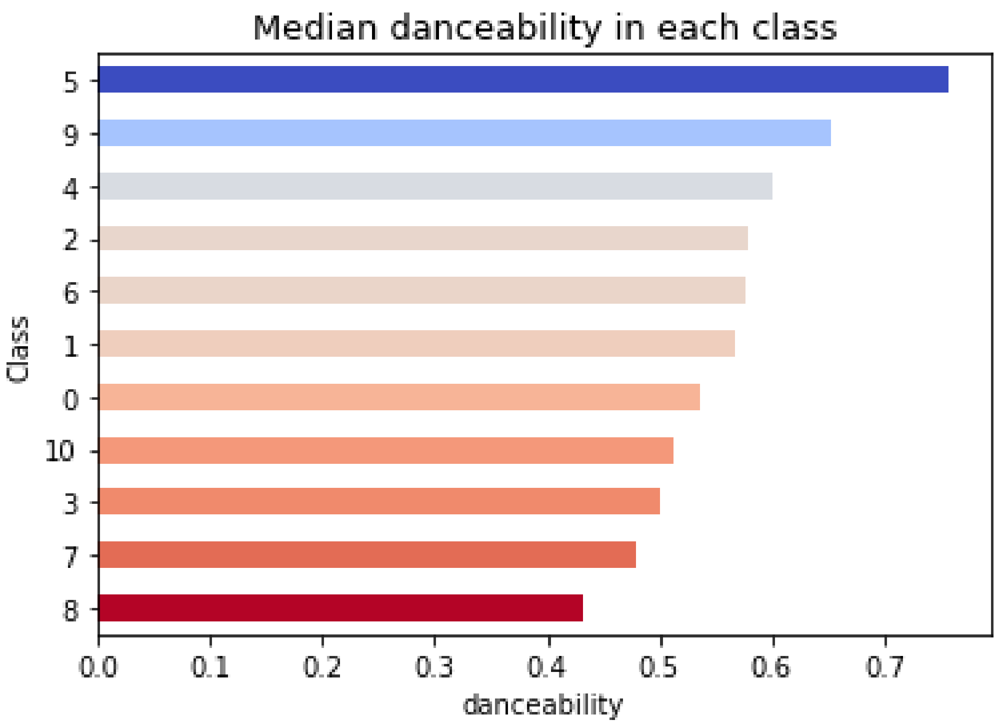
As we can see from the above pie chart, Class 10 stands the highest percentage of 28.2% followed by class 9 then 6. The classes with the lower data examples are classes 4,3,7,0. They have around 2% to 4% of data examples which is around 360 to 540.

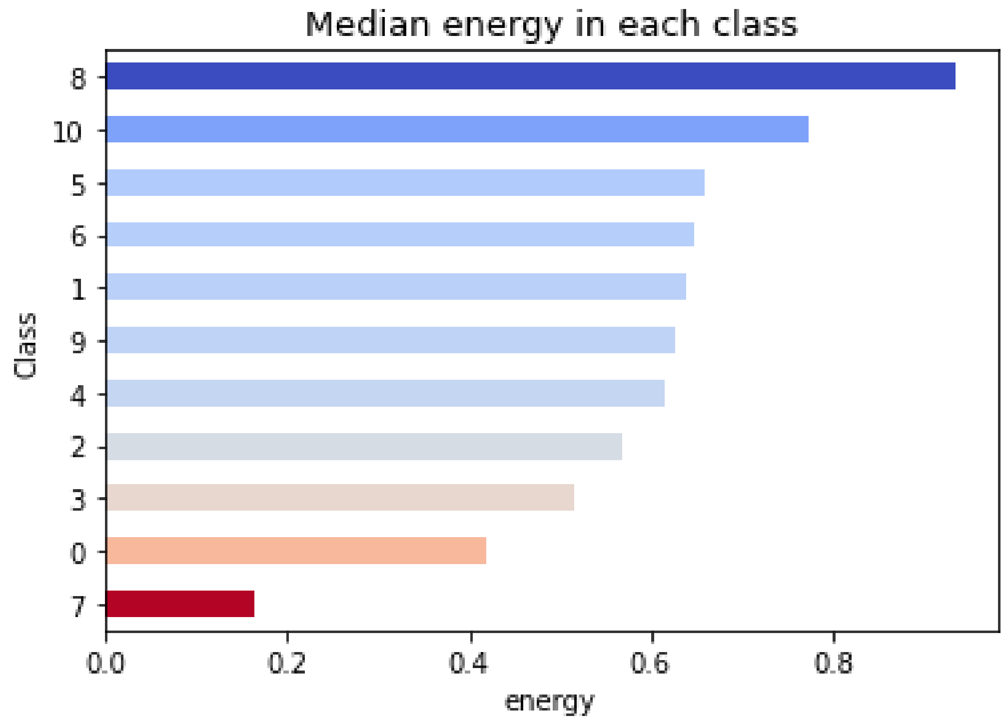
There is uneven distribution between 11 classes where Class 10 hits the highest percentage of 28.2%. There are many minority classes such as class 7,3,4,0 where their population is less than 5%. Hence, during modelling stage, I will be balancing the data.

3.4 Further Analysis of Music Charateristics and Class

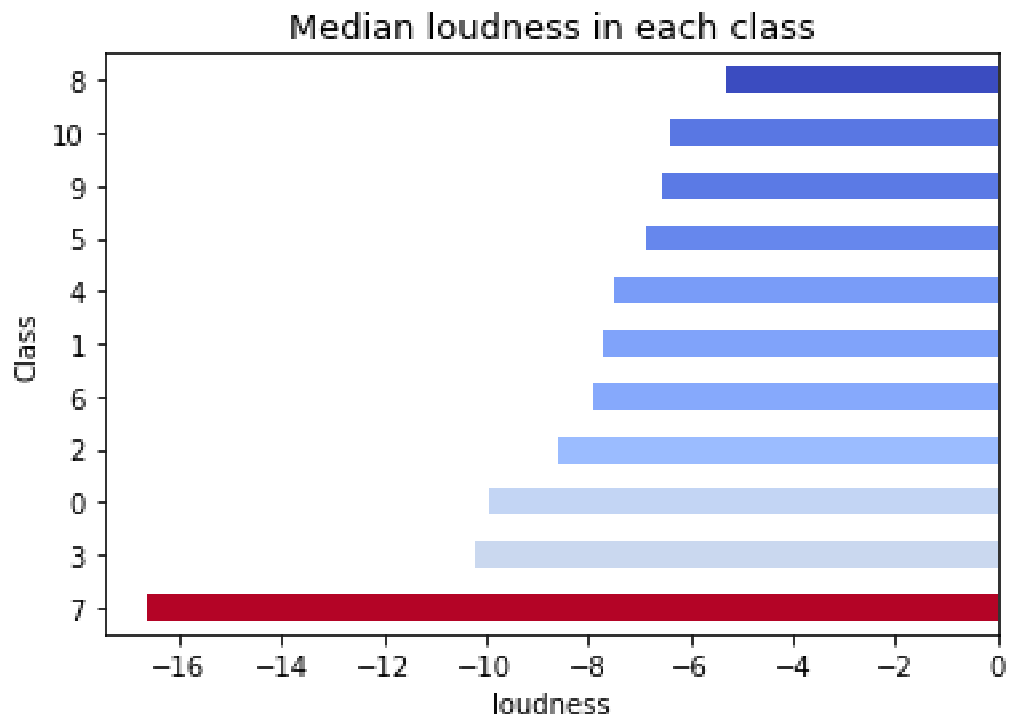
```
def plot_genre_horizontal_bar(col, title=None):
    data = df.groupby('Class')[col].median().sort_values()

    cmap = plt.cm.coolwarm_r
    norm = plt.Normalize(vmin=data.min(), vmax=data.max())
    colors = [cmap(norm(value)) for value in data]
```

```
    data.plot.barh(color=colors)
    plt.xlabel(col)
    plt.title(title, fontdict={'size': 13})
    plt.show()

labels
=["danceability","energy","loudness","speechiness","acousticness","liv
eness","valence","tempo","duration_in_min","instrumentalness"]
for i in range(len(labels)):
    plot_genre_horizontal_bar(labels[i],title="Median "+labels[i]+" in
each class")
```

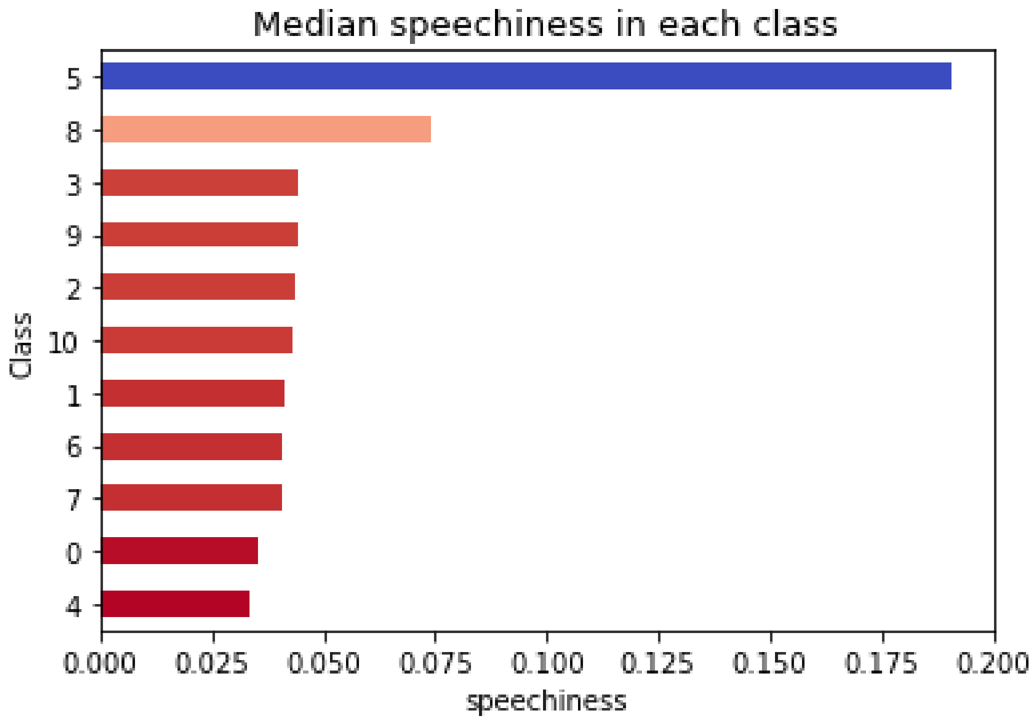


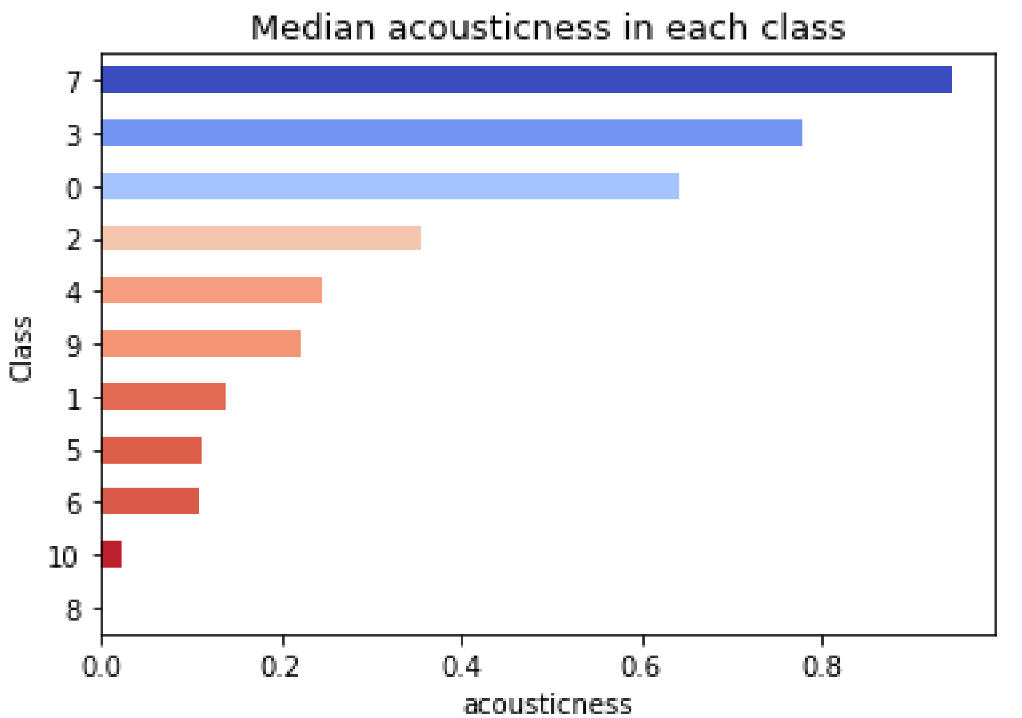


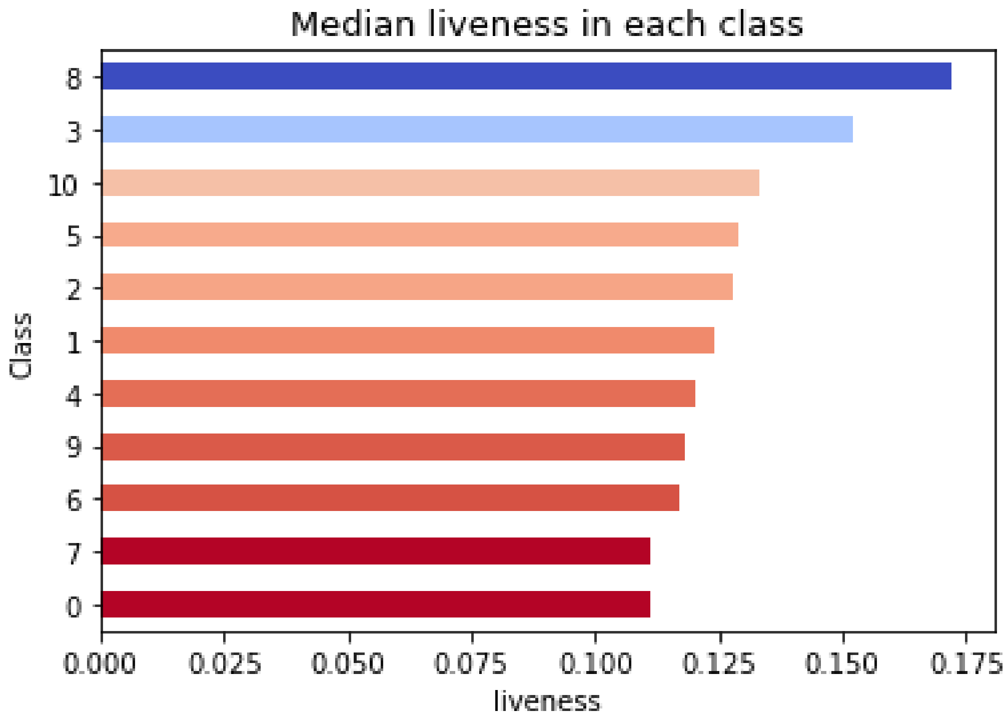
Median energy in each class
8
10
5
6
1
9
4
2
3
0
7
Class
0.0
0.2
0.4
0.6
0.8
energy



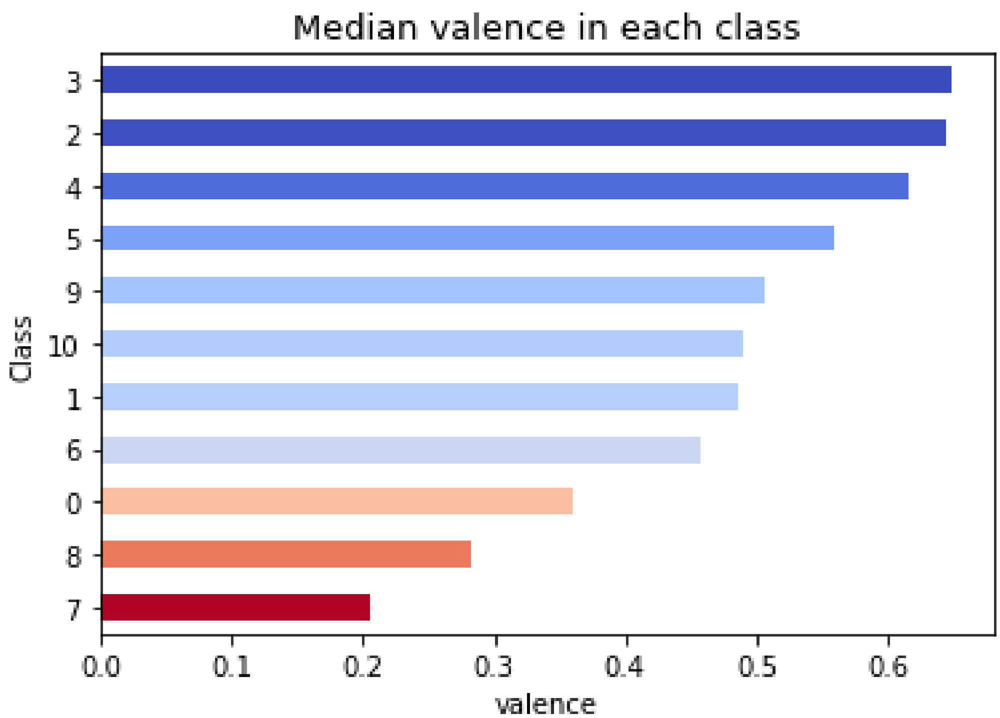
Median loudness in each class
8
10
9
5
4
1
6
2
0
3
7
Class
−16
−14
−12
−10
−8
−6
−4
−2
0
loudness


Median speechiness in each class
Class
5
8
3
9
2
10
1
6
7
0
4
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0.200
speechiness

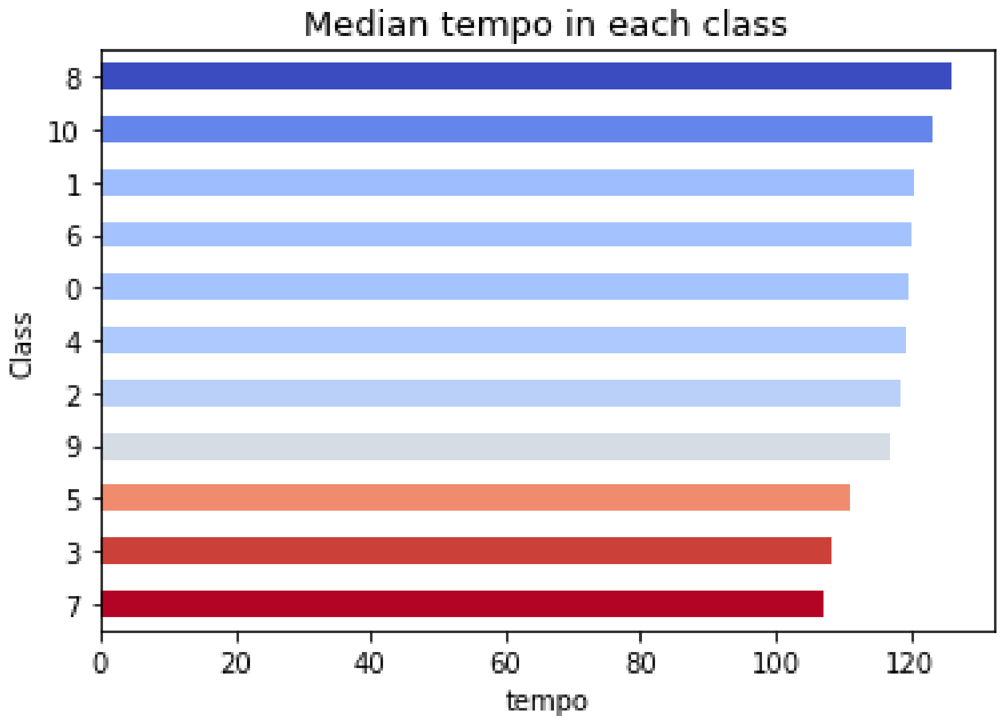

Median acousticness in each class
Class
7
3
0
2
4
9
1
5
6
10
8
0.0
0.2
0.4
0.6
0.8
acousticness

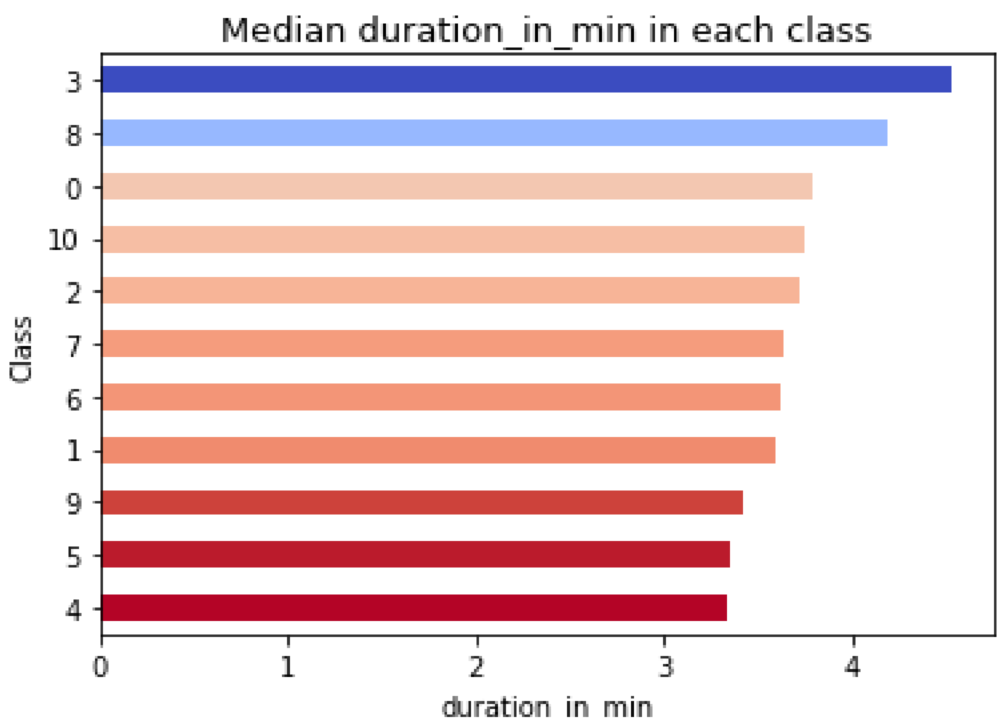


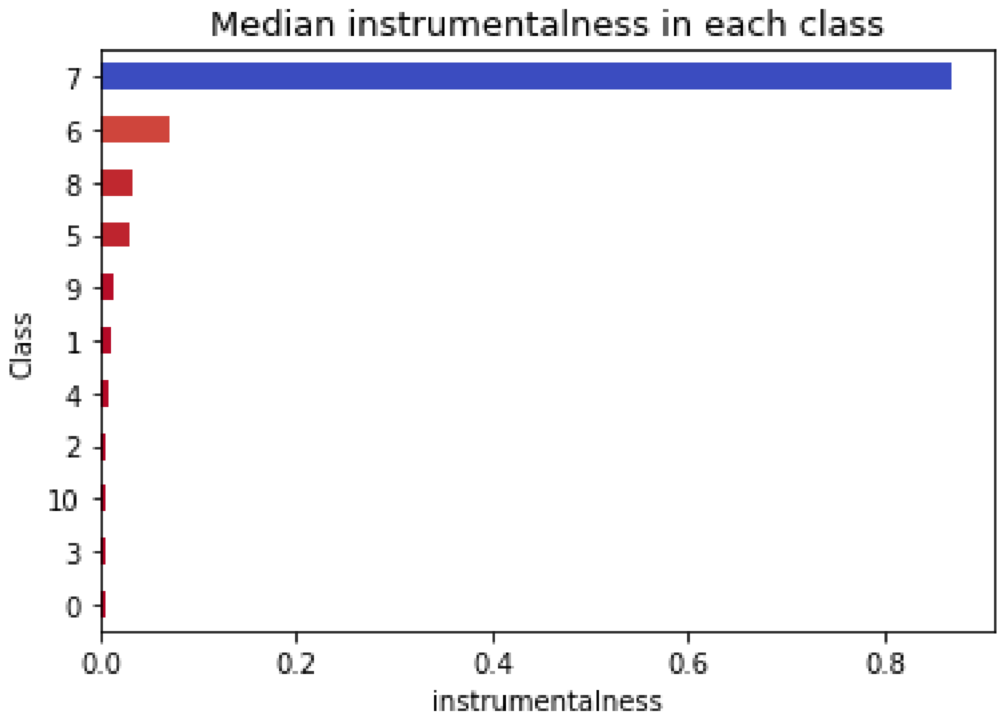
Median liveness in each class
8
3
10
5
2
1
4
9
6
7
0
Class
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
liveness



Median valence in each class
3
2
4
5
9
10
1
6
0
8
7
Class
0.0
0.1
0.2
0.3
0.4
0.5
0.6
valence


Median tempo in each class
Class
8
10
1
6
0
4
2
9
5
3
7
0
20
40
60
80
100
120
tempo


Median duration_in_min in each class
Class
3
8
0
10
2
7
6
1
9
5
4
0
1
2
3
4
duration_in_min



Class with highest danceability:Class 5

Class with highest energy:Class 8

Class with highest volume(loudness):Class 8

Class with highest speechiness:Class 5

Class with highest acousticness:Class 7

Class with highest liveness:Class 8

Class with highest valence:Class 3

Class with highest tempo:Class 8

Class with highest duration:Class 3

Class with highest instrumentalness:Class 7

1. Songs of class 8 have the highest value for most music characteristics.
2. For instrumentalness in each class, all classes except for class 7 have near to 0.0 instrumentalness.

In this section, we will only focus on the 3 most significant variables(Acousticness,Energy and Loudness) as mentioned in the correlation matrix above. I will dive deeper to analyse the relationship between each of the 3 factors and the Class to produce some insights from it.

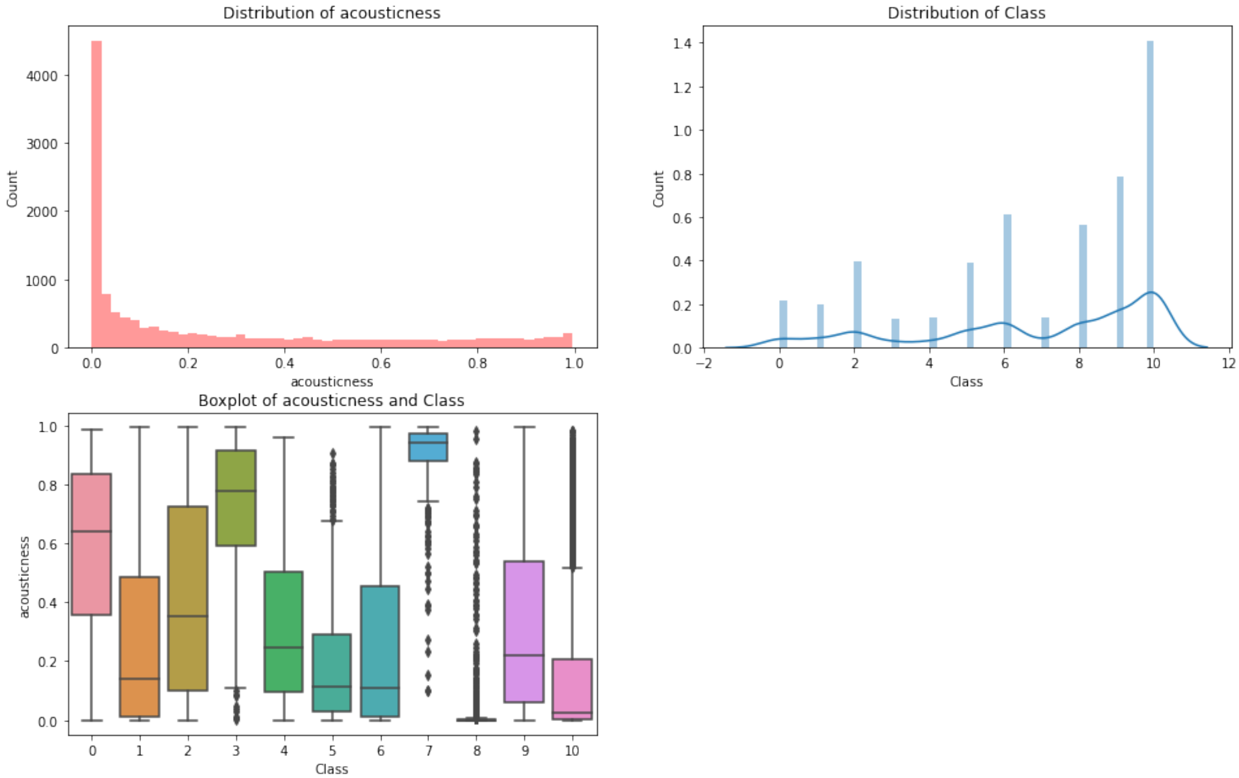
3.4.2.1 Acousticness Vs Class

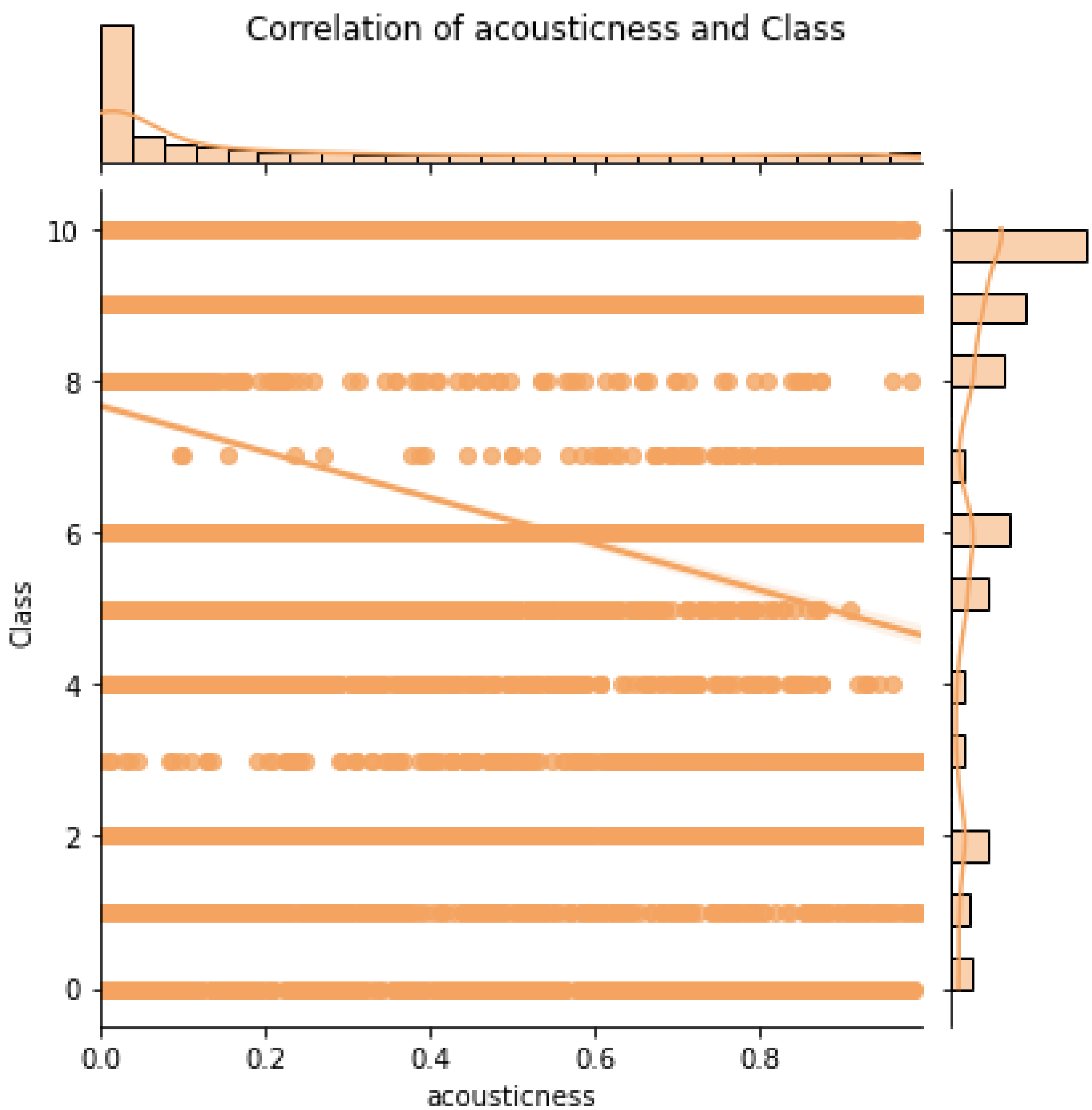
```
figure = plt.figure(figsize=(16,10))
# Distribution of acousticness
ax1 = figure.add_subplot(2, 2, 1)
sns.distplot(df['acousticness'], kde=False, norm_hist=False, bins=50,
color='red', ax=ax1).set_title('Distribution of acousticness')
ax1.set_ylabel("Count")
# Distribution of Class
ax2 = figure.add_subplot(2, 2, 2)
sns.distplot(df['Class'], bins=50, ax=ax2).set_title('Distribution of
Class')
ax2.set_ylabel("Count")
# Boxplot
ax3 = figure.add_subplot(2, 2, 3)
sns.boxplot(data = df, x = 'Class', y =
'acousticness',ax=ax3).set_title("Boxplot of acousticness and Class")
plt.suptitle("acousticness vs Class", fontsize=15)
# Correlation of acousticness and Class
sns.jointplot(x = "acousticness", y = "Class", kind = "reg", data =
df, dropna = True,color='sandybrown')
plt.suptitle("Correlation of acousticness and Class")
```

```
C:\Users\JiaYi\anaconda3\lib\site-packages\seaborn\
distributions.py:2557: FutureWarning: `distplot` is a deprecated
function and will be removed in a future version. Please adapt your
code to use either `displot` (a figure-level function with similar
flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

```
Text(0.5, 0.98, 'Correlation of acousticness and Class')
```

acousticness vs Class





a) Distribution Chart of acousticness: The distribution chart of acousticness is right-skewed which means that the quantity of acousticness is quite less in this dataset. The quantity of acousticness ranges between 0.0 to 1.0 g/dm^3 where the distribution mainly clusters around 0.0 to 0.2.

b) Distribution Chart of class: We can see that the chart is left-skewed where most of the class is class 10.

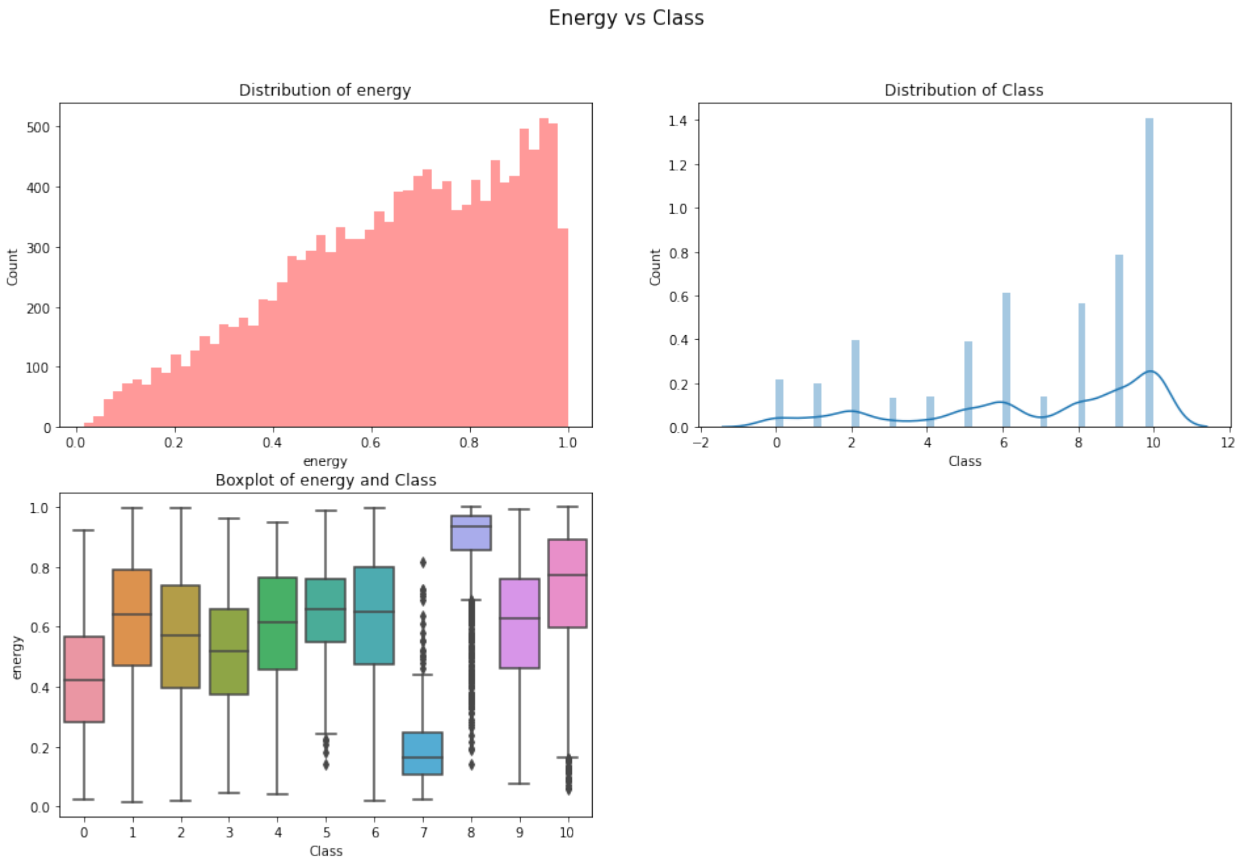
c) Boxplot for acousticness: From the boxplot, we can see that Classes 0,3,7 has a higher value of accousticness where the median value of acousticness is above 0.6. Whereas for classes 1 , 2 , 4 , 5 , 6 , 8 , 9 and 10, song in this class have a lower value of accoustiness where median value is bove 0.4.

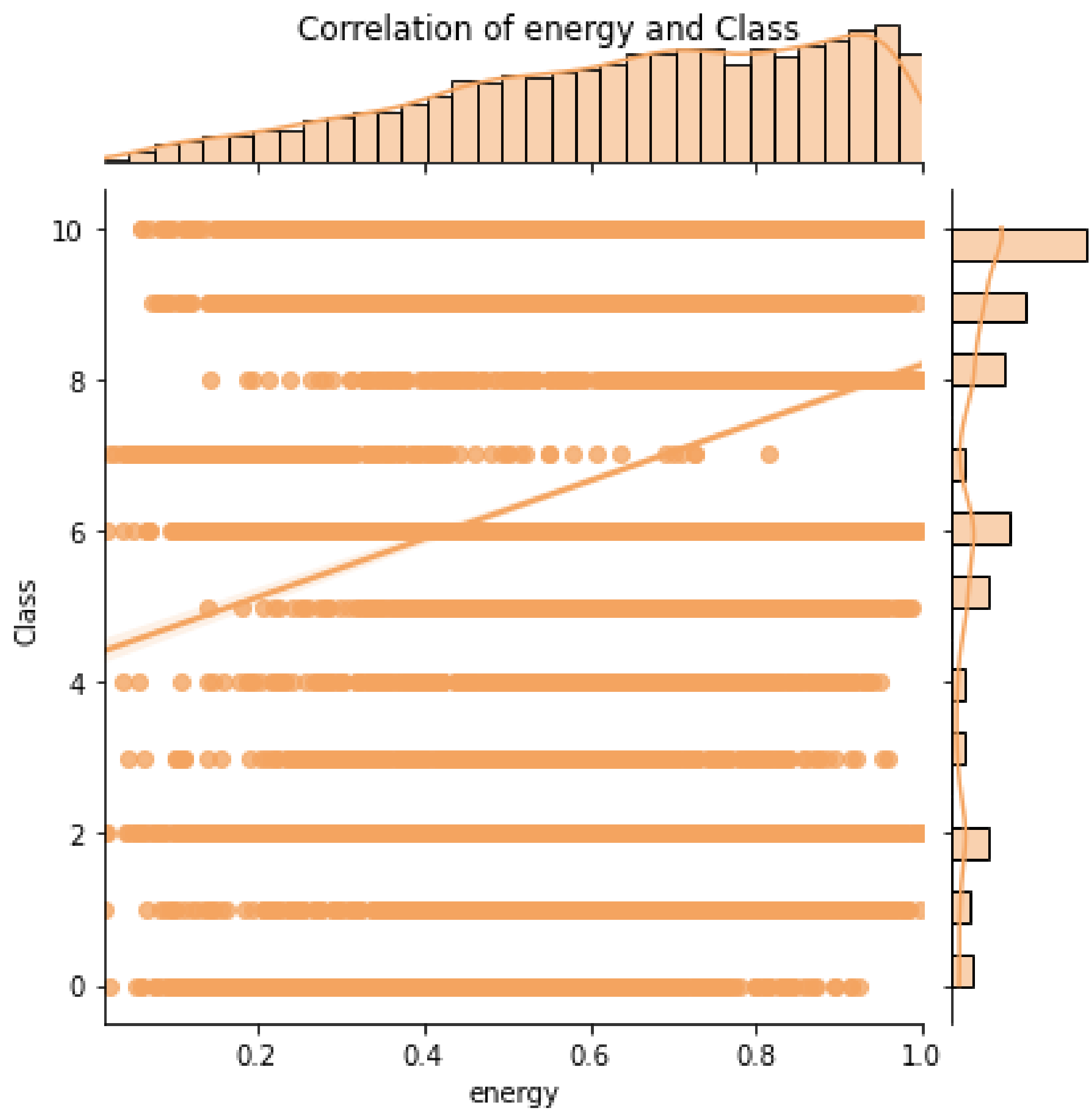
d) Joint Plot for acousticness and class: We have combined the two distribution charts together to see their relationship. We learned that there is an negative relationship between acousticness and class as the orange line shows a negative linear graph. This evidences the results from the boxplot where the increase in acousticness results in an decrease in class number. The graph is quite steeped up which shows that the relationship between acousticness and class is quite strong.

3.4.2.2 Energy Vs Class

```
figure = plt.figure(figsize=(16,10))
# Distribution of energy
ax1 = figure.add_subplot(2, 2, 1)
sns.distplot(df['energy'], kde=False, norm_hist=False, bins=50,
color='red', ax=ax1).set_title('Distribution of energy')
ax1.set_ylabel("Count")
# Distribution of Class
ax2 = figure.add_subplot(2, 2, 2)
sns.distplot(df['Class'], bins=50, ax=ax2).set_title('Distribution of
Class')
ax2.set_ylabel("Count")
# Boxplot
ax3 = figure.add_subplot(2, 2, 3)
sns.boxplot(data = df, x = 'Class', y =
'energy',ax=ax3).set_title("Boxplot of energy and Class")
plt.suptitle("Energy vs Class", fontsize=15)
# Correlation of energy and Class
sns.jointplot(x = "energy", y = "Class", kind = "reg", data = df,
dropna = True,color='sandybrown')
plt.suptitle("Correlation of energy and Class")
```

```
C:\Users\JiaYi\anaconda3\lib\site-packages\seaborn\
distributions.py:2557: FutureWarning: `distplot` is a deprecated
function and will be removed in a future version. Please adapt your
code to use either `displot` (a figure-level function with similar
flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)

Text(0.5, 0.98, 'Correlation of energy and Class')
```


Energy vs Class
Distribution of energy
Count
energy
Distribution of Class
Count
Class
Boxplot of energy and Class
energy
Class



a) Distribution Chart of Energy: The distribution chart of energy is very left-skewed which means that the quantity of energy is quite alot in this dataset. The quantity of energy ranges between 0.0 to 1.0 where the distribution is higher when value of energy increases

b) Distribution Chart of class: We can see that the chart is left-skewed where most of the class is class 10.

c) Boxplot for energy: From the boxplot, we can see that for most classes, their energy level is quite high for most classes the median value of energy is 0.5 and above. For Class 7, the energy level is very low as compared to other Classes with the median value of energy at 0.1~.

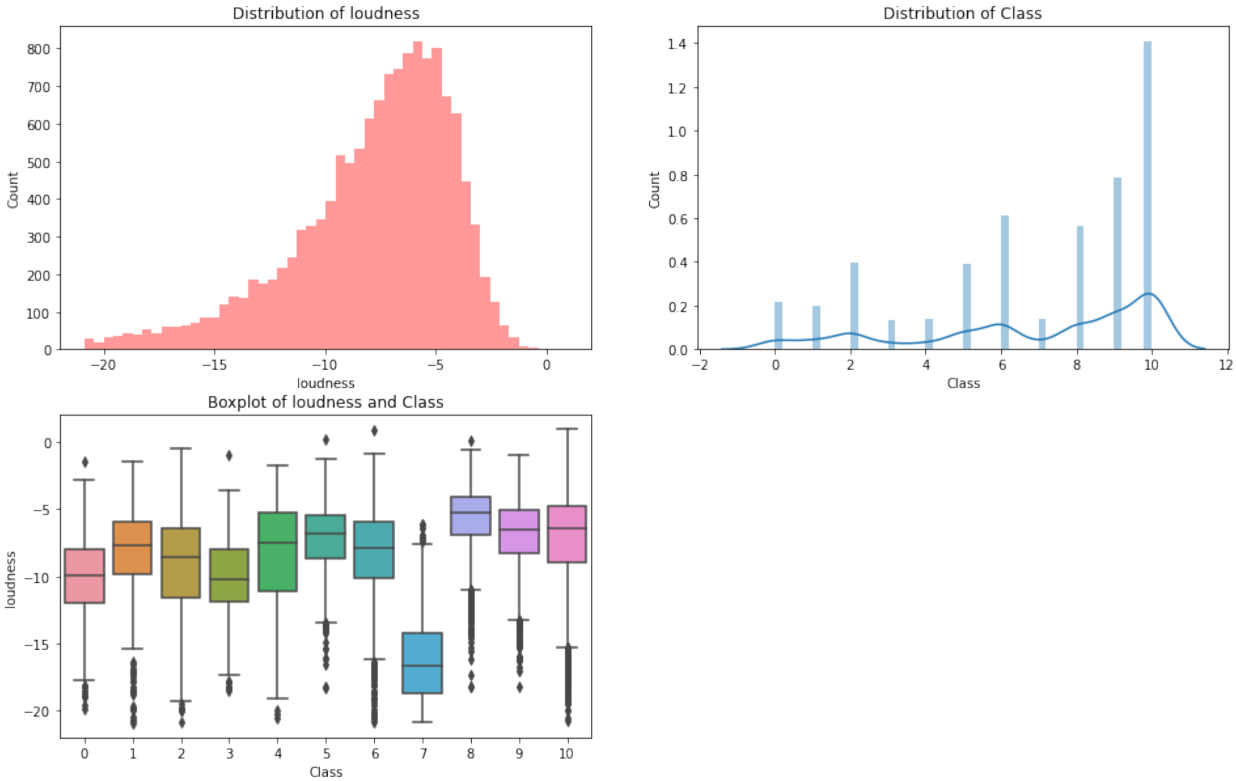
d) Joint Plot for energy and class: We have combined the two distribution charts together to see their relationship. We learned that there is an positive relationship between energy and class as the orange line shows a positive linear graph. This evidences the results from the boxplot where the increase in energy results in an increase in class number. The graph is quite steeped up which shows that the relationship between energy and class is quite strong.

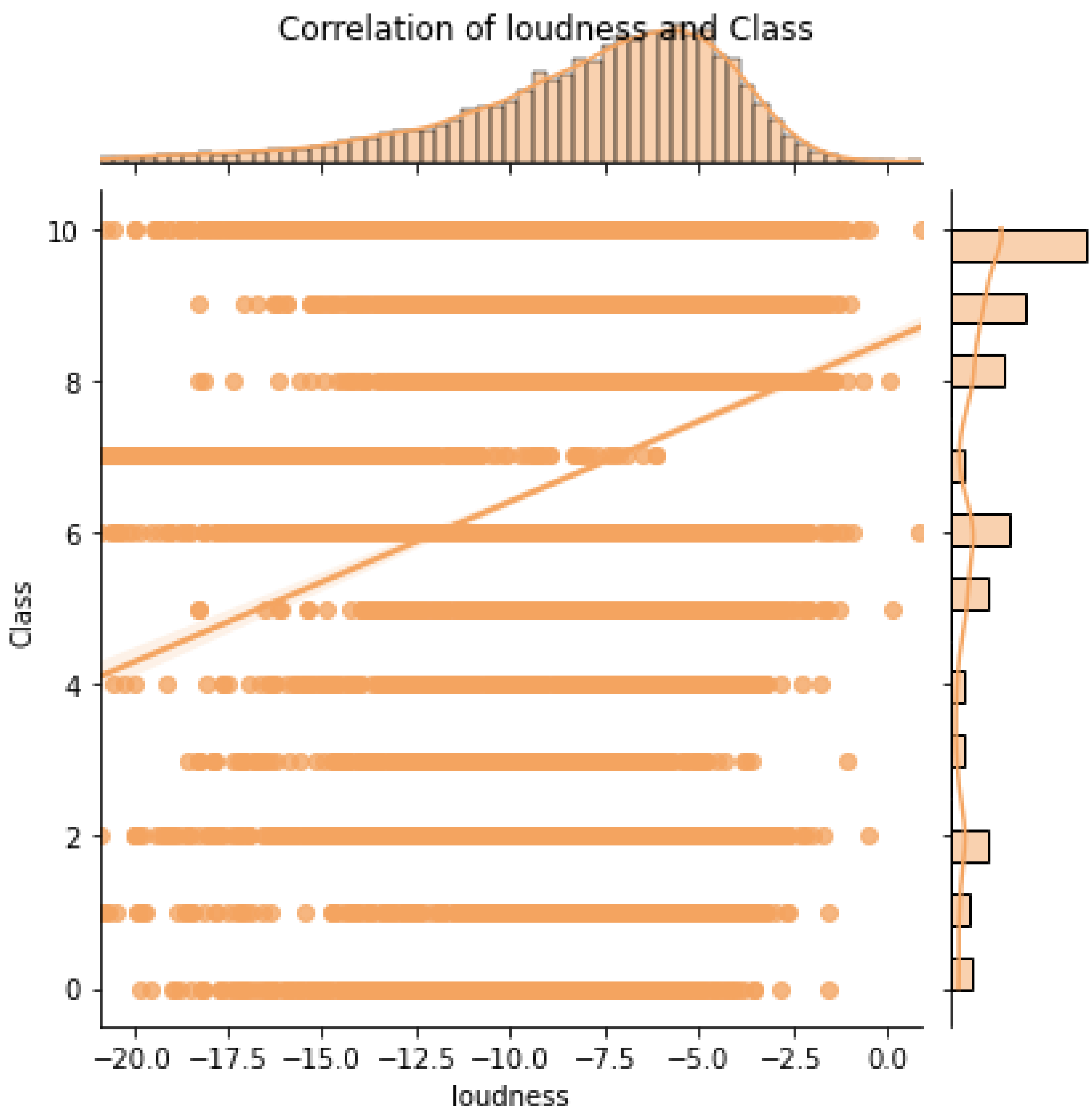
3.4.2.3 Loudness Vs Class

```
figure = plt.figure(figsize=(16,10))
# Distribution of loudness
ax1 = figure.add_subplot(2, 2, 1)
sns.distplot(df['loudness'], kde=False, norm_hist=False, bins=50,
color='red', ax=ax1).set_title('Distribution of loudness')
ax1.set_ylabel("Count")
# Distribution of Class
ax2 = figure.add_subplot(2, 2, 2)
sns.distplot(df['Class'], bins=50, ax=ax2).set_title('Distribution of
Class')
ax2.set_ylabel("Count")
# Boxplot
ax3 = figure.add_subplot(2, 2, 3)
sns.boxplot(data = df, x = 'Class', y =
'loudness',ax=ax3).set_title("Boxplot of loudness and Class")
plt.suptitle("Loudness vs Class", fontsize=15)
# Correlation of loudness and Class
sns.jointplot(x = "loudness", y = "Class", kind = "reg", data = df,
dropna = True,color='sandybrown')
plt.suptitle("Correlation of loudness and Class")
```

```
C:\Users\JiaYi\anaconda3\lib\site-packages\seaborn\
distributions.py:2557: FutureWarning: `distplot` is a deprecated
function and will be removed in a future version. Please adapt your
code to use either `displot` (a figure-level function with similar
flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)

Text(0.5, 0.98, 'Correlation of loudness and Class')
```

Loudness vs Class





a) Distribution Chart of loudness: The distribution chart of loudness is very left-skewed which means that the quantity of loudness is quite alot in this dataset. The quantity of loudness is higher when value is higher (around -10 to -5) 。

b) Distribution Chart of class: We can see that the chart is left-skewed where most of the class is class 10.

c) Boxplot for loudness: From the boxplot, we can see that for most classes, their loudness level is quite high for most classes the median value of loudness is -10 and above. For Class 7, the loudness level is very low as compared to other Classes with the median value of loudness at -17~.

d) Joint Plot for loudness and class: We have combined the two distribution charts together to see their relationship. We learned that there is an positive relationship between loudness and class as the orange line shows a positive linear graph. This evidences the results from the boxplot where the increase in loudness results in an increase in class number. The graph is quite steeped up which shows that the relationship between loudness and class is quite strong.

4. Modelling, Evaluation and Prediction

4.1 Preparing Data for modelling

Let us first drop columns that are not definitely not involved in the prediction of class.

```
df_modelling=df.drop(['Artist Name',"Track Name","Popularity"],axis=1)

df_modelling
```

```
       danceability  energy  loudness  mode  speechiness  acousticness  \
0             0.854   0.564    -4.964     1       0.0485      0.017100

1             0.382   0.814    -7.230     1       0.0406      0.001100

2             0.434   0.614    -8.334     1       0.0525      0.486000

3             0.853   0.597    -6.528     0       0.0555      0.021200

4             0.167   0.975    -4.279     1       0.2160      0.000169

...             ...     ...       ...   ...          ...           ...

13479         0.166   0.109   -17.100     0       0.0413      0.993000

13480         0.638   0.223   -10.174     0       0.0329      0.858000

13481         0.558   0.981    -4.683     0       0.0712      0.000030

13482         0.215   0.805   -12.757     0       0.1340      0.001290

13483         0.400   0.853    -5.320     0       0.0591      0.006040


       liveness  valence    tempo  time_signature  Class
instrumentalness  \
0        0.0849   0.8990  134.071               4      5
0.251237
1        0.1010   0.5690  116.454               4     10
0.004010
2        0.3940   0.7870  147.681               4      6
0.000196
3        0.1220   0.5690  107.033               4      5
0.057497
4        0.1720   0.0918  199.060               4     10
0.016100
...         ...      ...      ...             ...    ...
```

```
...
13479    0.0984   0.1770  171.587                3      6
0.824000
13480    0.0705   0.3350   73.016                4      2
0.000016
13481    0.6660   0.2620  105.000                4      8
0.000136
13482    0.2560   0.3550  131.363                4      8
0.916000
13483    0.3340   0.3770  138.102                4     10
0.212000

        key  duration_in_min
0       1.0         3.909933
1       3.0         4.195550
2       6.0         1.827783
3      10.0         2.899467
4       2.0         3.832667
...     ...              ...
13479   7.0         3.224167
13480  11.0         4.284450
13481   4.0         3.603700
13482   6.0         3.661550
13483   4.0         3.037117

[13484 rows x 14 columns]
```

Split Input Features and Label

```
X=df_modelling[["danceability","energy","mode","loudness","acousticnes
s","speechiness","liveness","valence","tempo","duration_in_min","time_
signature","instrumentalness","key"]]
y=df_modelling[["Class"]]

print(X.shape)
print(y.shape)

(13484, 13)
(13484, 1)
```

Data Balancing

```
#Enter your codes here
y.Class.value_counts()

10    3800
9     2125
6     1654
8     1524
```

```
2     1065
5     1062
0      592
1      543
4      380
7      378
3      361
Name: Class, dtype: int64
```

```
from imblearn.over_sampling import SMOTE
smote=SMOTE()
X,y=smote.fit_resample(X,y)
```

```
#Enter your codes here
y.Class.value_counts()
```

```
0     3800
1     3800
2     3800
3     3800
4     3800
5     3800
6     3800
7     3800
8     3800
9     3800
10    3800
Name: Class, dtype: int64
```

Feature Scaling

Noticed that I only feature scaled non-categorical variable. For encoded categorical variable such as "mode","time_signature" and "key", I did not scaled it beacuase it is already encoded. (Eg Key=B# is 9)

```
scaler = preprocessing.MinMaxScaler()
X[['danceability', 'energy', 'loudness', 'acousticness',
       'speechiness', 'liveness', 'valence', 'tempo',
'duration_in_min',
'instrumentalness']]=scaler.fit_transform(X[['danceability', 'energy',
'loudness', 'acousticness',
       'speechiness', 'liveness', 'valence', 'tempo',
'duration_in_min', 'instrumentalness']])
```

Feature Selection

Refer to section under Decision Tree Model

Note that for feature selection, I will be carrying it out in my first model(Decision Tree) as I will be able to compare the results before and after selecting the features needed to increase model performances. I would need to develop a model first to carry out this process

Train test and split

```
#Enter your codes to split the data into X_train, y_train, X_test and
y_test
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42,stratify=y)
```

4.2 Modelling

From our analysis above, we now have a better understanding on the relationship between each music's characteristics and classes.

As our objective is to better understand what factors affect the class(type of music), we will need to use classification model. There are many types of classification model:

1. Decision Tree Classifier
2. K-Neighbors Classifier
3. Random Forest Classifier
4. Gradient Boosting Classifier
5. XGBoost Classifier
6. Multinomial Nayes Bayes
7. Stochastic Gradient Descent(SGD) Classifier
8. One-Vs-Rest Classifier
9. Neural Network
10. Logistic Regression (only for binary class classification)
11. Support vector Machine (only for binary class classification)

However, I will be ommiting Logistic Regression and SVM as both can only are more suitable for binary classification but in this case we have 11 classes.

How will I be going through this process?

1. Try out each model
2. Fine tune parameters to give the BEST results for each model
3. Compare the best results of each model(produce a table)
4. Derive the model that gives the BEST results out of all models

Metrics: Why I use cross validation? I will be using cross validation rather than classfication report of precision, recall, f1-score and accuracy, as it allows a better representation assesment of the model performance as it test each and every portion of the dataset for each model. This is because our dataset. Also, given that the dataset is small, it is better to get a good representative of each portion of our dataset to get a good and reliable test result.

What area of metric will I be focusing on? However, I will be focusing more on precision in this project. This is becuase we are not targeting on health related classification or prediction where false negatives are important. In this case, more false negatives is not as costly. Hence, we be focusing on false positive, preciseness of ouor classification.

Model 1: Decision tree

A: Building Model - Decision Tree

```
#Enter your codes here to train a DecisionTreeClassifier

from sklearn.tree import DecisionTreeClassifier
tr = DecisionTreeClassifier(random_state=42)
tr.fit(X_train, y_train)
```

```
DecisionTreeClassifier(random_state=42)
```

```
#Enter your codes to print out the depth of the treeI will be
generating the classification report for each model which has
precision, recall, f1-score to help me evaluate the model's
performance. However, I will be focusing more on precision in this
project. This is becuase we are not targeting on health related
classification or prediction where false negatives are important. In
this case, more false negatives is not as costly. Hence, we be
focusing on false positive, preciseness of ouor classification.

tr.get_depth()
```

```
34
```

B: Model Evaluation

We can see that results for classification report performance is lower than from our cross validation performance. Thus, in the next few models, I will be using cross validation as it allows me to test and train every portion of the dataset. Also, given that the dataset is small, it is better to get a good representative of each portion of our dataset to get a good and reliable test result.

Classification

```
from sklearn.metrics import classification_report
y_pred = tr.predict(X_test)
print(y_pred)
print(classification_report(y_test,y_pred))
```

```
[9 6 6 ... 3 7 1]
              precision    recall  f1-score   support

           0       0.64      0.68      0.66       760
           1       0.56      0.57      0.57       760
           2       0.55      0.52      0.53       760
           3       0.83      0.82      0.82       760
```

```
           4       0.75      0.78      0.77       760
           5       0.73      0.76      0.74       760
           6       0.42      0.39      0.40       760
           7       0.90      0.92      0.91       760
           8       0.73      0.73      0.73       760
           9       0.44      0.44      0.44       760
          10       0.31      0.31      0.31       760

    accuracy                           0.63      8360
   macro avg       0.62      0.63      0.63      8360
weighted avg       0.62      0.63      0.63      8360
```

Cross Validation

```python
#Enter your codes to use cross_validation here
from sklearn.model_selection import cross_validate

result = cross_validate(tr, X, np.ravel(y), cv=14,
scoring=['accuracy', 'f1_macro', 'precision_macro', 'recall_macro'])
print("Average Accuracy:", np.mean(result['test_accuracy']))
print("Average Precision:", np.mean(result['test_precision_macro']))
print("Average Recall:", np.mean(result['test_recall_macro']))
print("Average F1:", np.mean(result['test_f1_macro']))
```

```
Average Accuracy: 0.6447868388605228
Average Precision: 0.640549121484521
Average Recall: 0.6447854091317843
Average F1: 0.641730492680019
```

Feature Selection

I will be using mlxtend.feature_selection to do forward featuure selection to find the best subset of features that gives highest precision.

```python
from mlxtend.feature_selection import SequentialFeatureSelector as SFS
sfsl=SFS(tr,k_features=13,forward=True,scoring="precision_macro",cv=10
)
sfsl.fit(X,y)
sfsl.subsets_
```

```
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\metrics\
_classification.py:1245: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\metrics\
_classification.py:1245: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
```

```
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\metrics\
_classification.py:1245: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\metrics\
_classification.py:1245: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\metrics\
_classification.py:1245: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\metrics\
_classification.py:1245: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\metrics\
_classification.py:1245: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\metrics\
_classification.py:1245: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\metrics\
_classification.py:1245: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\metrics\
_classification.py:1245: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\metrics\
_classification.py:1245: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\metrics\
_classification.py:1245: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in labels with no predicted samples. Use
```

```
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\metrics\
_classification.py:1245: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\metrics\
_classification.py:1245: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\metrics\
_classification.py:1245: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\metrics\
_classification.py:1245: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\metrics\
_classification.py:1245: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\metrics\
_classification.py:1245: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\metrics\
_classification.py:1245: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\metrics\
_classification.py:1245: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))

{1: {'feature_idx': (1,),
  'cv_scores': array([0.21350597, 0.2416838 , 0.22786318, 0.23265048,
0.24716546,
         0.24592788, 0.26500939, 0.25876495, 0.25044251, 0.26094909]),
  'avg_score': 0.2443962708445419,
  'feature_names': ('energy',)},
 2: {'feature_idx': (1, 8),
```

```
  'cv_scores': array([0.35639901, 0.33409008, 0.33579637, 0.33935256,
0.35831938,
         0.34739992, 0.33877922, 0.34555062, 0.34992518, 0.34929878]),
  'avg_score': 0.34549111213820444,
  'feature_names': ('energy', 'tempo')},
 3: {'feature_idx': (1, 8, 12),
  'cv_scores': array([0.43915131, 0.44531198, 0.45520738, 0.47340775,
0.46520821,
         0.47468086, 0.47169543, 0.46573106, 0.47843419, 0.48986071]),
  'avg_score': 0.46586888810354143,
  'feature_names': ('energy', 'tempo', 'key')},
 4: {'feature_idx': (1, 3, 8, 12),
  'cv_scores': array([0.51939245, 0.51118539, 0.5158219 , 0.51937717,
0.52980186,
         0.53289051, 0.53897941, 0.53685228, 0.55243178, 0.54740715]),
  'avg_score': 0.530413989685006,
  'feature_names': ('energy', 'loudness', 'tempo', 'key')},
 5: {'feature_idx': (0, 1, 3, 8, 12),
  'cv_scores': array([0.55966383, 0.54549534, 0.54112977, 0.56613674,
0.58306997,
         0.5766361 , 0.57797146, 0.5761864 , 0.5961333 , 0.58716457]),
  'avg_score': 0.5709587477061392,
  'feature_names': ('danceability', 'energy', 'loudness', 'tempo',
'key')},
 6: {'feature_idx': (0, 1, 3, 5, 8, 12),
  'cv_scores': array([0.57786061, 0.56978809, 0.56135708, 0.58060079,
0.58389691,
         0.59339467, 0.60075203, 0.59973715, 0.60919016, 0.59893111]),
  'avg_score': 0.5875508585104094,
  'feature_names': ('danceability',
   'energy',
   'loudness',
   'speechiness',
   'tempo',
   'key')},
 7: {'feature_idx': (0, 1, 3, 5, 8, 9, 12),
  'cv_scores': array([0.59831976, 0.58988287, 0.585654  , 0.60869619,
0.59798934,
         0.59324506, 0.60633809, 0.60712874, 0.61170828, 0.61309464]),
  'avg_score': 0.6012056957651637,
  'feature_names': ('danceability',
   'energy',
   'loudness',
   'speechiness',
   'tempo',
   'duration_in_min',
   'key')},
 8: {'feature_idx': (0, 1, 3, 4, 5, 8, 9, 12),
  'cv_scores': array([0.59512377, 0.6044309 , 0.59625483, 0.60169327,
```

```
0.60002835,
         0.61597486, 0.6223141 , 0.615246  , 0.61665465, 0.63198554]),
  'avg_score': 0.6099706257061465,
  'feature_names': ('danceability',
   'energy',
   'loudness',
   'acousticness',
   'speechiness',
   'tempo',
   'duration_in_min',
   'key')},
 9: {'feature_idx': (0, 1, 3, 4, 5, 7, 8, 9, 12),
  'cv_scores': array([0.60975908, 0.6062045 , 0.59161267, 0.61830645,
0.62934054,
         0.63165594, 0.62910373, 0.6282445 , 0.62623306, 0.65160675]),
  'avg_score': 0.6222067207509479,
  'feature_names': ('danceability',
   'energy',
   'loudness',
   'acousticness',
   'speechiness',
   'valence',
   'tempo',
   'duration_in_min',
   'key')},
 10: {'feature_idx': (0, 1, 3, 4, 5, 7, 8, 9, 11, 12),
  'cv_scores': array([0.60810893, 0.60782551, 0.61091339, 0.61457514,
0.63835222,
         0.63928998, 0.62793504, 0.63696474, 0.64246476, 0.65073374]),
  'avg_score': 0.6277163433081687,
  'feature_names': ('danceability',
   'energy',
   'loudness',
   'acousticness',
   'speechiness',
   'valence',
   'tempo',
   'duration_in_min',
   'instrumentalness',
   'key')},
 11: {'feature_idx': (0, 1, 3, 4, 5, 6, 7, 8, 9, 11, 12),
  'cv_scores': array([0.61860028, 0.62214775, 0.6087508 , 0.62338293,
0.64147109,
         0.63523342, 0.63500247, 0.63191086, 0.64986031, 0.65325907]),
  'avg_score': 0.631961898546043,
  'feature_names': ('danceability',
   'energy',
   'loudness',
   'acousticness',
```

```
    'speechiness',
    'liveness',
    'valence',
    'tempo',
    'duration_in_min',
    'instrumentalness',
    'key')},
 12: {'feature_idx': (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12),
  'cv_scores': array([0.60750589, 0.60128516, 0.61191168, 0.63031825,
0.63883637,
         0.64906892, 0.64467488, 0.63550017, 0.66396102, 0.65680919]),
  'avg_score': 0.633987153755126,
  'feature_names': ('danceability',
    'energy',
    'mode',
    'loudness',
    'acousticness',
    'speechiness',
    'liveness',
    'valence',
    'tempo',
    'duration_in_min',
    'instrumentalness',
    'key')},
 13: {'feature_idx': (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12),
  'cv_scores': array([0.60793708, 0.60550309, 0.61544585, 0.63556984,
0.64032896,
         0.64158872, 0.64799074, 0.63849689, 0.66753903, 0.65578172]),
  'avg_score': 0.6356181914524793,
  'feature_names': ('danceability',
    'energy',
    'mode',
    'loudness',
    'acousticness',
    'speechiness',
    'liveness',
    'valence',
    'tempo',
    'duration_in_min',
    'time_signature',
    'instrumentalness',
    'key')}}
```

Looking at the results, subset 12 which contains 'danceability', 'energy', 'loudness', 'acousticness', 'speechiness', 'valence', 'tempo', 'duration_in_min', 'time_signature', 'instrumentalness', 'key'gives highest precision of 0.6339. Hence we will be using this subset of features to built a new decision tree model and do cross validation to find out the performance of the model.

```
X=df_modelling[['danceability',
   'energy',
   'loudness',
   'acousticness',
   'speechiness',
   'valence',
   'tempo',
   'duration_in_min',
   'time_signature',
   'instrumentalness',
   'key']]
y=df_modelling[["Class"]]

from imblearn.over_sampling import SMOTE
smote=SMOTE()
X,y=smote.fit_resample(X,y)

scaler = preprocessing.MinMaxScaler()
X[['danceability', 'energy', 'loudness', 'acousticness',
       'speechiness', 'valence', 'tempo', 'duration_in_min',
'instrumentalness']]=scaler.fit_transform(X[['danceability', 'energy',
'loudness', 'acousticness',
       'speechiness', 'valence', 'tempo', 'duration_in_min',
'instrumentalness']])

#Enter your codes to split the data into X_train, y_train, X_test and
y_test
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42,stratify=y)

tr = DecisionTreeClassifier(random_state=42)
tr.fit(X_train, y_train)

DecisionTreeClassifier(random_state=42)


result = cross_validate(tr,X, np.ravel(y), cv=14, scoring=['accuracy',
'f1_macro', 'precision_macro', 'recall_macro'])
print("Average Accuracy:", np.mean(result['test_accuracy']))
print("Average Precision:", np.mean(result['test_precision_macro']))
print("Average Recall:", np.mean(result['test_recall_macro']))
print("Average F1:", np.mean(result['test_f1_macro']))

Average Accuracy: 0.639762482555988
Average Precision: 0.6361851740258518
Average Recall: 0.6397611166237712
Average F1: 0.6370620010225501
```

Looks like the results have improved slightly after using subset 12 of features. It increase from 0.645 to 0.636 for the precision metric. Hence, we will be using this subset of features consisting of 'danceability', 'energy', 'loudness', 'acousticness', 'speechiness', 'valence', 'tempo', 'duration_in_min', 'time_signature', 'instrumentalness', 'key' for the rest of our models.

C: Tuning Parameters

How will I be tuning the parameters? ** Hyer-parameter tuning: Use Grid SearchCV to find the best combination of parameters that gives highest precision

Hyper-parameter tuning

Now, we will be trying the Grid SearchCV method.

```
tr = DecisionTreeClassifier(random_state=42)

from sklearn.model_selection import GridSearchCV

params = {
    'max_depth':list(range(1,50)),
     "min_samples_split":list(range(1,10)),
    "min_samples_leaf":list(range(1,10)),
    'criterion': ["gini", "entropy"]
}

# Instantiate the grid search model
grid_search = GridSearchCV(estimator=tr,
                           param_grid=params,
                           cv=4, n_jobs=-1, verbose=1, scoring =
"precision_macro")

%%time

with tf.device('/GPU:0'):
    grid_search.fit(X_train, y_train)
```

```
Fitting 4 folds for each of 7938 candidates, totalling 31752 fits

C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\model_selection\
_search.py:918: UserWarning: One or more of the test scores are non-
finite: [       nan 0.06636748 0.06636748 ... 0.5313526  0.5313526
0.5313526 ]
  warnings.warn(

Wall time: 25min 43s
```

```
score_df = pd.DataFrame(grid_search.cv_results_)
score_df.head()
```

```
   mean_fit_time  std_fit_time  mean_score_time  std_score_time  \
0       0.035261      0.002165         0.000000        0.000000
1       0.070515      0.002872         0.013753        0.001299
2       0.069015      0.001732         0.014004        0.001871
```

```
3       0.050762      0.001299         0.028757        0.005403
4       0.062513      0.005410         0.013505        0.001119

  param_criterion param_max_depth param_min_samples_leaf  \
0            gini               1                      1
1            gini               1                      1
2            gini               1                      1
3            gini               1                      1
4            gini               1                      1

  param_min_samples_split
params  \
0                       1  {'criterion': 'gini', 'max_depth': 1,
'min_sam...
1                       2  {'criterion': 'gini', 'max_depth': 1,
'min_sam...
2                       3  {'criterion': 'gini', 'max_depth': 1,
'min_sam...
3                       4  {'criterion': 'gini', 'max_depth': 1,
'min_sam...
4                       5  {'criterion': 'gini', 'max_depth': 1,
'min_sam...

   split0_test_score  split1_test_score  split2_test_score
split3_test_score  \
0                NaN                NaN                NaN
NaN
1           0.061481           0.066313           0.069185
0.068491
2           0.061481           0.066313           0.069185
0.068491
3           0.061481           0.066313           0.069185
0.068491
4           0.061481           0.066313           0.069185
0.068491

   mean_test_score  std_test_score  rank_test_score
0              NaN             NaN             7938
1         0.066367        0.003014             6913
2         0.066367        0.003014             6913
3         0.066367        0.003014             6913
4         0.066367        0.003014             6913

score_df.nlargest(5,"mean_test_score")

      mean_fit_time  std_fit_time  mean_score_time  std_score_time  \
5671       1.452075      0.021758         0.009252        0.000829
5833       1.519718      0.059970         0.009252        0.001090
5590       1.457577      0.015695         0.008002        0.000001
5509       1.561849      0.091202         0.010002        0.001872
```

```
6076       1.532844      0.061208         0.010003        0.002917

     param_criterion param_max_depth param_min_samples_leaf  \
5671         entropy              22                      1
5833         entropy              24                      1
5590         entropy              21                      1
5509         entropy              20                      1
6076         entropy              27                      1

     param_min_samples_split  \
5671                       2
5833                       2
5590                       2
5509                       2
6076                       2

                                                 params
split0_test_score  \
5671  {'criterion': 'entropy', 'max_depth': 22, 'min...
0.589786
5833  {'criterion': 'entropy', 'max_depth': 24, 'min...
0.584088
5590  {'criterion': 'entropy', 'max_depth': 21, 'min...
0.584271
5509  {'criterion': 'entropy', 'max_depth': 20, 'min...
0.584497
6076  {'criterion': 'entropy', 'max_depth': 27, 'min...
0.584747

      split1_test_score  split2_test_score  split3_test_score  \
5671           0.597872           0.592996           0.596382
5833           0.597898           0.596668           0.596710
5590           0.597949           0.596848           0.594828
5509           0.593150           0.597338           0.597088
6076           0.594409           0.596764           0.595211

      mean_test_score  std_test_score  rank_test_score
5671         0.594259        0.003129                1
5833         0.593841        0.005652                2
5590         0.593474        0.005430                3
5509         0.593018        0.005193                4
6076         0.592783        0.004716                5
```

```
tr_best=grid_search.best_estimator_
tr_best
```

```
DecisionTreeClassifier(criterion='entropy', max_depth=22,
random_state=42)
```

```
tr_best=DecisionTreeClassifier(criterion='entropy', max_depth=22,
random_state=42)


result = cross_validate(tr_best, X, np.ravel(y), cv=14,
scoring=['accuracy', 'f1_macro', 'precision_macro', 'recall_macro'])
print("Average Accuracy:", np.mean(result['test_accuracy']))
print("Average Precision:", np.mean(result['test_precision_macro']))
print("Average Recall:", np.mean(result['test_recall_macro']))
print("Average F1:", np.mean(result['test_f1_macro']))
```

```
Average Accuracy: 0.6449539343449618
Average Precision: 0.6400327611905886
Average Recall: 0.6449605378884193
Average F1: 0.6416296859707229
```

D: Model Insights

After much tuning, let's view the summary of model results:

From this table we can see that we can use the last model which is DecisionTreeClassifier(criterion='entropy', max_depth=22, random_state=42) as it gives the higher precision, accuracy, recall and f1-score

Conclusion: Highest precision of tuned decicison tree is 64%

E: Predictions

```
d = {'danceability': [0.473, 0.55], 'energy': [0.23, 0.33],"loudness":
[-1,-9],
"acousticness":[0.33,0.01],"speechiness":[0.02,0.1],"valence":
[0.32,0.90],"tempo":[0.10,0.90],
"duration_in_min":[7,9],"time_signature":[3,4],"instrumentalness":
[0.23,0.78],"key":[3,4]}
df = pd.DataFrame(data=d)
tr_best.fit(X_train,y_train)
df['class']=tr_best.predict(df)
display(df)
```

```
   danceability  energy  loudness  acousticness  speechiness  valence
tempo  \
0         0.473    0.23        -1          0.33         0.02     0.32
0.1
1         0.550    0.33        -9          0.01         0.10     0.90
0.9

   duration_in_min  time_signature  instrumentalness  key  class
0                7               3              0.23    3      0
1                9               4              0.78    4      4
```

Model 2: K-Neighbors Classifier

A: Building Model - K-Neighbors Classifier

```
from sklearn.neighbors import KNeighborsClassifier as KNC

classifier = KNC()
classifier.fit(X_train, y_train)
```

```
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\neighbors\
_classification.py:179: DataConversionWarning: A column-vector y was
passed when a 1d array was expected. Please change the shape of y to
(n_samples,), for example using ravel().
  return self._fit(X, y)
```

```
KNeighborsClassifier()
```

B: Model Evaluation

Cross Validation

```
#Enter your codes to use cross_validation here
from sklearn.model_selection import cross_validate

result = cross_validate(classifier, X, np.ravel(y), cv=14,
scoring=['accuracy', 'f1_macro', 'precision_macro', 'recall_macro'])
print("Average Accuracy:", np.mean(result['test_accuracy']))
print("Average Precision:", np.mean(result['test_precision_macro']))
print("Average Recall:", np.mean(result['test_recall_macro']))
print("Average F1:", np.mean(result['test_f1_macro']))
```

```
Average Accuracy: 0.6529678100578484
Average Precision: 0.6446341092518711
Average Recall: 0.6529785807311853
Average F1: 0.6346146245080541
```

C: Tuning Parameters

```
from sklearn.model_selection import GridSearchCV
knn_2 = KNC()
params = {
    'leaf_size': list(range(1,50)),
    'n_neighbors': list(range(1,100)),
    'p': [1,2]
}

# Instantiate the grid search model
grid_search = GridSearchCV(estimator=knn_2,
                           param_grid=params,
```

```
                          cv=4, n_jobs=-1, verbose=1, scoring =
"precision_macro")


%%time

#with tf.device('/GPU:0'):
grid_search.fit(X_train, y_train)

score_df = pd.DataFrame(grid_search.cv_results_)
score_df.head()
```

```
Fitting 4 folds for each of 9702 candidates, totalling 38808 fits

C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\neighbors\
_classification.py:179: DataConversionWarning: A column-vector y was
passed when a 1d array was expected. Please change the shape of y to
(n_samples,), for example using ravel().
  return self._fit(X, y)

Wall time: 10h 35min 31s

   mean_fit_time  std_fit_time  mean_score_time  std_score_time  \
0       0.932252      0.252040         3.303913        0.104637
1       0.882536      0.086670         3.180631        0.155629
2       1.274602      0.109488         5.133861        0.138603
3       1.020289      0.130202         4.819467        0.172410
4       0.675785      0.033167         6.085378        0.270490

  param_leaf_size param_n_neighbors param_p  \
0               1                 1       1
1               1                 1       2
2               1                 2       1
3               1                 2       2
4               1                 3       1

                                       params  split0_test_score  \
0  {'leaf_size': 1, 'n_neighbors': 1, 'p': 1}           0.766863
1  {'leaf_size': 1, 'n_neighbors': 1, 'p': 2}           0.749810
2  {'leaf_size': 1, 'n_neighbors': 2, 'p': 1}           0.725704
3  {'leaf_size': 1, 'n_neighbors': 2, 'p': 2}           0.702868
4  {'leaf_size': 1, 'n_neighbors': 3, 'p': 1}           0.703114

   split1_test_score  split2_test_score  split3_test_score
mean_test_score  \
0           0.777561           0.769614           0.778194
0.773058
1           0.752977           0.749671           0.761857
0.753579
2           0.722294           0.722202           0.731138
0.725334
```

```
3           0.708004           0.692397           0.714996
0.704566
4           0.701149           0.703853           0.720392
0.707127

   std_test_score  rank_test_score
0        0.004922               45
1        0.004959               88
2        0.003636              145
3        0.008240              216
4        0.007722              151
```

```
score_df.nlargest(5,"mean_test_score")
```

```
      mean_fit_time  std_fit_time  mean_score_time  std_score_time  \
4752       0.377834      0.026230         3.190966        1.193384
4950       0.363832      0.024206         3.941885        0.775656
5148       0.374084      0.034352         3.822608        1.110617
5346       0.361831      0.011391         4.169937        1.155537
5544       0.354579      0.019050         4.664179        1.132935

     param_leaf_size param_n_neighbors param_p  \
4752              25                 1       1
4950              26                 1       1
5148              27                 1       1
5346              28                 1       1
5544              29                 1       1

                                               params
split0_test_score  \
4752  {'leaf_size': 25, 'n_neighbors': 1, 'p': 1}           0.767001

4950  {'leaf_size': 26, 'n_neighbors': 1, 'p': 1}           0.767001

5148  {'leaf_size': 27, 'n_neighbors': 1, 'p': 1}           0.767001

5346  {'leaf_size': 28, 'n_neighbors': 1, 'p': 1}           0.767001

5544  {'leaf_size': 29, 'n_neighbors': 1, 'p': 1}           0.767001


      split1_test_score  split2_test_score  split3_test_score  \
4752           0.777561           0.769614           0.778329
4950           0.777561           0.769614           0.778329
5148           0.777561           0.769614           0.778329
5346           0.777561           0.769614           0.778329
5544           0.777561           0.769614           0.778329

      mean_test_score  std_test_score  rank_test_score
4752         0.773126        0.004914                1
4950         0.773126        0.004914                1
```

```
5148        0.773126        0.004914                1
5346        0.773126        0.004914                1
5544        0.773126        0.004914                1
```

```
knc_best=grid_search.best_estimator_
knc_best
```

```
KNeighborsClassifier(leaf_size=25, n_neighbors=1, p=1)
```

```
knc_best=KNC(leaf_size=25, n_neighbors=1, p=1)
```

```
#Enter your codes to use cross_validation here
from sklearn.model_selection import cross_validate

result = cross_validate(knc_best, X, np.ravel(y), cv=14,
scoring=['accuracy', 'f1_macro', 'precision_macro', 'recall_macro'])
print("Average Accuracy:", np.mean(result['test_accuracy']))
print("Average Precision:", np.mean(result['test_precision_macro']))
print("Average Recall:", np.mean(result['test_recall_macro']))
print("Average F1:", np.mean(result['test_f1_macro']))
```

```
Average Accuracy: 0.7947866385158336
Average Precision: 0.7880893693831723
Average Recall: 0.7947811630522723
Average F1: 0.7864581015088197
```

D: Model Insights

After much tuning, let's view the summary of model results:

Conclusion: Highest precision of tuned k-neighbors classifier is 79%.

E: Predictions

```
d = {'danceability': [0.473, 0.55], 'energy': [0.23, 0.33],"loudness":
[-1,-9],
"acousticness":[0.33,0.01],"speechiness":[0.02,0.1],"valence":
[0.32,0.90],"tempo":[0.10,0.90],
"duration_in_min":[7,9],"time_signature":[3,4],"instrumentalness":
[0.23,0.78],"key":[3,4]}
df = pd.DataFrame(data=d)
knc_best.fit(X_train,y_train)
df['class']=knc_best.predict(df)
display(df)
```

```
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\neighbors\
_classification.py:179: DataConversionWarning: A column-vector y was
passed when a 1d array was expected. Please change the shape of y to
(n_samples,), for example using ravel().
  return self._fit(X, y)
```

```
   danceability  energy  loudness  acousticness  speechiness  valence  
tempo  \
0         0.473    0.23        -1          0.33         0.02     0.32   
0.1   
1         0.550    0.33        -9          0.01         0.10     0.90   
0.9   

   duration_in_min  time_signature  instrumentalness  key  class  
0                7               3              0.23    3      3  
1                9               4              0.78    4      1  
```

Model 3: Random Forest Classifier

A: Building Model - Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier
RFC=RandomForestClassifier()
RFC.fit(X_train, y_train)
```

```
<ipython-input-271-8594d7246451>:3: DataConversionWarning: A column-
vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples,), for example using ravel().
  RFC.fit(X_train, y_train)

RandomForestClassifier()
```

B: Model Evaluation

Cross Validation

```
#Enter your codes to use cross_validation here
from sklearn.model_selection import cross_validate

result = cross_validate(RFC, X, np.ravel(y), cv=14,
scoring=['accuracy', 'f1_macro', 'precision_macro', 'recall_macro'])
print("Average Accuracy:", np.mean(result['test_accuracy']))
print("Average Precision:", np.mean(result['test_precision_macro']))
print("Average Recall:", np.mean(result['test_recall_macro']))
print("Average F1:", np.mean(result['test_f1_macro']))
```

```
Average Accuracy: 0.7978252904557233
Average Precision: 0.79187426913976
Average Recall: 0.7978170218104016
Average F1: 0.788701617264592
```

C: Tuning Parameters

```

with tf.device('/GPU:0'):
```

```
    from sklearn.model_selection import RepeatedStratifiedKFold
    RFC2 = RandomForestClassifier()
    n_estimators = [10, 100, 1000]
    max_features = ['sqrt', 'log2']
    # define grid search
    grid = dict(n_estimators=n_estimators,max_features=max_features)
    cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3,
random_state=1)
    grid_search = GridSearchCV(estimator=RFC2, param_grid=grid,
n_jobs=-1, cv=cv, scoring='precision_macro',error_score=0)
    grid_result = grid_search.fit(X, y)
```

```
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\model_selection\
_search.py:880: DataConversionWarning: A column-vector y was passed
when a 1d array was expected. Please change the shape of y to
(n_samples,), for example using ravel().
  self.best_estimator_.fit(X, y, **fit_params)
```

```
print("Best: %f using %s" % (grid_result.best_score_,
grid_result.best_params_))
```

```
Best: 0.795515 using {'max_features': 'sqrt', 'n_estimators': 1000}
```

```
RFC_best =
RandomForestClassifier(max_features='sqrt',n_estimators=1000)
RFC_best
```

```
RandomForestClassifier(max_features='sqrt', n_estimators=1000)
```

```
RFC_best=RandomForestClassifier(max_features='sqrt',
n_estimators=1000)
```

```
#Enter your codes to use cross_validation here
from sklearn.model_selection import cross_validate

result = cross_validate(RFC_best, X, np.ravel(y), cv=14,
scoring=['accuracy', 'f1_macro', 'precision_macro', 'recall_macro'])
print("Average Accuracy:", np.mean(result['test_accuracy']))
print("Average Precision:", np.mean(result['test_precision_macro']))
print("Average Recall:", np.mean(result['test_recall_macro']))
print("Average F1:", np.mean(result['test_f1_macro']))
```

```
Average Accuracy: 0.8050500324798812
Average Precision: 0.8004426710191744
Average Recall: 0.8050405544639864
Average F1: 0.7963544425675914
```

D: Model Insights

After much tuning, let's view the summary of model results:

Conclusion: Highest precision of tuned Random Forest classifier is 80%

E: Predictions

```
d = {'danceability': [0.473, 0.55], 'energy': [0.23, 0.33],"loudness":
[-1,-9],
"acousticness":[0.33,0.01],"speechiness":[0.02,0.1],"valence":
[0.32,0.90],"tempo":[0.10,0.90],
"duration_in_min":[7,9],"time_signature":[3,4],"instrumentalness":
[0.23,0.78],"key":[3,4]}
df = pd.DataFrame(data=d)
RFC_best.fit(X_train,y_train)
df['class']=RFC_best.predict(df)
display(df)
```

```
<ipython-input-294-345b35654605>:5: DataConversionWarning: A column-
vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples,), for example using ravel().
  RFC_best.fit(X_train,y_train)
```

```
   danceability  energy  loudness  acousticness  speechiness  valence
tempo  \
0         0.473    0.23        -1          0.33         0.02     0.32
0.1
1         0.550    0.33        -9          0.01         0.10     0.90
0.9

   duration_in_min  time_signature  instrumentalness  key  class
0                7               3              0.23    3      2
1                9               4              0.78    4      2
```

Model 4: XGBoost Classifier

A: Building Model - XGBoost Classifier

```
from xgboost import XGBClassifier

xgb = XGBClassifier()
xgb.fit(X_train, y_train)
```

```
C:\Users\JiaYi\anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\utils\
validation.py:63: DataConversionWarning: A column-vector y was passed
```

```
when a 1d array was expected. Please change the shape of y to
(n_samples, ), for example using ravel().
  return f(*args, **kwargs)

[10:33:33] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0,
the default evaluation metric used with the objective 'multi:softprob'
was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if
you'd like to restore the old behavior.

XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1,
enable_categorical=False,
              gamma=0, gpu_id=-1, importance_type=None,
              interaction_constraints='', learning_rate=0.300000012,
              max_delta_step=0, max_depth=6, min_child_weight=1,
missing=nan,
              monotone_constraints='()', n_estimators=100, n_jobs=16,
              num_parallel_tree=1, objective='multi:softprob',
predictor='auto',
              random_state=0, reg_alpha=0, reg_lambda=1,
scale_pos_weight=None,
              subsample=1, tree_method='exact', validate_parameters=1,
              verbosity=None)
```

B: Model Evaluation

Cross Validation

```
#Enter your codes to use cross_validation here
from sklearn.model_selection import cross_validate

result = cross_validate(xgb, X, np.ravel(y), cv=14,
scoring=['accuracy', 'f1_macro', 'precision_macro', 'recall_macro'])
print("Average Accuracy:", np.mean(result['test_accuracy']))
print("Average Precision:", np.mean(result['test_precision_macro']))
print("Average Recall:", np.mean(result['test_recall_macro']))
print("Average F1:", np.mean(result['test_f1_macro']))
```

```
C:\Users\JiaYi\anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[10:33:44] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0,
the default evaluation metric used with the objective 'multi:softprob'
```

```
was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if
you'd like to restore the old behavior.

C:\Users\JiaYi\anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[10:33:57] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0,
the default evaluation metric used with the objective 'multi:softprob'
was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if
you'd like to restore the old behavior.

C:\Users\JiaYi\anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[10:34:10] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0,
the default evaluation metric used with the objective 'multi:softprob'
was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if
you'd like to restore the old behavior.

C:\Users\JiaYi\anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[10:34:23] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0,
the default evaluation metric used with the objective 'multi:softprob'
was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if
you'd like to restore the old behavior.

C:\Users\JiaYi\anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
```

```
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[10:34:36] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0,
the default evaluation metric used with the objective 'multi:softprob'
was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if
you'd like to restore the old behavior.

C:\Users\JiaYi\anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[10:34:48] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0,
the default evaluation metric used with the objective 'multi:softprob'
was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if
you'd like to restore the old behavior.

C:\Users\JiaYi\anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[10:35:02] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0,
the default evaluation metric used with the objective 'multi:softprob'
was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if
you'd like to restore the old behavior.

C:\Users\JiaYi\anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[10:35:14] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0,
the default evaluation metric used with the objective 'multi:softprob'
was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if
you'd like to restore the old behavior.
```

```
C:\Users\JiaYi\anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[10:35:28] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0,
the default evaluation metric used with the objective 'multi:softprob'
was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if
you'd like to restore the old behavior.

C:\Users\JiaYi\anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[10:35:41] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0,
the default evaluation metric used with the objective 'multi:softprob'
was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if
you'd like to restore the old behavior.

C:\Users\JiaYi\anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[10:35:54] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0,
the default evaluation metric used with the objective 'multi:softprob'
was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if
you'd like to restore the old behavior.

C:\Users\JiaYi\anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
```

```
[10:36:07] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'multi:softprob' was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if you'd like to restore the old behavior.

C:\Users\JiaYi\anaconda3\lib\site-packages\xgboost\sklearn.py:1224: UserWarning: The use of label encoder in XGBClassifier is deprecated and will be removed in a future release. To remove this warning, do the following: 1) Pass option use_label_encoder=False when constructing XGBClassifier object; and 2) Encode your labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[10:36:20] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'multi:softprob' was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if you'd like to restore the old behavior.

C:\Users\JiaYi\anaconda3\lib\site-packages\xgboost\sklearn.py:1224: UserWarning: The use of label encoder in XGBClassifier is deprecated and will be removed in a future release. To remove this warning, do the following: 1) Pass option use_label_encoder=False when constructing XGBClassifier object; and 2) Encode your labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[10:36:33] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'multi:softprob' was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
Average Accuracy: 0.7339262814887758
Average Precision: 0.7332907976461368
Average Recall: 0.7339384554559832
Average F1: 0.7269116476683986
```

C: Tuning Parameters

```
params={"learning_rate"    : [0.05, 0.10, 0.15, 0.20, 0.25, 0.30 ] ,
 "max_depth"        : [ 3, 4, 5, 6, 8, 10, 12, 15],
 "min_child_weight" : [ 1, 3, 5, 7 ],
 "gamma"            : [ 0.0, 0.1, 0.2 , 0.3, 0.4 ],
 "colsample_bytree" : [ 0.3, 0.4, 0.5 , 0.7 ] }


xgb_2 = XGBClassifier()
xgb_2.fit(X_train, y_train)
grid_search = GridSearchCV(estimator=xgb_2,
```

```
                          param_grid=params,
                          cv=4, n_jobs=-1, verbose=1, scoring =
"precision_macro")
```

```
[08:40:27] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0,
the default evaluation metric used with the objective 'multi:softprob'
was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if
you'd like to restore the old behavior.
```

```
%%time
with tf.device('/GPU:0'):
    grid_search.fit(X_train, y_train)

    score_df = pd.DataFrame(grid_search.cv_results_)
    score_df.head()
```

```
Fitting 4 folds for each of 3840 candidates, totalling 15360 fits
```

```
score_df.nlargest(5,"mean_test_score")
```

```
xgb_best=grid_search.best_estimator_
xgb_best
```

```
from xgboost import XGBClassifier
xgb_best=XGBClassifier(base_score=0.5, booster='gbtree',
colsample_bylevel=1, colsample_bynode=1, colsample_bytree=0.7,
enable_categorical=False, gamma=0.0, gpu_id=-1, importance_type=None,
interaction_constraints='', learning_rate=0.2, max_delta_step=0,
max_depth=15, min_child_weight=1, missing=np.nan,
monotone_constraints='()', n_estimators=100, n_jobs=16,
num_parallel_tree=1, objective='multi:softprob', predictor='auto',
random_state=0, reg_alpha=0, reg_lambda=1, scale_pos_weight=None,
subsample=1, tree_method='exact', validate_parameters=1,
verbosity=None)
```

```
#Enter your codes to use cross_validation here
from sklearn.model_selection import cross_validate

result = cross_validate(xgb_best, X, np.ravel(y), cv=14,
scoring=['accuracy', 'f1_macro', 'precision_macro', 'recall_macro'])
print("Average Accuracy:", np.mean(result['test_accuracy']))
print("Average Precision:", np.mean(result['test_precision_macro']))
print("Average Recall:", np.mean(result['test_recall_macro']))
print("Average F1:", np.mean(result['test_f1_macro']))
```

```
C:\Users\JiaYi\anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
```

```
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[10:36:46] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'multi:softprob' was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if you'd like to restore the old behavior.

C:\Users\JiaYi\anaconda3\lib\site-packages\xgboost\sklearn.py:1224: UserWarning: The use of label encoder in XGBClassifier is deprecated and will be removed in a future release. To remove this warning, do the following: 1) Pass option use_label_encoder=False when constructing XGBClassifier object; and 2) Encode your labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[10:37:13] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'multi:softprob' was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if you'd like to restore the old behavior.

C:\Users\JiaYi\anaconda3\lib\site-packages\xgboost\sklearn.py:1224: UserWarning: The use of label encoder in XGBClassifier is deprecated and will be removed in a future release. To remove this warning, do the following: 1) Pass option use_label_encoder=False when constructing XGBClassifier object; and 2) Encode your labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[10:37:41] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'multi:softprob' was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if you'd like to restore the old behavior.

C:\Users\JiaYi\anaconda3\lib\site-packages\xgboost\sklearn.py:1224: UserWarning: The use of label encoder in XGBClassifier is deprecated and will be removed in a future release. To remove this warning, do the following: 1) Pass option use_label_encoder=False when constructing XGBClassifier object; and 2) Encode your labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[10:38:09] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'multi:softprob' was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
```

```
C:\Users\JiaYi\anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[10:38:37] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0,
the default evaluation metric used with the objective 'multi:softprob'
was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if
you'd like to restore the old behavior.

C:\Users\JiaYi\anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[10:39:05] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0,
the default evaluation metric used with the objective 'multi:softprob'
was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if
you'd like to restore the old behavior.

C:\Users\JiaYi\anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[10:39:33] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0,
the default evaluation metric used with the objective 'multi:softprob'
was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if
you'd like to restore the old behavior.

C:\Users\JiaYi\anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
```

```
[10:40:02] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'multi:softprob' was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if you'd like to restore the old behavior.

C:\Users\JiaYi\anaconda3\lib\site-packages\xgboost\sklearn.py:1224: UserWarning: The use of label encoder in XGBClassifier is deprecated and will be removed in a future release. To remove this warning, do the following: 1) Pass option use_label_encoder=False when constructing XGBClassifier object; and 2) Encode your labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[10:40:30] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'multi:softprob' was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if you'd like to restore the old behavior.

C:\Users\JiaYi\anaconda3\lib\site-packages\xgboost\sklearn.py:1224: UserWarning: The use of label encoder in XGBClassifier is deprecated and will be removed in a future release. To remove this warning, do the following: 1) Pass option use_label_encoder=False when constructing XGBClassifier object; and 2) Encode your labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[10:40:58] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'multi:softprob' was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if you'd like to restore the old behavior.

C:\Users\JiaYi\anaconda3\lib\site-packages\xgboost\sklearn.py:1224: UserWarning: The use of label encoder in XGBClassifier is deprecated and will be removed in a future release. To remove this warning, do the following: 1) Pass option use_label_encoder=False when constructing XGBClassifier object; and 2) Encode your labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[10:41:27] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'multi:softprob' was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if you'd like to restore the old behavior.

C:\Users\JiaYi\anaconda3\lib\site-packages\xgboost\sklearn.py:1224: UserWarning: The use of label encoder in XGBClassifier is deprecated and will be removed in a future release. To remove this warning, do the following:
```

```
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[10:41:55] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0,
the default evaluation metric used with the objective 'multi:softprob'
was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if
you'd like to restore the old behavior.

C:\Users\JiaYi\anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[10:42:24] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0,
the default evaluation metric used with the objective 'multi:softprob'
was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if
you'd like to restore the old behavior.

C:\Users\JiaYi\anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[10:42:51] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0,
the default evaluation metric used with the objective 'multi:softprob'
was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if
you'd like to restore the old behavior.
Average Accuracy: 0.8123956063928868
Average Precision: 0.8122806728460841
Average Recall: 0.8123934075475209
Average F1: 0.8075725628072515
```

D: Model Insights

After much tuning, let's view the summary of model results:

Conclusion: Highest precision of tuned XGBoost Classifier is 81%.

E: Predictions

```
d = {'danceability': [0.473, 0.55], 'energy': [0.23, 0.33],"loudness":
[-1,-9],
"acousticness":[0.33,0.01],"speechiness":[0.02,0.1],"valence":
[0.32,0.90],"tempo":[0.10,0.90],
"duration_in_min":[7,9],"time_signature":[3,4],"instrumentalness":
[0.23,0.78],"key":[3,4]}
df = pd.DataFrame(data=d)
xgb_best.fit(X_train,y_train)
df['class']=xgb_best.predict(df)
display(df)
```

```
C:\Users\JiaYi\anaconda3\lib\site-packages\xgboost\sklearn.py:1224:
UserWarning: The use of label encoder in XGBClassifier is deprecated
and will be removed in a future release. To remove this warning, do
the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\utils\
validation.py:63: DataConversionWarning: A column-vector y was passed
when a 1d array was expected. Please change the shape of y to
(n_samples, ), for example using ravel().
  return f(*args, **kwargs)
```

```
[11:24:16] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0,
the default evaluation metric used with the objective 'multi:softprob'
was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if
you'd like to restore the old behavior.
```

```
   danceability  energy  loudness  acousticness  speechiness  valence
tempo  \
0         0.473    0.23        -1          0.33         0.02     0.32
0.1
1         0.550    0.33        -9          0.01         0.10     0.90
0.9

   duration_in_min  time_signature  instrumentalness  key  class
0                7               3              0.23    3     10
1                9               4              0.78    4      2
```

Model 5: Gradient Boosting Classifier

A: Building Model - Gradient Boosting Classifier

```
from sklearn.ensemble import GradientBoostingClassifier
GBC=GradientBoostingClassifier()
GBC.fit(X_train, y_train)
```

```
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\utils\
validation.py:63: DataConversionWarning: A column-vector y was passed
when a 1d array was expected. Please change the shape of y to
(n_samples, ), for example using ravel().
  return f(*args, **kwargs)
```

```
GradientBoostingClassifier()
```

B: Model Evaluation

Cross Validation

```
#Enter your codes to use cross_validation here
from sklearn.model_selection import cross_validate

result = cross_validate(GBC, X, np.ravel(y), cv=14,
scoring=['accuracy', 'f1_macro', 'precision_macro', 'recall_macro'])
print("Average Accuracy:", np.mean(result['test_accuracy']))
print("Average Precision:", np.mean(result['test_precision_macro']))
print("Average Recall:", np.mean(result['test_recall_macro']))
print("Average F1:", np.mean(result['test_f1_macro']))
```

```
Average Accuracy: 0.5865564065663692
Average Precision: 0.5756074345990829
Average Recall: 0.586559766476198
Average F1: 0.5743339638877611
```

I will not be tuning this classifier as the base model already has a much lower performance than the base model of other algorithms. Hence, even with tuning, it may only increase a little bit but not enough to hit the 70% range of precision/accuracy/recall.

Also, since accuracy is low, I will not be using this model to do any predictions.

Conclusion: Highest precision of tuned Gradient Boosting Classifier is 58%.

Model 6: MultinomialNB

A: Building Model - MultinomialNB

```
from sklearn.naive_bayes import MultinomialNB

MNB = MultinomialNB()
MNB.fit(X_train, y_train)
```

```
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\utils\
validation.py:63: DataConversionWarning: A column-vector y was passed
when a 1d array was expected. Please change the shape of y to
(n_samples, ), for example using ravel().
  return f(*args, **kwargs)

MultinomialNB()
```

B: Model Evaluation

Cross Validation

```
#Enter your codes to use cross_validation here
from sklearn.model_selection import cross_validate

result = cross_validate(MNB, X, np.ravel(y), cv=14,
scoring=['accuracy', 'f1_macro', 'precision_macro', 'recall_macro'])
print("Average Accuracy:", np.mean(result['test_accuracy']))
print("Average Precision:", np.mean(result['test_precision_macro']))
print("Average Recall:", np.mean(result['test_recall_macro']))
print("Average F1:", np.mean(result['test_f1_macro']))

Average Accuracy: 0.4122731396913762
Average Precision: 0.3756781065570345
Average Recall: 0.4122744116096624
Average F1: 0.377663304597036
```

I will not be tuning this classifier as the base model already has a much lower performance than the base model of other algorithms. Hence, even with tuning, it may only increase a little bit but not enough to hit the 70% range of precision/accuracy/recall.

Also, since accuracy is low, I will not be using this model to do any predictions.

Conclusion: Highest precision of tuned Multinomial Naives Bayes Classifier is 38%.

Model 7: SGD Classifier

A: Building Model - SGD Classifier

```
from sklearn.linear_model import SGDClassifier
SGD = SGDClassifier()
SGD.fit(X_train, y_train)

C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\utils\
validation.py:63: DataConversionWarning: A column-vector y was passed
when a 1d array was expected. Please change the shape of y to
(n_samples, ), for example using ravel().
  return f(*args, **kwargs)
```

```
SGDClassifier()
```

B: Model Evaluation

Cross Validation

```
#Enter your codes to use cross_validation here
from sklearn.model_selection import cross_validate

result = cross_validate(SGD, X, np.ravel(y), cv=14,
scoring=['accuracy', 'f1_macro', 'precision_macro', 'recall_macro'])
print("Average Accuracy:", np.mean(result['test_accuracy']))
print("Average Precision:", np.mean(result['test_precision_macro']))
print("Average Recall:", np.mean(result['test_recall_macro']))
print("Average F1:", np.mean(result['test_f1_macro']))
```

```
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\metrics\
_classification.py:1245: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\metrics\
_classification.py:1245: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\metrics\
_classification.py:1245: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\metrics\
_classification.py:1245: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\metrics\
_classification.py:1245: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))

Average Accuracy: 0.42311169921626757
Average Precision: 0.4356611694349228
Average Recall: 0.42311530449147117
Average F1: 0.36351728042316084
```

I will not be tuning this classifier as the base model already has a much lower performance than the base model of other algorithms. Hence, even with tuning, it may only increase a little bit but not enough to hit the 70% range of precision/accuracy/recall.

Also, since accuracy is low, I will not be using this model to do any predictions.

Conclusion: Highest precision of tuned SGD Classifier is 44%.

Model 8:OneVsRest Classifier

A: Building Model - OneVsRest Classifier

```
from sklearn.multiclass import OneVsRestClassifier
from sklearn.svm import LinearSVC

ORC=OneVsRestClassifier(LinearSVC(random_state=0))
ORC.fit(X_train, y_train)
```

```
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985: ConvergenceWarning: Liblinear failed to converge, increase the number of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985: ConvergenceWarning: Liblinear failed to converge, increase the number of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985: ConvergenceWarning: Liblinear failed to converge, increase the number of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985: ConvergenceWarning: Liblinear failed to converge, increase the number of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985: ConvergenceWarning: Liblinear failed to converge, increase the number of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985: ConvergenceWarning: Liblinear failed to converge, increase the number of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985: ConvergenceWarning: Liblinear failed to converge, increase the number of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985: ConvergenceWarning: Liblinear failed to converge, increase the number of iterations.
  warnings.warn("Liblinear failed to converge, increase "
```

```
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "

OneVsRestClassifier(estimator=LinearSVC(random_state=0))
```

B: Model Evaluation

Cross Validation

```
#Enter your codes to use cross_validation here
from sklearn.model_selection import cross_validate

result = cross_validate(ORC, X, np.ravel(y), cv=14,
scoring=['accuracy', 'f1_macro', 'precision_macro', 'recall_macro'])
print("Average Accuracy:", np.mean(result['test_accuracy']))
print("Average Precision:", np.mean(result['test_precision_macro']))
print("Average Recall:", np.mean(result['test_recall_macro']))
print("Average F1:", np.mean(result['test_f1_macro']))
```

```
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
```

```
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
```

```
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
```

```
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "

C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
```

```
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
```

```
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\metrics\
```

```
_classification.py:1245: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
```

```
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985: 
ConvergenceWarning: Liblinear failed to converge, increase the number 
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985: 
ConvergenceWarning: Liblinear failed to converge, increase the number 
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985: 
ConvergenceWarning: Liblinear failed to converge, increase the number 
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985: 
ConvergenceWarning: Liblinear failed to converge, increase the number 
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985: 
ConvergenceWarning: Liblinear failed to converge, increase the number 
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985: 
ConvergenceWarning: Liblinear failed to converge, increase the number 
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985: 
ConvergenceWarning: Liblinear failed to converge, increase the number 
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985: 
ConvergenceWarning: Liblinear failed to converge, increase the number 
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985: 
ConvergenceWarning: Liblinear failed to converge, increase the number 
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985: 
ConvergenceWarning: Liblinear failed to converge, increase the number 
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985: 
ConvergenceWarning: Liblinear failed to converge, increase the number 
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985: 
ConvergenceWarning: Liblinear failed to converge, increase the number 
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985: 
ConvergenceWarning: Liblinear failed to converge, increase the number
```

```
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
```

```
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
```

```
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "

C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
```

```
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
```

```
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
```

```
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "

Average Accuracy: 0.4718432689072896
Average Precision: 0.4229064059627106
Average Recall: 0.4718399478207922
Average F1: 0.41048095739272616

C:\Users\JiaYi\anaconda3\lib\site-packages\sklearn\svm\_base.py:985:
ConvergenceWarning: Liblinear failed to converge, increase the number
of iterations.
  warnings.warn("Liblinear failed to converge, increase "
```

I will not be tuning this classifier as the base model already has a much lower performance than the base model of other algorithms. Hence, even with tuning, it may only increase a little bit but not enough to hit the 70% range of precision/accuracy/recall.

Also, since accuracy is low, I will not be using this model to do any predictions.

Conclusion: Highest precision of tuned One-Vs-Rest Classifier is 42%.

Model 9: Neural Network Model

A: Preparing Data for neural network model

```
from keras.utils import np_utils

encoder = LabelEncoder()
encoder.fit(y_train)
encoded_Y_train = encoder.transform(y_train)
# convert integers to dummy variables (i.e. one hot encoded)
dummy_y_train = np_utils.to_categorical(encoded_Y_train)
encoder = LabelEncoder()
encoder.fit(y_test)
encoded_Y_test = encoder.transform(y_test)
# convert integers to dummy variables (i.e. one hot encoded)
dummy_y_test = np_utils.to_categorical(encoded_Y_test)

dummy_y_train

array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 1., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 1., 0.],
```

```
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]], dtype=float32)
```

B: Building Model - Deep Neural Network Model

```
model1 = Sequential()
model1.add(Dense(512, input_shape = (X_train.shape[1],)))
model1.add(Activation('relu'))
model1.add(Dropout(0.1))

model1.add(Dense(512, input_shape = (X_train.shape[1],)))
model1.add(Activation('relu'))
model1.add(Dropout(0.1))

model1.add(Dense(512, input_shape = (X_train.shape[1],)))
model1.add(Activation('relu'))
model1.add(Dropout(0.1))


model1.add(Dense(11))
model1.add(Activation('softmax'))

model1.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

import keras
from keras.callbacks import EarlyStopping

# early stopping callback
# This callback will stop the training when there is no improvement in

# the validation loss for 10 consecutive epochs.
es = keras.callbacks.EarlyStopping(monitor='val_loss', mode='min',
                                   patience=10,
                                   restore_best_weights=True) #
important - otherwise you just return the last weigths...

# now we just update our model fit call
history = model1.fit(X_train,
                    dummy_y_train,
                    callbacks=[es],
                    epochs=8000000, # you can set this to a big
number!
                    batch_size=10,
                    shuffle=True,
                    validation_split=0.2,
                    verbose=1)
```

```
Epoch 1/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
1.6515 - accuracy: 0.4154 - val_loss: 1.4877 - val_accuracy: 0.4779
Epoch 2/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
1.4819 - accuracy: 0.4791 - val_loss: 1.4747 - val_accuracy: 0.4827
Epoch 3/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
1.4448 - accuracy: 0.4893 - val_loss: 1.3965 - val_accuracy: 0.5096
Epoch 4/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
1.4104 - accuracy: 0.5034 - val_loss: 1.3601 - val_accuracy: 0.5163
Epoch 5/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
1.3835 - accuracy: 0.5119 - val_loss: 1.3456 - val_accuracy: 0.5280
Epoch 6/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
1.3579 - accuracy: 0.5225 - val_loss: 1.3406 - val_accuracy: 0.5254
Epoch 7/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
1.3316 - accuracy: 0.5295 - val_loss: 1.3157 - val_accuracy: 0.5305
Epoch 8/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
1.3092 - accuracy: 0.5354 - val_loss: 1.2726 - val_accuracy: 0.5456
Epoch 9/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
1.2910 - accuracy: 0.5399 - val_loss: 1.2488 - val_accuracy: 0.5498
Epoch 10/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
1.2647 - accuracy: 0.5510 - val_loss: 1.2867 - val_accuracy: 0.5508
Epoch 11/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
1.2378 - accuracy: 0.5589 - val_loss: 1.2267 - val_accuracy: 0.5653
Epoch 12/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
1.2216 - accuracy: 0.5649 - val_loss: 1.2151 - val_accuracy: 0.5731
Epoch 13/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
1.1994 - accuracy: 0.5698 - val_loss: 1.1997 - val_accuracy: 0.5795
Epoch 14/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
1.1799 - accuracy: 0.5789 - val_loss: 1.1757 - val_accuracy: 0.5860
Epoch 15/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
1.1605 - accuracy: 0.5886 - val_loss: 1.1666 - val_accuracy: 0.5926
Epoch 16/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
1.1344 - accuracy: 0.5960 - val_loss: 1.1540 - val_accuracy: 0.5948
Epoch 17/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
```

```
1.1256 - accuracy: 0.6000 - val_loss: 1.1515 - val_accuracy: 0.5973
Epoch 18/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
1.1076 - accuracy: 0.6034 - val_loss: 1.1509 - val_accuracy: 0.5921
Epoch 19/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
1.0843 - accuracy: 0.6136 - val_loss: 1.1507 - val_accuracy: 0.6014
Epoch 20/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
1.0670 - accuracy: 0.6151 - val_loss: 1.1099 - val_accuracy: 0.6129
Epoch 21/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
1.0523 - accuracy: 0.6249 - val_loss: 1.0925 - val_accuracy: 0.6213
Epoch 22/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
1.0384 - accuracy: 0.6292 - val_loss: 1.1072 - val_accuracy: 0.6169
Epoch 23/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
1.0217 - accuracy: 0.6350 - val_loss: 1.1106 - val_accuracy: 0.6132
Epoch 24/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
1.0045 - accuracy: 0.6396 - val_loss: 1.0802 - val_accuracy: 0.6266
Epoch 25/8000000
2676/2676 [==============================] - 7s 2ms/step - loss:
0.9984 - accuracy: 0.6450 - val_loss: 1.0895 - val_accuracy: 0.6241
Epoch 26/8000000
2676/2676 [==============================] - 7s 3ms/step - loss:
0.9855 - accuracy: 0.6461 - val_loss: 1.0395 - val_accuracy: 0.6450
Epoch 27/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
0.9754 - accuracy: 0.6510 - val_loss: 1.0405 - val_accuracy: 0.6410
Epoch 28/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
0.9603 - accuracy: 0.6568 - val_loss: 1.0284 - val_accuracy: 0.6557
Epoch 29/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
0.9478 - accuracy: 0.6641 - val_loss: 1.0421 - val_accuracy: 0.6462
Epoch 30/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
0.9401 - accuracy: 0.6619 - val_loss: 1.0349 - val_accuracy: 0.6549
Epoch 31/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
0.9226 - accuracy: 0.6659 - val_loss: 1.0179 - val_accuracy: 0.6567
Epoch 32/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
0.9161 - accuracy: 0.6694 - val_loss: 1.0079 - val_accuracy: 0.6552
Epoch 33/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
0.9064 - accuracy: 0.6765 - val_loss: 0.9840 - val_accuracy: 0.6655
Epoch 34/8000000
```

```
2676/2676 [==============================] - 6s 2ms/step - loss:
0.8969 - accuracy: 0.6783 - val_loss: 1.0073 - val_accuracy: 0.6474
Epoch 35/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
0.8922 - accuracy: 0.6810 - val_loss: 1.0115 - val_accuracy: 0.6585
Epoch 36/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
0.8746 - accuracy: 0.6877 - val_loss: 0.9840 - val_accuracy: 0.6682
Epoch 37/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
0.8644 - accuracy: 0.6915 - val_loss: 0.9875 - val_accuracy: 0.6694
Epoch 38/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
0.8575 - accuracy: 0.6920 - val_loss: 0.9869 - val_accuracy: 0.6731
Epoch 39/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
0.8600 - accuracy: 0.6914 - val_loss: 0.9869 - val_accuracy: 0.6681
Epoch 40/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
0.8470 - accuracy: 0.6957 - val_loss: 0.9770 - val_accuracy: 0.6770
Epoch 41/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
0.8359 - accuracy: 0.7007 - val_loss: 0.9929 - val_accuracy: 0.6731
Epoch 42/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
0.8332 - accuracy: 0.7025 - val_loss: 0.9851 - val_accuracy: 0.6778
Epoch 43/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
0.8231 - accuracy: 0.7069 - val_loss: 0.9549 - val_accuracy: 0.6914
Epoch 44/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
0.8172 - accuracy: 0.7021 - val_loss: 0.9626 - val_accuracy: 0.6815
Epoch 45/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
0.8003 - accuracy: 0.7139 - val_loss: 0.9498 - val_accuracy: 0.6845
Epoch 46/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
0.8053 - accuracy: 0.7110 - val_loss: 0.9573 - val_accuracy: 0.6842
Epoch 47/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
0.7983 - accuracy: 0.7136 - val_loss: 0.9563 - val_accuracy: 0.6869
Epoch 48/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
0.7886 - accuracy: 0.7192 - val_loss: 0.9803 - val_accuracy: 0.6832
Epoch 49/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
0.7816 - accuracy: 0.7201 - val_loss: 0.9485 - val_accuracy: 0.6853
Epoch 50/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
0.7751 - accuracy: 0.7202 - val_loss: 0.9672 - val_accuracy: 0.6866
```

```
Epoch 51/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
0.7820 - accuracy: 0.7163 - val_loss: 0.9264 - val_accuracy: 0.6957
Epoch 52/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
0.7719 - accuracy: 0.7247 - val_loss: 0.9907 - val_accuracy: 0.6871
Epoch 53/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
0.7647 - accuracy: 0.7276 - val_loss: 0.9570 - val_accuracy: 0.6909
Epoch 54/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
0.7617 - accuracy: 0.7285 - val_loss: 0.9453 - val_accuracy: 0.6929
Epoch 55/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
0.7596 - accuracy: 0.7304 - val_loss: 0.9758 - val_accuracy: 0.6939
Epoch 56/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
0.7487 - accuracy: 0.7314 - val_loss: 0.9327 - val_accuracy: 0.7090
Epoch 57/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
0.7469 - accuracy: 0.7345 - val_loss: 0.9513 - val_accuracy: 0.6974
Epoch 58/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
0.7355 - accuracy: 0.7373 - val_loss: 0.9387 - val_accuracy: 0.6999
Epoch 59/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
0.7287 - accuracy: 0.7407 - val_loss: 0.9332 - val_accuracy: 0.7020
Epoch 60/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
0.7266 - accuracy: 0.7428 - val_loss: 0.9360 - val_accuracy: 0.7038
Epoch 61/8000000
2676/2676 [==============================] - 6s 2ms/step - loss:
0.7323 - accuracy: 0.7377 - val_loss: 0.9508 - val_accuracy: 0.6936
```

```
model1.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_4 (Dense)             (None, 512)               6144

 activation_4 (Activation)   (None, 512)               0

 dropout_3 (Dropout)         (None, 512)               0

 dense_5 (Dense)             (None, 512)               262656

 activation_5 (Activation)   (None, 512)               0
```

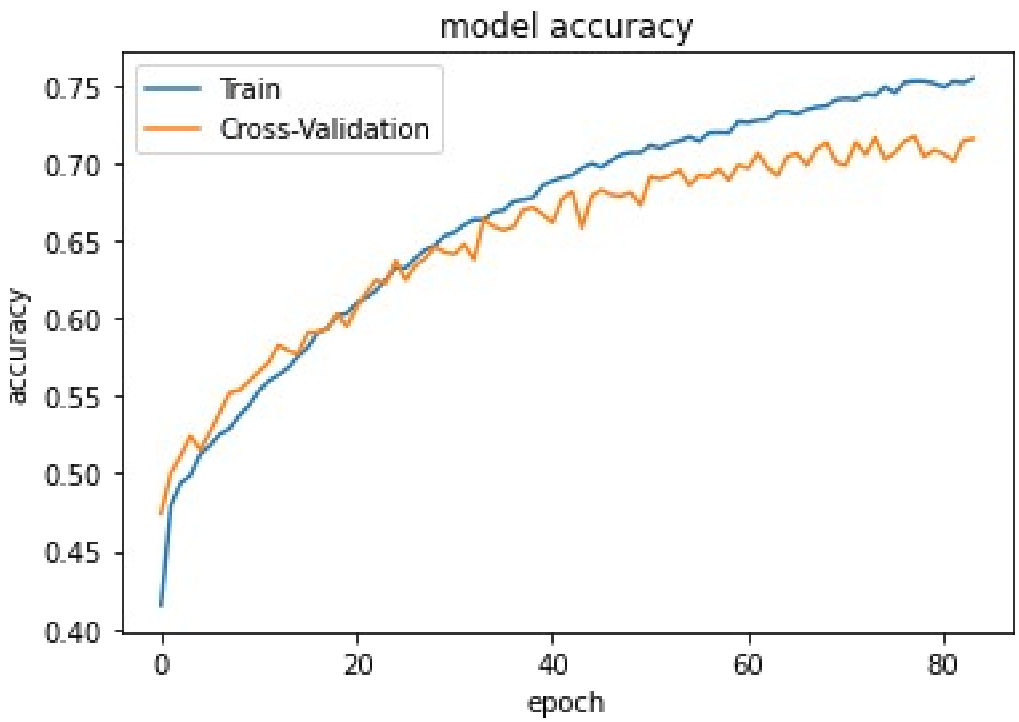
```
 dropout_4 (Dropout)         (None, 512)               0

 dense_6 (Dense)             (None, 512)               262656

 activation_6 (Activation)   (None, 512)               0

 dropout_5 (Dropout)         (None, 512)               0

 dense_7 (Dense)             (None, 11)                5643

 activation_7 (Activation)   (None, 11)                0

=================================================================
Total params: 537,099
Trainable params: 537,099
Non-trainable params: 0
_________________________________________________________________
```

C: Model Evaluation

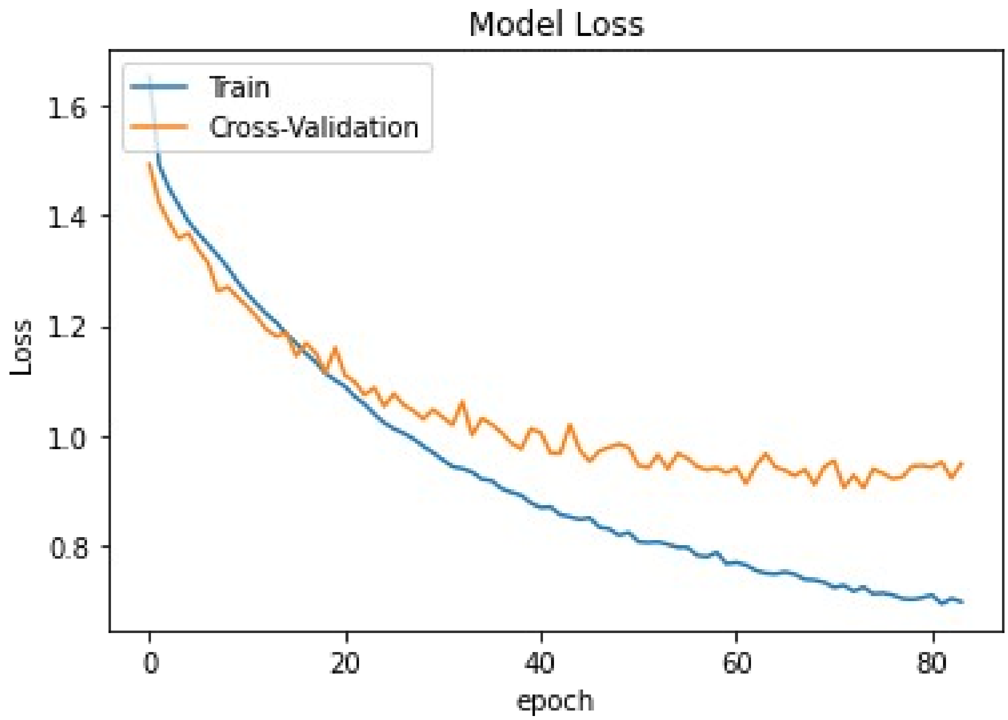
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

preds = model1.predict(X_test) # see how the model did!
print(classification_report(dummy_y_test.argmax(axis=1),
preds.argmax(axis=1)))

              precision    recall  f1-score   support

           0       0.67      0.80      0.73       760
           1       0.69      0.67      0.68       760
           2       0.64      0.63      0.64       760
           3       0.88      0.95      0.91       760
           4       0.76      0.93      0.83       760
           5       0.75      0.86      0.80       760
           6       0.59      0.38      0.46       760
           7       0.93      0.97      0.95       760
           8       0.73      0.86      0.79       760
           9       0.57      0.49      0.53       760
          10       0.46      0.31      0.37       760

    accuracy                           0.71      8360
   macro avg       0.70      0.71      0.70      8360
weighted avg       0.70      0.71      0.70      8360


accuracy = model1.evaluate(X_test, dummy_y_test, verbose=2)
print("Accuracy:",accuracy[1]*100)
```

```
262/262 - 0s - loss: 0.9021 - accuracy: 0.7145 - 158ms/epoch -
603us/step
Accuracy: 71.44736647605896
```

```python
from matplotlib import pyplot as plt
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['Train', 'Cross-Validation'], loc='upper left')
plt.show()
```




```python
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('epoch')
plt.legend(['Train', 'Cross-Validation'], loc='upper left')
plt.show()
```



E: Predictions

```
d = {'danceability': [0.473, 0.55], 'energy': [0.23, 0.33],"loudness":
[-1,-9],
"acousticness":[0.33,0.01],"speechiness":[0.02,0.1],"valence":
[0.32,0.90],"tempo":[0.10,0.90],
"duration_in_min":[7,9],"time_signature":[3,4],"instrumentalness":
[0.23,0.78],"key":[3,4]}
df = pd.DataFrame(data=d)
df['class']=model1.predict(df).argmax(axis=1)
display(df)

   danceability  energy  loudness  acousticness  speechiness  valence
tempo  \
0         0.473    0.23        -1          0.33         0.02     0.32
0.1
1         0.550    0.33        -9          0.01         0.10     0.90
0.9

   duration_in_min  time_signature  instrumentalness  key  class
0                7               3              0.23    3      2
1                9               4              0.78    4      2
```

Conclusion: Highest precision of Deep Neural Network Model is 70%

5. Model Comparisons

Let's compare the summary of all model results:

Conclusion: The best model to be used is XGBoost Classifier with the highest accuracy of 81%, precision of 81%, recall 0f 81% and f1-score of 81%.

6. Conclusion

For this report, I have gone through the whole data pipeline from data understanding, preparation, exploratory data analysis, modelling, evaluation, prediction and finally comparison of models. To optimize the performance of the model, I have tried many methodologies and switched the order of the data preparations and modelling preparation steps. The insights derived from a wide variety of models also contributed greatly to the overall objective, which is to classify songs by music characteristics. It is evident that different models with different parameters can indeed affect model performance. Additionally, the more models tested, the higher probability of finding a good model for our predictions. I hope this report will provide great value to you and assist you to find the best model that will derive the most accurate and precise predictions.