```
import warnings
warnings.simplefilter("ignore", UserWarning)

!pip install xgboost -U
!pip install librosa -U
!pip install hyperopt

import os from tqdm import
tqdm import pickle import
scipy import sys import
pandas as pd import numpy as
np import IPython import
seaborn as sns import
tensorflow as tf import
matplotlib.pyplot as plt
%matplotlib inline
import librosa #Python package for music & audio
files import librosa.display import librosa.display
as lplt

from tensorflow import keras from keras.models import Sequential from
IPython.display import Audio from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder from sklearn.preprocessing
import normalize from sklearn.feature_selection import
RFECV,mutual_info_regression from sklearn.metrics import confusion_matrix,
accuracy_score,classification_report from sklearn.model_selection import
train_test_split, cross_val_score from sklearn.decomposition import PCA from
xgboost import XGBClassifier
from hyperopt import STATUS_OK, Trials, fmin, hp, tpe
```

```
Requirement already satisfied: xgboost in /usr/local/lib/python3.10/dist-packages (2.0.3)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from xgboost) (1.25.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from xgboost) (1.11.4)
Requirement already satisfied: librosa in /usr/local/lib/python3.10/dist-packages (0.10.1)
Requirement already satisfied: audioread>=2.1.9 in /usr/local/lib/python3.10/dist-packages (from librosa) (3.0.1)
Requirement already satisfied: numpy!=1.22.0,!=1.22.1,!=1.22.2,>=1.20.3 in /usr/local/lib/python3.10/dist-packages (from librosa) (1.25.2)
Requirement already satisfied: scipy>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from librosa) (1.11.4)
Requirement already satisfied: scikit-learn>=0.20.0 in /usr/local/lib/python3.10/dist-packages (from librosa) (1.2.2)
Requirement already satisfied: joblib>=0.14 in /usr/local/lib/python3.10/dist-packages (from librosa) (1.4.0)
Requirement already satisfied: decorator>=4.3.0 in /usr/local/lib/python3.10/dist-packages (from librosa) (4.4.2)
Requirement already satisfied: numba>=0.51.0 in /usr/local/lib/python3.10/dist-packages (from librosa) (0.58.1)
Requirement already satisfied: soundfile>=0.12.1 in /usr/local/lib/python3.10/dist-packages (from librosa) (0.12.1)
Requirement already satisfied: pooch>=1.0 in /usr/local/lib/python3.10/dist-packages (from librosa) (1.8.1)
Requirement already satisfied: soxr>=0.3.2 in /usr/local/lib/python3.10/dist-packages (from librosa) (0.3.7)
Requirement already satisfied: typing-extensions>=4.1.1 in /usr/local/lib/python3.10/dist-packages (from librosa) (4.11.0)
Requirement already satisfied: lazy-loader>=0.1 in /usr/local/lib/python3.10/dist-packages (from librosa) (0.4)
Requirement already satisfied: msgpack>=1.0 in /usr/local/lib/python3.10/dist-packages (from librosa) (1.0.8)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from lazy-loader>=0.1->librosa) (24.0)
```

```
Requirement already satisfied: llvmlite<0.42,>=0.41.0dev0 in /usr/local/lib/python3.10/dist-packages (from numba>=0.51.0->librosa) (0.41.1)
Requirement already satisfied: platformdirs>=2.5.0 in /usr/local/lib/python3.10/dist-packages (from pooch>=1.0->librosa) (4.2.0)
Requirement already satisfied: requests>=2.19.0 in /usr/local/lib/python3.10/dist-packages (from pooch>=1.0->librosa) (2.31.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.20.0->librosa)
(3.4.0)
Requirement already satisfied: cffi>=1.0 in /usr/local/lib/python3.10/dist-packages (from soundfile>=0.12.1->librosa) (1.16.0)
Requirement already satisfied: pycparser in /usr/local/lib/python3.10/dist-packages (from cffi>=1.0->soundfile>=0.12.1->librosa) (2.22)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->pooch>=1.0->librosa)
(3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->pooch>=1.0->librosa) (3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->pooch>=1.0->librosa) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->pooch>=1.0->librosa)
(2024.2.2)
Requirement already satisfied: hyperopt in /usr/local/lib/python3.10/dist-packages (0.2.7)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from hyperopt) (1.25.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from hyperopt) (1.11.4)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from hyperopt) (1.16.0)
Requirement already satisfied: networkx>=2.2 in /usr/local/lib/python3.10/dist-packages (from hyperopt) (3.3)
Requirement already satisfied: future in /usr/local/lib/python3.10/dist-packages (from hyperopt) (0.18.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from hyperopt) (4.66.2)
Requirement already satisfied: cloudpickle in /usr/local/lib/python3.10/dist-packages (from hyperopt) (2.2.1)
Requirement already satisfied: py4j in /usr/local/lib/python3.10/dist-packages (from hyperopt) (0.10.9.7)
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
final_data = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/Data/features_3_sec.csv")
final_data.head()
```

	filename	length	chroma_stft_mean	chroma_stft_var	rms_mean	rms_var	spectral_centroid_mean	spectral_centroid_var
0	blues.00000.0.wav	66149	0.335406	0.091048	0.130405	0.003521	1773.065032	167541.630869
1	blues.00000.1.wav	66149	0.343065	0.086147	0.112699	0.001450	1816.693777	90525.690866
2	blues.00000.2.wav	66149	0.346815	0.092243	0.132003	0.004620	1788.539719	111407.437613
3	blues.00000.3.wav	66149	0.363639	0.086856	0.132565	0.002448	1655.289045	111952.284517
4	blues.00000.4.wav	66149	0.335579	0.088129	0.143289	0.001701	1630.656199	79667.267654

5 rows × 60 columns

```
final_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 9990 entries, 0 to 9989
Data columns (total 60 columns):
```

```
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   filename                9990 non-null   object
 1   length                  9990 non-null   int64
 2   chroma_stft_mean        9990 non-null   float64
 3   chroma_stft_var         9990 non-null   float64
 4   rms_mean                9990 non-null   float64
 5   rms_var                 9990 non-null   float64
 6   spectral_centroid_mean  9990 non-null   float64
 7   spectral_centroid_var   9990 non-null   float64
 8   spectral_bandwidth_mean 9990 non-null   float64
 9   spectral_bandwidth_var  9990 non-null   float64
 10  rolloff_mean            9990 non-null   float64
 11  rolloff_var             9990 non-null   float64
 12  zero_crossing_rate_mean 9990 non-null   float64
 13  zero_crossing_rate_var  9990 non-null   float64
 14  harmony_mean            9990 non-null   float64
 15  harmony_var             9990 non-null   float64
 16  perceptr_mean           9990 non-null   float64
 17  perceptr_var            9990 non-null   float64
 18  tempo                   9990 non-null   float64
 19  mfcc1_mean              9990 non-null   float64
 20  mfcc1_var               9990 non-null   float64
 21  mfcc2_mean              9990 non-null   float64
 22  mfcc2_var               9990 non-null   float64
 23  mfcc3_mean              9990 non-null   float64
 24  mfcc3_var               9990 non-null   float64
 25  mfcc4_mean              9990 non-null   float64
 26  mfcc4_var               9990 non-null   float64
 27  mfcc5_mean              9990 non-null   float64
 28  mfcc5_var               9990 non-null   float64
 29  mfcc6_mean              9990 non-null   float64
 30  mfcc6_var               9990 non-null   float64
 31  mfcc7_mean              9990 non-null   float64
 32  mfcc7_var               9990 non-null   float64
 33  mfcc8_mean              9990 non-null   float64
 34  mfcc8_var               9990 non-null   float64
 35  mfcc9_mean              9990 non-null   float64
 36  mfcc9_var               9990 non-null   float64
 37  mfcc10_mean             9990 non-null   float64
 38  mfcc10_var              9990 non-null   float64
 39  mfcc11_mean             9990 non-null   float64
 40  mfcc11_var              9990 non-null   float64
 41  mfcc12_mean             9990 non-null   float64
 42  mfcc12_var              9990 non-null   float64
 43  mfcc13_mean             9990 non-null   float64
 44  mfcc13_var              9990 non-null   float64
 45  mfcc14_mean             9990 non-null   float64
 46  mfcc14_var              9990 non-null   float64
 47  mfcc15_mean             9990 non-null   float64
 48  mfcc15_var              9990 non-null   float64
 49  mfcc16_mean             9990 non-null   float64
 50  mfcc16_var              9990 non-null   float64
 51  mfcc17_mean             9990 non-null   float64
 52  mfcc17_var              9990 non-null   float64
```

```
final_data.shape
```

```
(9990, 60)
```

```
final_data.dtypes
```

```
filename                  object
length                     int64
chroma_stft_mean         float64
chroma_stft_var          float64
rms_mean                 float64
rms_var                  float64
spectral_centroid_mean   float64
spectral_centroid_var    float64
spectral_bandwidth_mean  float64
spectral_bandwidth_var   float64
rolloff_mean             float64
rolloff_var              float64
zero_crossing_rate_mean  float64
zero_crossing_rate_var   float64
harmony_mean             float64
harmony_var              float64
perceptr_mean            float64
perceptr_var             float64
tempo                    float64
mfcc1_mean               float64
mfcc1_var                float64
mfcc2_mean               float64
mfcc2_var                float64
mfcc3_mean               float64
mfcc3_var                float64
mfcc4_mean               float64
mfcc4_var                float64
mfcc5_mean               float64
mfcc5_var                float64
mfcc6_mean               float64
mfcc6_var                float64
mfcc7_mean               float64
mfcc7_var                float64
mfcc8_mean               float64
mfcc8_var                float64
mfcc9_mean               float64
mfcc9_var                float64
mfcc10_mean              float64
mfcc10_var               float64
mfcc11_mean              float64
mfcc11_var               float64
mfcc12_mean              float64
mfcc12_var               float64
mfcc13_mean              float64
mfcc13_var               float64
mfcc14_mean              float64
mfcc14_var               float64
mfcc15_mean              float64
```

```
mfcc15_var                float64
mfcc16_mean               float64
mfcc16_var                float64
mfcc17_mean               float64
mfcc17_var                float64
mfcc18_mean               float64
mfcc18_var                float64
mfcc19_mean               float64
mfcc19_var                float64
mfcc20 mean               float64
```

```
final_data = final_data.drop(labels='filename',axis=1)
```

```
audio = "/content/drive/MyDrive/Colab Notebooks/Data/genres_original/classical/classical.00035.wav"
#Load & decode the audio as a time series, where sr represents the sampling rate
data , sr = librosa.load(audio)
print(type(data), type(sr))
```

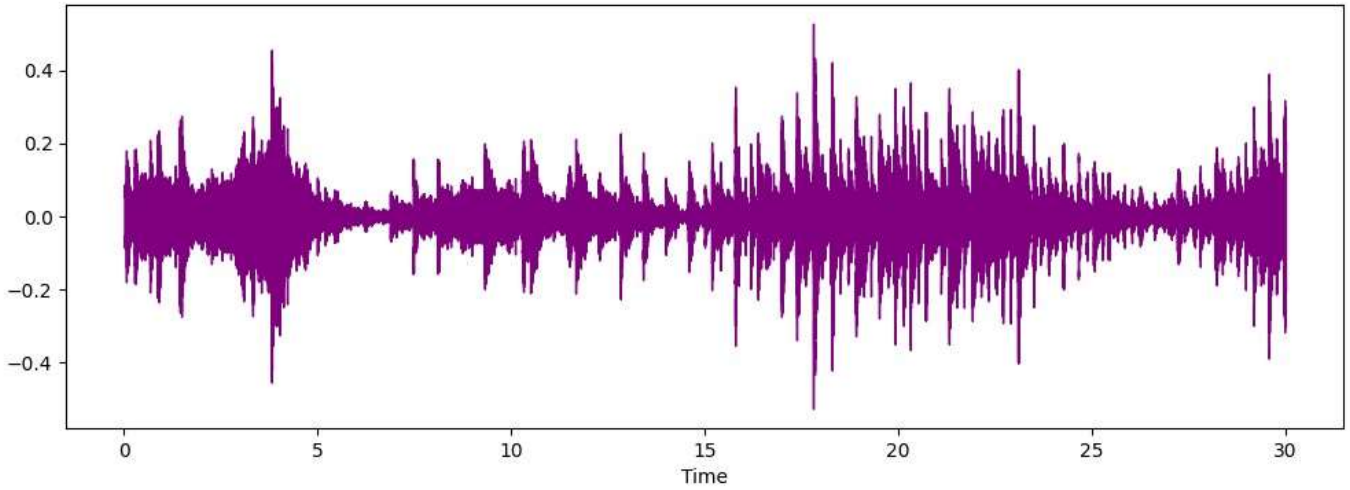
```
<class 'numpy.ndarray'> <class 'int'>
```

```
librosa.load(audio, sr=45600)
```

```
(array([0.00896395, 0.01348585, 0.01432906, ..., 0.01875842, 0.01658624,
        0.00845782], dtype=float32),
 45600)
```

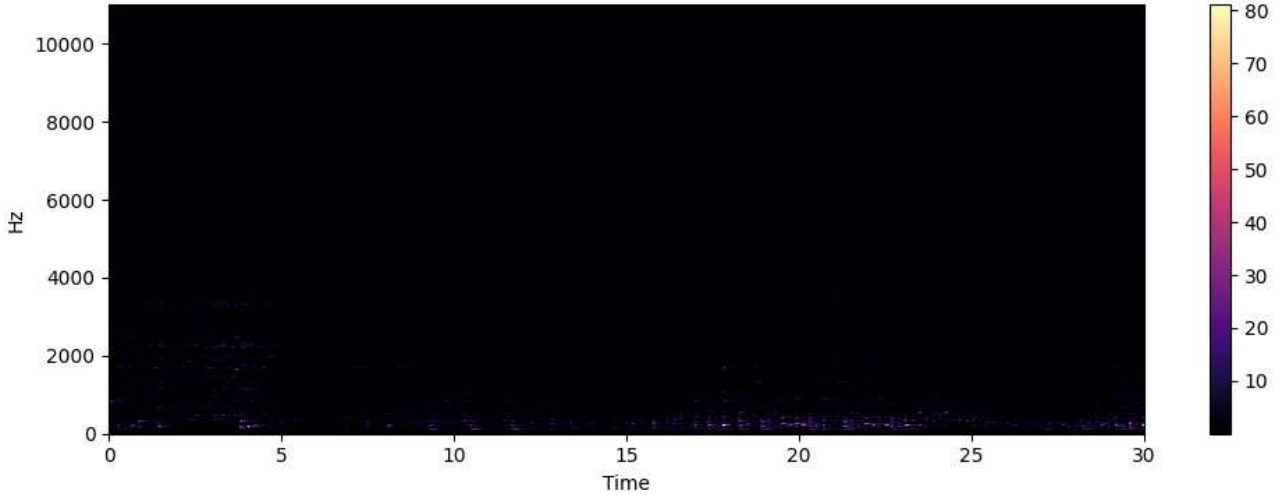
```
IPython.display.Audio(data, rate=sr)
```

```
plt.figure(figsize=(12,4))
librosa.display.waveshow(data, color = "Purple")
plt.show()
```




```
#Spectrogram stft = librosa.stft(data) stft_db =
librosa.amplitude_to_db(abs(stft)) plt.figure(figsize=(12,4))
librosa.display.specshow(stft, sr=sr, x_axis='time',
y_axis='hz') plt.colorbar()
```

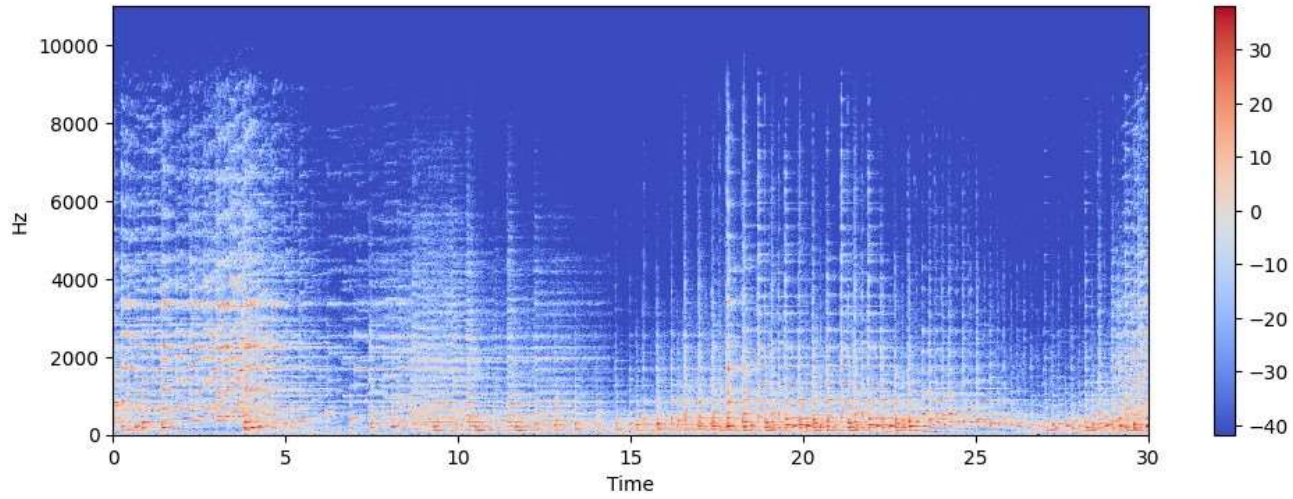
```
<matplotlib.colorbar.Colorbar at 0x7ce5c26f4610>
```



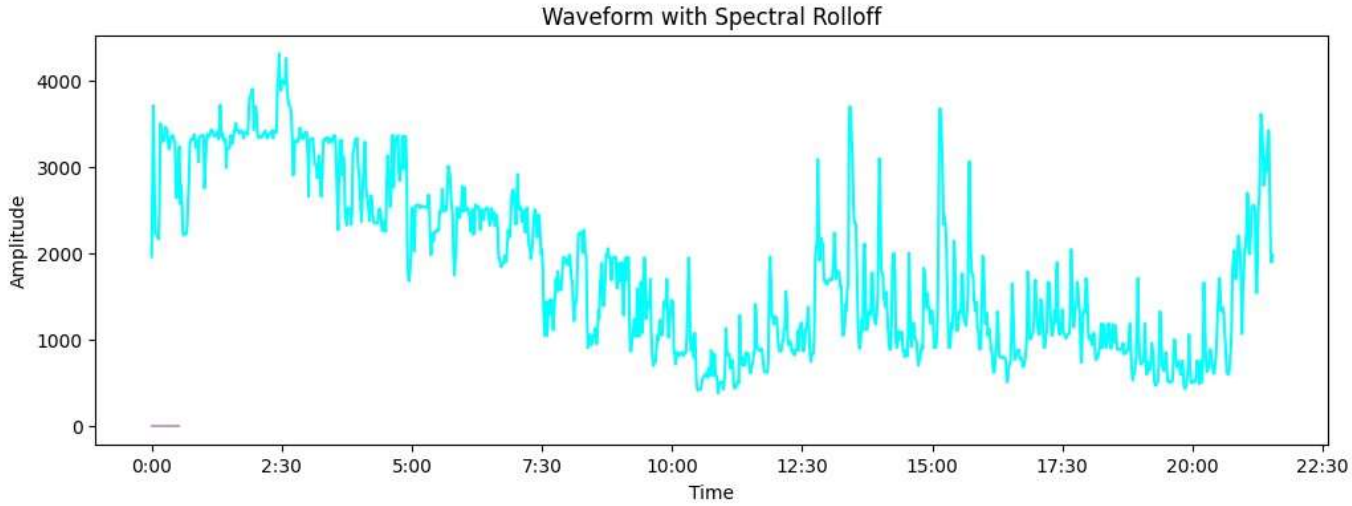

```
stft = librosa.stft(data) stft_db =
librosa.amplitude_to_db(abs(stft))
plt.figure(figsize=(12,4))
```

```
librosa.display.specshow(stft_db, sr=sr, x_axis='time', y_axis='hz')
plt.colorbar()
```

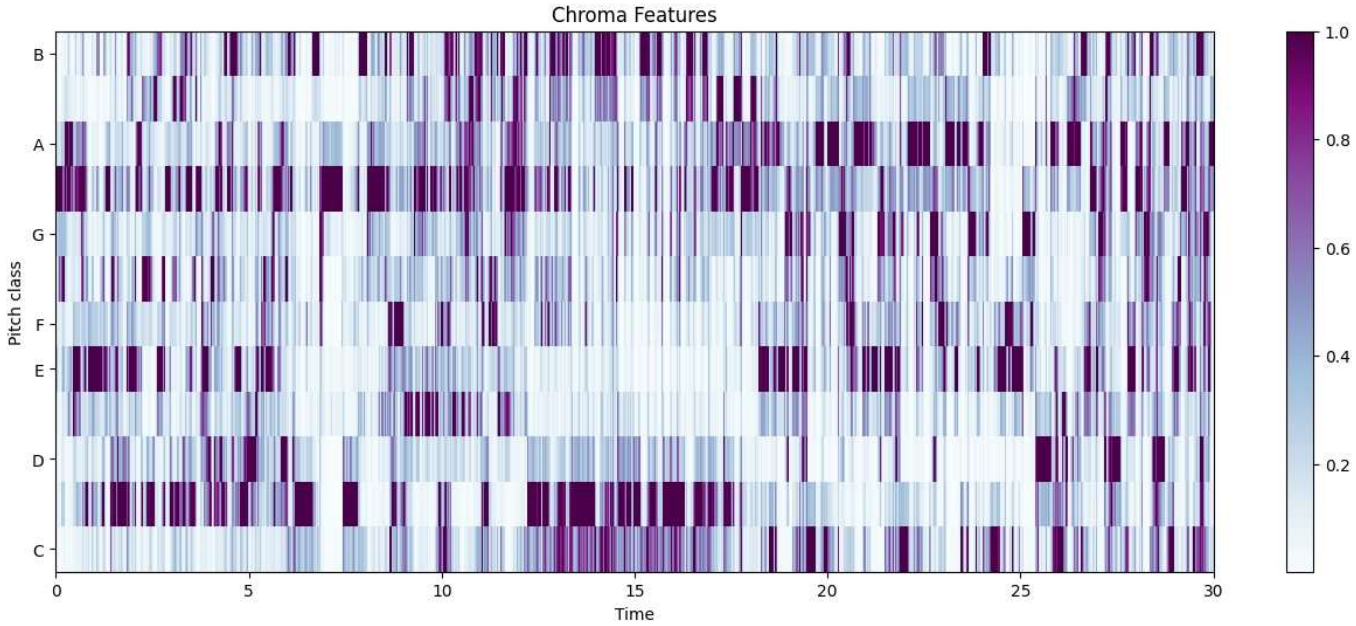
```
<matplotlib.colorbar.Colorbar at 0x7ce5c26aa440>
```




```
spectral_rolloff = librosa.feature.spectral_rolloff(y=data,
sr=sr)[0] plt.figure(figsize=(12,4))
librosa.display.waveshow(data, sr=sr, alpha=0.4, color='Purple')
plt.plot(spectral_rolloff, color='cyan') plt.title('Waveform with
Spectral Rolloff') plt.xlabel('Time') plt.ylabel('Amplitude')
plt.show()
```



```
spectral_rolloff = librosa.feature.spectral_rolloff(y=data, sr=sr)[0]


chroma = librosa.feature.chroma_stft(y=data, sr=sr)
plt.figure(figsize=(16,6)) lplt.specshow(chroma, sr=sr, x_axis='time',
y_axis='chroma', cmap='BuPu') plt.colorbar() plt.title("Chroma
Features") plt.show()
```




```
start = 1000 end = 1200
plt.figure(figsize=(14,5))
plt.plot(data[start:end],
color='Purple') plt.grid()
```


```
zero_cross_rate = librosa.zero_crossings(data[start:end], pad=False)
print("The number of zero crossings is:",sum(zero_cross_rate))
```

```
The number of zero crossings is: 27
```

```
class_list = final_data.iloc[:, -1]
convertor = LabelEncoder()
```

```
y = convertor.fit_transform(class_list)
```

```
y array([0, 0, 0, ..., 9, 9, 9])
```

```
print(final_data.iloc[:, :-1])
```

```
      length  chroma_stft_mean  chroma_stft_var  rms_mean   rms_var
\ 0      66149          0.335406         0.091048  0.130405  0.003521
1      66149          0.343065         0.086147  0.112699  0.001450
2      66149          0.346815         0.092243  0.132003  0.004620
3      66149          0.363639         0.086856  0.132565  0.002448
4      66149          0.335579         0.088129  0.143289  0.001701   ...      ...              ...            ...       ...       ...
9985   66149          0.349126         0.080515  0.050019  0.000097
9986   66149          0.372564         0.082626  0.057897  0.000088
```

```
9987   66149          0.347481         0.089019  0.052403  0.000701
9988   66149          0.387527         0.084815  0.066430  0.000320
9989   66149          0.369293         0.086759  0.050524  0.000067

      spectral_centroid_mean  spectral_centroid_var  spectral_bandwidth_mean  \
0                1773.065032          167541.630869              1972.744388
1                1816.693777           90525.690866              2010.051501
2                1788.539719          111407.437613              2084.565132
3                1655.289045          111952.284517              1960.039988
4                1630.656199           79667.267654              1948.503884
...                      ...                    ...                      ...
9985             1499.083005          164266.886443              1718.707215
9986             1847.965128          281054.935973              1906.468492
9987             1346.157659          662956.246325              1561.859087
9988             2084.515327          203891.039161              2018.366254
9989             1634.330126          411429.169769              1867.422378

      spectral_bandwidth_var  rolloff_mean  ...  mfcc16_mean  mfcc16_var  \
0              117335.771563   3714.560359  ...    -2.853603   39.687145
1               65671.875673   3869.682242  ...     4.074709   64.748276
2               75124.921716   3997.639160  ...     4.806280   67.336563
3               82913.639269   3568.300218  ...    -1.359111   47.739452
4               60204.020268   3469.992864  ...     2.092937   30.336359
...                      ...           ...  ...          ...         ...
9985            85931.574523   3015.559458  ...     5.773784   42.485981
9986            99727.037054   3746.694524  ...     2.074155   32.415203
9987           138762.841945   2442.362154  ...    -1.005473   78.228149
9988            22860.992562   4313.266226  ...     4.123402   28.323744
9989           119722.211518   3462.042142  ...     1.342274   38.801735

      mfcc17_mean  mfcc17_var  mfcc18_mean  mfcc18_var  mfcc19_mean  \
0       -3.241280   36.488243     0.722209   38.099152    -5.050335
1       -6.055294   40.677654     0.159015   51.264091    -2.837699
2       -1.768610   28.348579     2.378768   45.717648    -1.938424
3       -3.841155   28.337118     1.218588   34.770935    -3.580352
4        0.664582   45.880913     1.689446   51.363583    -3.392489
...           ...         ...          ...         ...          ...
9985    -9.094270   38.326839    -4.246976   31.049839    -5.625813
9986   -12.375726   66.418587    -3.081278   54.414265   -11.960546
9987    -2.524483   21.778994     4.809936   25.980829     1.775686
9988    -5.363541   17.209942     6.462601   21.442928     2.354765
9989   -11.598399   58.983097    -0.178517   55.761299    -6.903252

      mfcc19_var  mfcc20_mean  mfcc20_var
0      33.618073    -0.243027   43.771767
1      97.030830     5.784063   59.943081
2      53.050835     2.517375   33.105122
3      50.836224     3.630866   32.023678
4      26 738789     0 536961   29 146694
```

```
fit = StandardScaler()
X = fit.fit_transform(np.array(final_data.iloc[:, :-1], dtype = float))


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

```
len(y_train)
```

```
6693
```

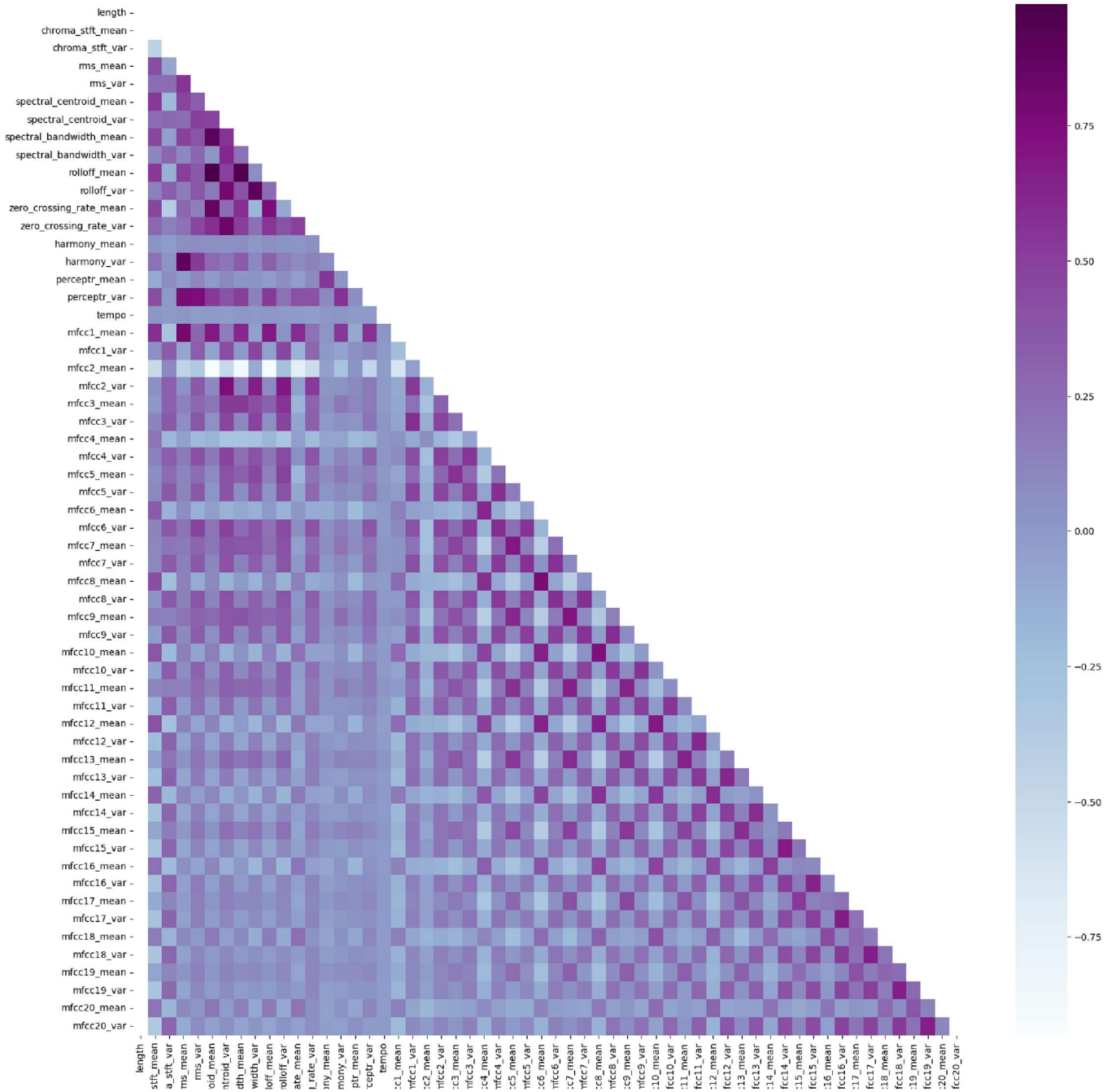
```
len(y_test)
```

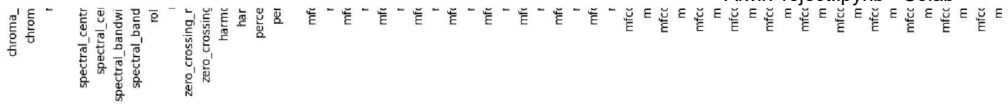
```
3297
```

```
def trainModel(model, epochs, optimizer):
batch_size = 128     model.compile(optimizer=optimizer,
loss='sparse_categorical_crossentropy',
metrics='accuracy'
    )     return model.fit(X_train, y_train, validation_data=(X_test, y_test),
epochs=epochs,                       batch_size=batch_size)
```

```
def plotValidate(history):
    print("Validation
Accuracy",max(history.history["val_accuracy"]))
pd.DataFrame(history.history).plot(figsize=(12,6))     plt.show()
```

```
# Check for non-numeric columns non_numeric_cols =
final_data.select_dtypes(exclude=np.number).columns

# Remove non-numeric columns final_data_numeric =
final_data.drop(columns=non_numeric_cols)

# Compute correlation matrix
corr =
final_data_numeric.corr()

# Create a mask for the heatmap mask
= np.zeros_like(corr)
mask[np.triu_indices_from(mask)] =
True

# Plot heatmap
plt.subplots(figsize=(20, 20))
sns.heatmap(corr, mask=mask,
cmap="BuPu") plt.show()
```


length
chroma_stft_mean
chroma_stft_var
rms_mean
rms_var
spectral_centroid_mean
spectral_centroid_var
spectral_bandwidth_mean
spectral_bandwidth_var
rolloff_mean
rolloff_var
zero_crossing_rate_mean
zero_crossing_rate_var
harmony_mean
harmony_var
perceptr_mean
perceptr_var
tempo
mfcc1_mean
mfcc1_var
mfcc2_mean
mfcc2_var
mfcc3_mean
mfcc3_var
mfcc4_mean
mfcc4_var
mfcc5_mean
mfcc5_var
mfcc6_mean
mfcc6_var
mfcc7_mean
mfcc7_var
mfcc8_mean
mfcc8_var
mfcc9_mean
mfcc9_var
mfcc10_mean
mfcc10_var
mfcc11_mean
mfcc11_var
mfcc12_mean
mfcc12_var
mfcc13_mean
mfcc13_var
mfcc14_mean
mfcc14_var
mfcc15_mean
mfcc15_var
mfcc16_mean
mfcc16_var
mfcc17_mean
mfcc17_var
mfcc18_mean
mfcc18_var
mfcc19_mean
mfcc19_var
mfcc20_mean
mfcc20_var
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75



```
model = XGBClassifier(n_estimators=1000)
model.fit(X_train,y_train,eval_metric='merror')
```

XGBClassifier

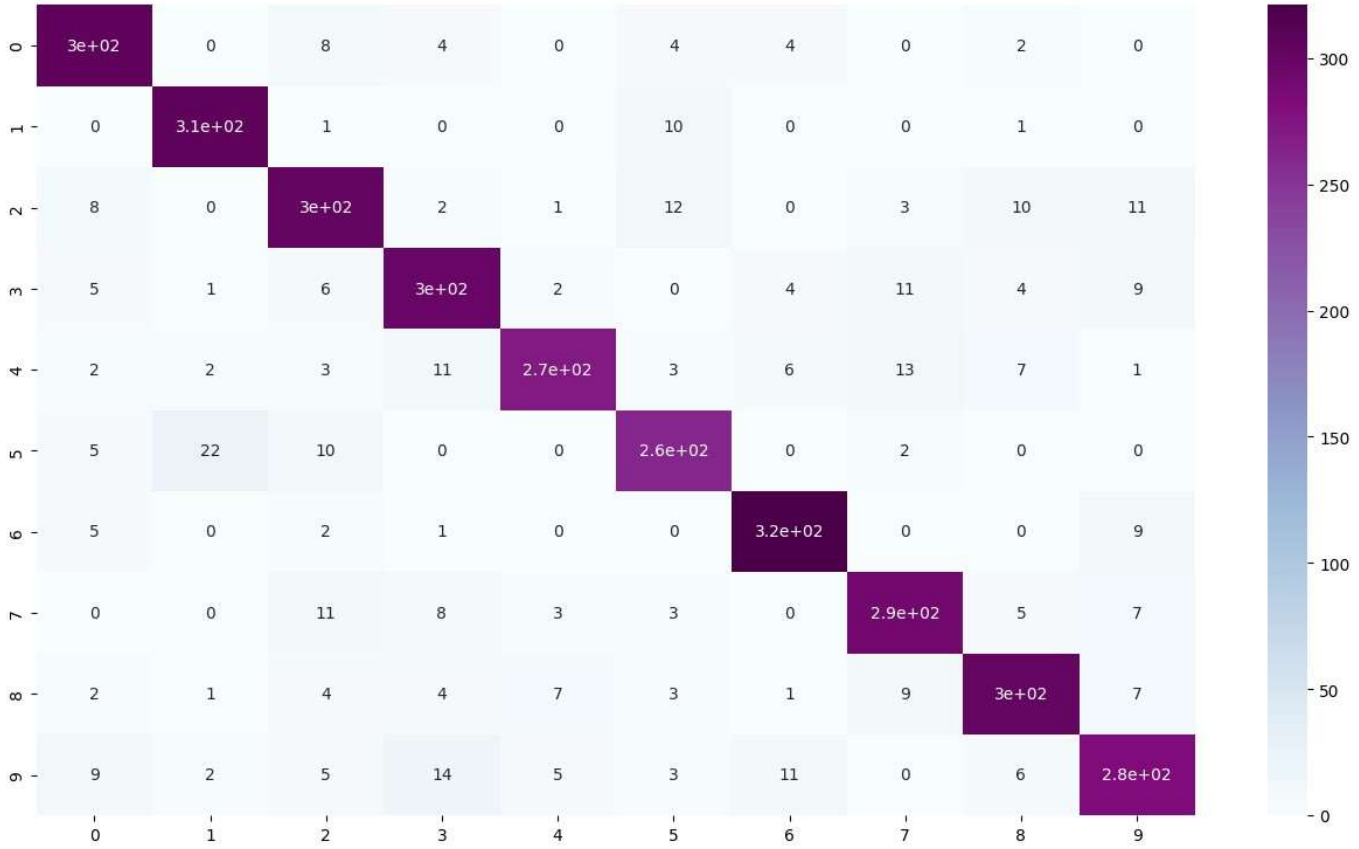
```
XGBClassifier(base_score=None, booster=None, callbacks=None,
colsample_bylevel=None, colsample_bynode=None,
colsample_bytree=None, device=None, early_stopping_rounds=None,
enable_categorical=False, eval_metric=None, feature_types=None,
gamma=None, grow_policy=None, importance_type=None,
interaction_constraints=None, learning_rate=None, max_bin=None,
max_cat_threshold=None, max_cat_to_onehot=None,
max_delta_step=None, max_depth=None, max_leaves=None,
min_child_weight=None, missing=nan, monotone_constraints=None,
multi_strategy=None, n_estimators=1000, n_jobs=None,
num_parallel_tree=None, objective='multi:softprob', ...) y_pred_train =
model.predict(X_train) y_pred_test = model.predict(X_test) target_names =
sorted(set(y))
```

```
print(f'Training accuracy: {accuracy_score(y_train,y_pred_train)}')
print(f'Training:\n {classification_report(y_train, y_pred_train,
labels=target_names)}') print(f'Testing accuracy:
{accuracy_score(y_test,y_pred_test)}')
```

```
print(f'Testing:\n {classification_report(y_test, y_pred_test, labels=target_names)}')
```

```
Training accuracy: 0.9998505901688332
Training:               precision    recall  f1-
score   support

0       1.00      1.00      1.00       673
1       1.00      1.00      1.00       679
2       1.00      1.00      1.00       647
3       1.00      1.00      1.00       659
4       1.00      1.00      1.00       678
5       1.00      1.00      1.00       701
6       1.00      1.00      1.00       662
7       1.00      1.00      1.00       672
8       1.00      1.00      1.00       660           9       1.00      1.00      1.00       662

    accuracy                           1.00      6693
macro avg       1.00      1.00      1.00      6693
weighted avg       1.00      1.00      1.00      6693

Testing accuracy: 0.89171974522293
Testing:               precision    recall  f1-
score   support

0       0.89      0.93      0.91       327
1       0.92      0.96      0.94       319
2       0.86      0.87      0.86       350
3       0.87      0.88      0.87       340
4       0.94      0.85      0.89       320
5       0.87      0.87      0.87       299
6       0.93      0.95      0.94       338
7       0.88      0.89      0.89       328
8       0.90      0.89      0.89       340           9       0.86      0.84      0.85       336

    accuracy                           0.89      3297
macro avg       0.89      0.89      0.89      3297
weighted avg       0.89      0.89      0.89      3297
```

```
#Confusion matrix
conf_mat = confusion_matrix(y_test, y_pred_test) plt.figure(figsize = (16, 9))
sns.heatmap(conf_mat,cmap="BuPu", annot=True, xticklabels = target_names, yticklabels =
target_names ) <Axes: >
```




```
#Here, we use Adam optimizer to train the model
#All of the hidden layers are using RELU activation function
#Output Layer uses softmax function


model = keras.models.Sequential([     keras.layers.Dense(512,
activation="relu", input_shape=(X_train.shape[1],)),
keras.layers.Dropout(0.2),

    keras.layers.Dense(256,activation="relu"),
keras.layers.Dropout(0.2),

    keras.layers.Dense(128,activation="relu"),
keras.layers.Dropout(0.2),
```

```
        keras.layers.Dense(64,activation="relu"),

    keras.layers.Dropout(0.2),

    keras.layers.Dense(10, activation="softmax"),


    ])
    print(model.summary())
    model_history = trainModel(model=model, epochs=1500, optimizer='adam')
```

```
Epoch 464/1500 53/53 [==============================] - 1s 12ms/step - loss: 0.0096 - accuracy: 0.9973 - val_loss: 0.5320 -
val_accuracy: 0.9211
Epoch 465/1500
   53/53 [==============================] 1s 13ms/step loss: 0 0100 accuracy: 0 9967 val loss: 0 5591     val accuracy: 0 9193
```