# Music Genre Classification

**Chilukuri Nagababu**

ncg4q@mail.umkc.edu

https://github.com/Nagu34/CSEE5590_Python_DL/tree/master/Project/Music%20Genre%20Classification

***Abstract*—What exactly is it that makes us, humans, able to tell apart two songs of different genres? The deduction cannot be trivially made from basic music features, such as BPM, or different pitches. Many music listeners create playlists based on genre, leaving potential applications such as playlist recommendation and management. Despite previous study on music genre classification with machine learning approaches, there is still room to delve into and build sophisticated models for Music Information Retrieval (MIR). A total of nine features, namely Chroma ,Tonnetz ,MFCC ,Spectral centroid ,Spectral bandwidth ,Spectral contrast ,Spectral rolloff ,RMS energy ,Zero-crossing rate were used for obtaining feature vectors for the classifiers from the GTZAN genre dataset and ultimately classify the music. By finding a way of understanding music on a deeper scientific level, this project aims to classify various music samples into genres.**

**Key words: music, genre, classification, MFCC, chroma, spectral features, GTZAN genre dataset,Neural Networks.**

## I. Introduction

Wikipedia states that “music genre is a conventional category that identifies pieces of music as belonging to a shared tradition or set of conventions.” The term “genre” is a subject to interpretation and it is often the case that genres may very fuzzy in their definition. Further, genres do not always have sound music theoretic foundations, e.g. - Indian genres are geographically defined, Baroque is classical music genre based on time period. Despite the lack of a standard criteria for defining genres, the classification of music based on genres is one of the broadest and most widely used. Genre usually assumes high weight in music recommender systems. Genre classification, till now, had been done manually by appending it to metadata of audio files or including it in album info. This project however aims at content-based classification, focusing on information within the audio rather than extraneously appended information. The traditional machine learning approach for classification is used - find suitable features of data, train classifier on feature data, make predictions. The novel thing that we have tried is the use of ensemble classifier on fundamentally different classifiers to achieve our end goal

## II. Related Work

The most influential work on genre classification using machine learning techniques was pioneered by Tzanetakis and Cook . The GTZAN dataset was created by them and is to date considered as a standard for genre classification. Scaringella et al.gives a comprehensive survey of both features and classification techniques used in the genre classification. Changsheng Xu et al.have shown how to use support vector machines for this task. Most of the work deals with supervised learning approaches. Riedmiller et al. use unsupervised learning creating a dictionary of features. gives a detailed account of evaluation of previous work on genre classification.

## III. Dataset

We have used the GTZAN dataset from the MARYSAS website. It contains 9 music genres, each genre has 100 audio clips in .au format. The genres are - blues, classical, country, disco, pop, jazz, reggae, rock, metal. Each audio clips has a length 30 seconds, are 22050Hz Mono 16-bit files. The dataset incorporates samples from variety of sources like CDs, radios, microphone recordings etc. We split the datset in 0.8 : 0.2 ratio and used confusion matrix for reporting the results.

## IV. preprocessing

The preprocessing part involved converting the audio from .au format to .wav format to make it compatible to python’s wave module for reading audio files. The open source SoX[3] utility was used for this conversion.

## V. Workflow

To classify our audio clips, we chose 9 features: Chroma ,Tonnetz ,MFCC ,Spectral centroid ,Spectral bandwidth ,Spectral contrast ,Spectral rolloff ,RMS energy ,Zero-crossing rate. we computed the features by using above features. These are all the features the librosa Python library, version 0.5.0 [25], was able to extract. Each feature set (except zero-crossing rate) is computed on windows of 2048 samples spaced by hops of 512 samples. Seven statistics were then computed over all windows: the mean, standard deviation, skew, kurtosis, median, minimum and maximum. Those 518 pre-computed features are distributed in features.csv for all tracks. Then, we used different multi-class classifiers obtain our results.

## VI. METHODOLOGY

### VI.I EXTRACTION OF FEATURES

Ten Features are extracted from raw audio file and then apply seven statistics are applied to obtain final 518 feature vector.

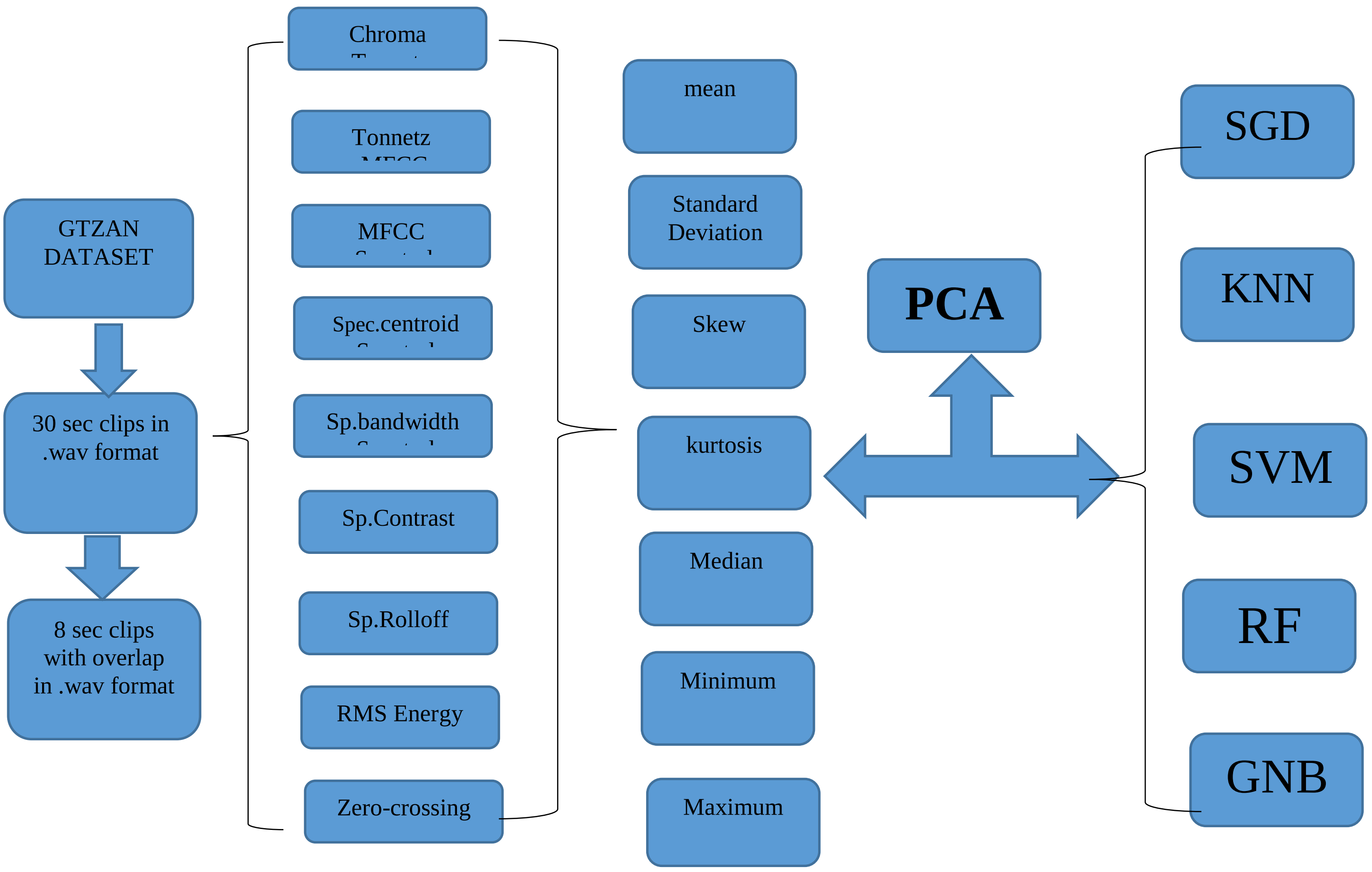


Fig. 1. Diagramtic representation of the method used

**Mel-Frequency Cepstral Coefficients**:
MFCC represents a set of short term power spectrum characteristics of the sound and have been used in the state-ofthe-art recognition and sound categorisation techniques. It models the characterics of human voice. This features is a large part of the final feature vector (13 coefficients).
The method to implement this feature is below :
– Dividing the signal into several short frames. The aim of this step is to keep an audio signal constant.
– For each frame, we calculated the periodogram estimate of the power spectrum. This is to know frequencies present in the short frames.
– Pushing the power spectra into the mel filterbank and collecting the energy in each filter to sum it. We will then know the number of energy existing in the various frequency regions.

$$M(f) = 1125\ln(1 + f/700)$$

Fig. 2. Formula to work with Mel Scale

– Calculating the logarithm of the filterbank energies in the previous It enables humans to have our features closer to what humans can hear.
– Calculating the Discrete Cosine Transform (DCT) of the result. It decorrelates the filterbank energies with each others
– Keep first 13 DCT coefficients. We remove the higher DCT coefficients which can introduce errors by representing changing in the filterbank energies
From PracticalCryptography.com tutorial

## Chroma Frequencies:

Chroma frequency vector discretizes the spectrum into chromatic keys, and represents the presence of each key. We take the histogram of present notes on a 12-note scale as a 12 length feature vector. The chroma frequency have a music theory interpretation. The histogram over the 12-note scale actually is sufficient to describe the chord played in that window. It provides a robust way to describe a similarity measure between music pieces.

## Spectral Centroid:

It describes where the ”centre of mass” for sound is. It essentially is the weighted mean of the frequencies present in the sound. Consider two songs, one from blues and one from metal. A blues song is generally consistent throughout it length while a metal song usually has more frequencies accumulated towards the end part. So spectral centroid for blues song will lie somewhere near the middle of its spectrum while that for a metal song would usually be towards its end.

$$Centroid = \frac{\sum_{n=0}^{N-1} f(n)\, x(n)}{\sum_{n=0}^{N-1} x(n)}$$

Fig. 3. Formula to calculate the Spectral Centroid

x(n) is the weighted frequency value of bin number n
f(n) is the center frequency of that bin

## Zero Crossing Rate:

It represents the number of times the waveform crosses 0. It usually has higher values for highly percussive sounds like those in metal and rock.

$$zcr = \frac{1}{T-1} \sum_{t=1}^{T-1} \mathbb{I}\{s_t s_{t-1} < 0\}$$

Fig. 4. Formula to calculate the Zero Crossing Rate

st is the signal of length t
II{X} is the indicator function (=1 if X true, else =0)

## Spectral Roll-off:

It is a measure of the shape of the signal. It represents the frequency at which high frequencies decline to 0. To obtain it, we have to calculate the fraction of bins in the power spectrum where 85% of its power is at lower frequencies.

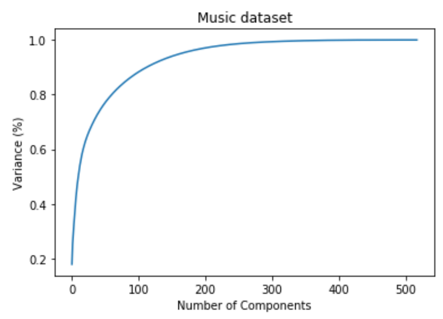
**VI.II CLASSIFICATION**

Once the feature vectors are obtained, we train different classifiers on the training set of feature vectors. Following are the different classifiers that were used –

-SGD CLASSIFIER
-K-NEAREST NEIGHBOURS
-SUPPORT VECTOR MACHINES(SVM)
-RANDOM FOREST REGRESSOR(RF)
-GAUSSSIAN NAÏVE BAYES(GNB)
-NEURAL NETWORKS

To simply the models and use as features as necessary to achieve best result, we used common approach to reduce the number of features called PRINICIPAL COMPONENT ANALYSIS(PCA). In this project we

```
<matplotlib.legend.Legend at 0x7f6cc248c048>
```



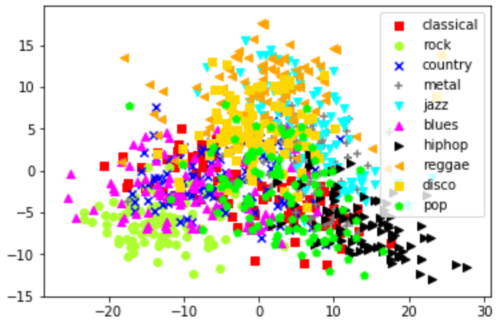


**Fig variance increase in no of components** **fig digramatic representation of all class labels**

reduced the 518 features to 150 features without any loss of data. The parameters used for various classifiers were obtained by manual tuning. It was observed that any single classifier did not classify all the genres well, but of all the classifiers SVM classifier works better. For example in the SVM with linear kernel worked well for most genres except pop and metal (See fig 5).

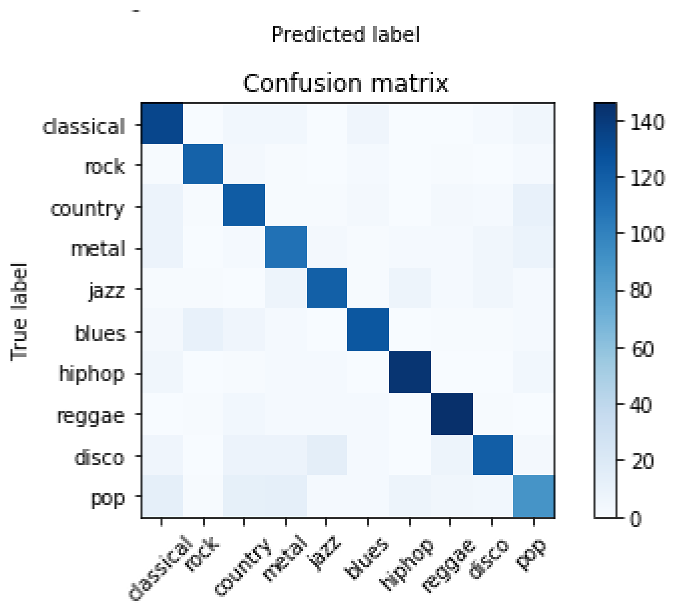


Fig. 4. Confusion Matrix for SVM classifier with linear kernel

**Neural Networks:** Neural Network (NN) based numerical method provides an alternate approach to solving classification problems. The principal advantages of the NN based numerical method are the discrete data points where field is computed, can be unstructured and therefore, the issues of meshing (uniform/non-uniform) are not a factor; the solutions are in a differentiable, closed analytic form which avoids the need to interpolate between data points where solutions are obtained using other methods. Also, recent advances in the field of Machine Learning (ML) and Artificial Intelligence (AI) has jump started the design and implementation of computer architectures that are optimized to implement training and inference tasks more efficiently.

Here the NN that will be implemented for this solution is 4- layer network with 3 hidden networks and one output layer of 10 classes.

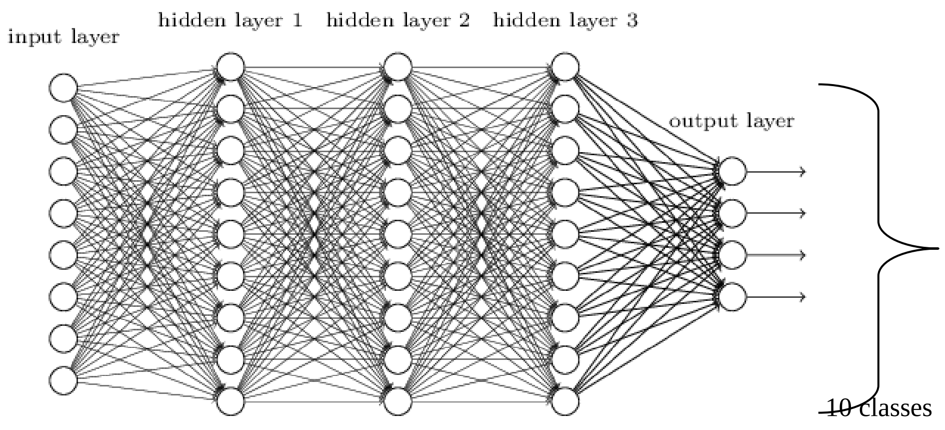


Fig. 5: Neural Network Architecture with 150 input nodes, 3 hidden nodes and 10 output node.

## VII.RESULTS

In this project we applied various algorithms to predict the results. Following are the all results obtained using 5000 samples by splitting the given dataset in 8sec clips with overlap of 2 secs. Here the comparision of two different types of implemetaions.

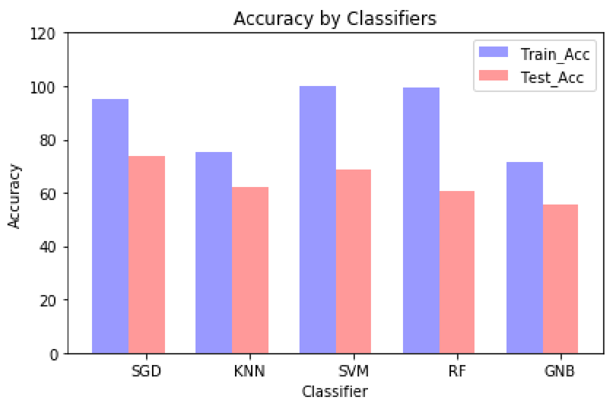


Fig 6. Accuracy obtained by different classifiers with 1000 samples

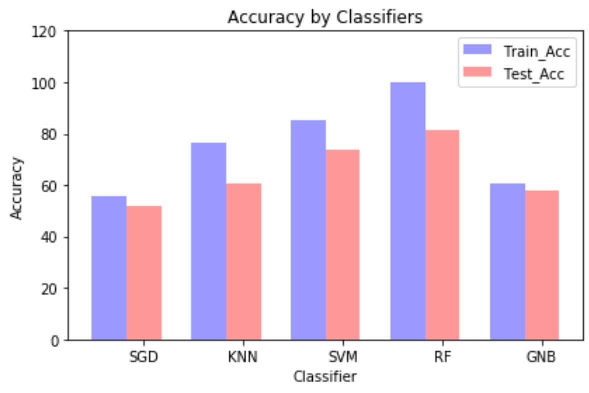


fig 7. Accuracy obtained by different classifeirs with 5000 samples

After implementations of neural networks the following are the results obtained with input 150 nodes after PCA implementation and then hidden dense layers and then output layer with 10 classes with softmax function. The obtained training accuracy is about 98% which is high and finally the validation accuracy is around **90+-3%** which is higher among all the state of art models.

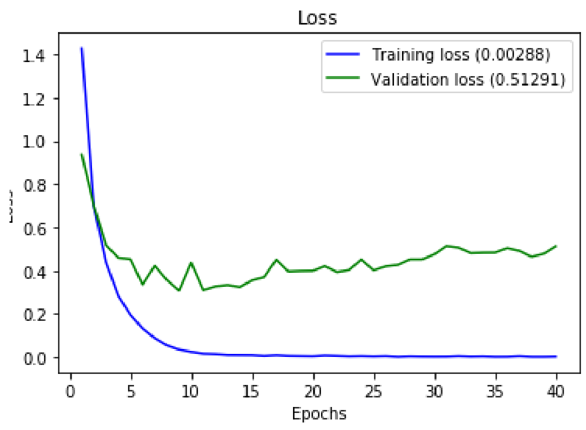


Fig 8. Both training and test loses with NN

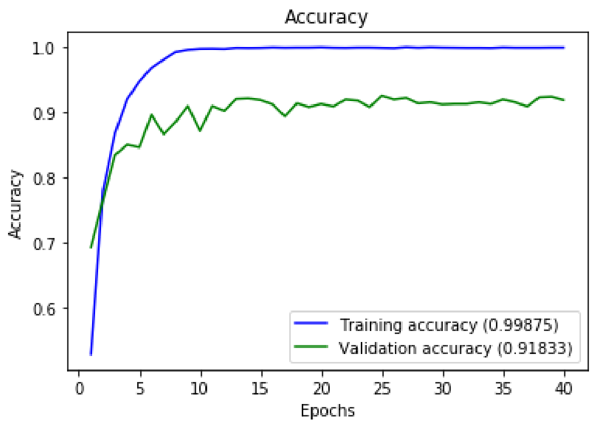
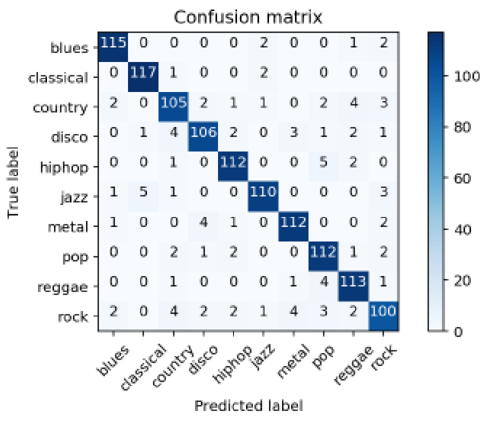


fig 9. Both training and test accuracies with NN



| | blues | classical | country | disco | hiphop | jazz | metal | pop | reggae | rock |
|---|---|---|---|---|---|---|---|---|---|---|
| blues | 115 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 1 | 2 |
| classical | 0 | 117 | 1 | 0 | 0 | 2 | 0 | 0 | 0 | 0 |
| country | 2 | 0 | 105 | 2 | 1 | 1 | 0 | 2 | 4 | 3 |
| disco | 0 | 1 | 4 | 106 | 2 | 0 | 3 | 1 | 2 | 1 |
| hiphop | 0 | 0 | 1 | 0 | 112 | 0 | 0 | 5 | 2 | 0 |
| jazz | 1 | 5 | 1 | 0 | 0 | 110 | 0 | 0 | 0 | 3 |
| metal | 1 | 0 | 0 | 4 | 1 | 0 | 112 | 0 | 0 | 2 |
| pop | 0 | 0 | 2 | 1 | 2 | 0 | 0 | 112 | 1 | 2 |
| reggae | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 4 | 113 | 1 |
| rock | 2 | 0 | 4 | 2 | 2 | 1 | 4 | 3 | 2 | 100 |

Fig 10. Confusion matrix of predicted and true label.

## VII CONCLUSION AND FUTURE WORK

Across all the models, neural networks work better after implementation of PCA and reduced the feature vector gives accuracy of about **91%**, which is higher than the rest of the models. In the future, we hope to experiment with other types of deep learning methods, given they performed the best. Given that this is time series data, some sort of RNN model may work well (GRU, LSTM, for example). We are also curious about generative aspects of this project, including some sort of genre conversion (in the same vein as generative adversarial networks which repaint photos in the style of Van Gogh, but for specifically for music). Additionally, we suspect that we may have opportunities for transfer learning, for example in classifying music by artist or by decade.

## VII REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Hareesh Bahuleyan. Music genre classification using machine learning techniques. *CoRR*, abs/1804.01149, 2018.

[3] François Chollet et al. Keras. https://keras.io, 2015.

[4] Mingwen Dong. Convolutional neural network achieves human-level accuracy in music genre classification. *CoRR*, abs/1802.09697, 2018.

[5] Brian McFee, Matt McVicar, Stefan Balke, Carl Thomé, Vincent Lostanlen, Colin Raffel, Dana Lee, Oriol Nieto, Eric Battenberg, Dan Ellis, Ryuichi Yamamoto, Josh Moore, WZY, Rachel Bittner, Keunwoo Choi, Pius Friesch, Fabian-Robert Stöter, Matt Vollrath, Siddhartha Kumar, nehz, Simon Waloschek, Seth, Rimvydas Naktinis, Douglas Repetto, Curtis "Fjord" Hawthorne, CJ Carr, João Felipe Santos, JackieWu, Erik, and Adrian Holovaty. librosa/librosa: 0.6.2, August 2018.

[6] Y. Panagakis, C. Kotropoulos, and G. R. Arce. Music genre classification via sparse representations of auditory temporal modulations. In *2009 17th European Signal Processing Conference*, pages 1–5, Aug 2009.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[8] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, July 2002.

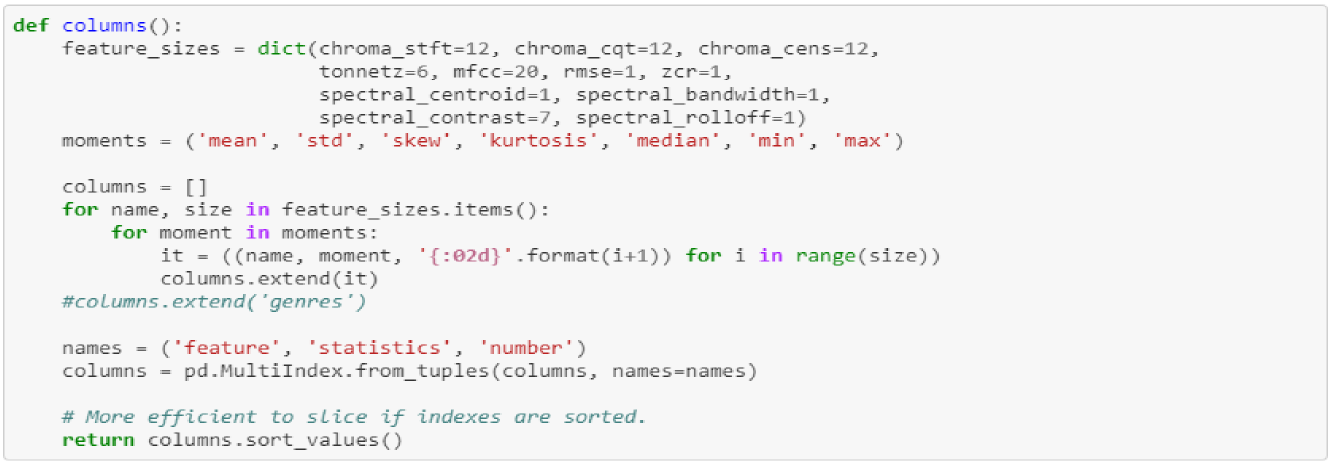
[9] Changsheng Xu, N. C. Maddage, Xi Shao, Fang Cao, and Qi Tian. Musical genre classification using support vector machines. In *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03).*, volume 5, pages V–429, April 2003.

[10] https://github.com/mdeff/fma

[11] https://github.com/derekahuang/Music-Classification

## Code Snippets of various implementations:

**Seven statistics implementation of raw audio samples obtained from 9 features.**

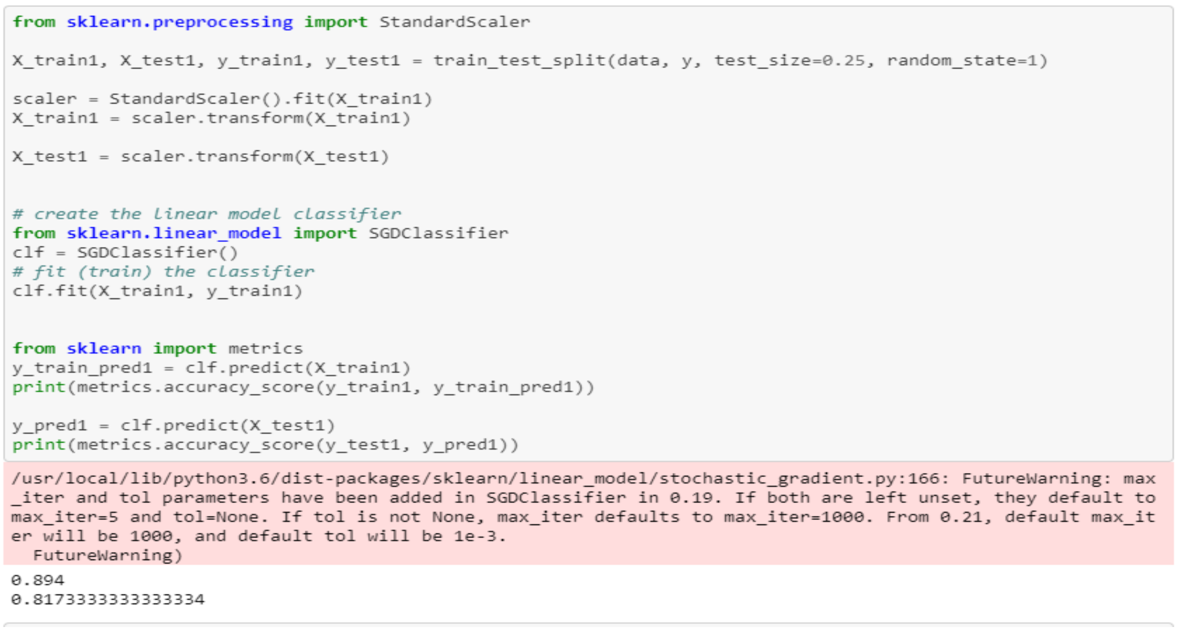


```python
def columns():
    feature_sizes = dict(chroma_stft=12, chroma_cqt=12, chroma_cens=12,
                         tonnetz=6, mfcc=20, rmse=1, zcr=1,
                         spectral_centroid=1, spectral_bandwidth=1,
                         spectral_contrast=7, spectral_rolloff=1)
    moments = ('mean', 'std', 'skew', 'kurtosis', 'median', 'min', 'max')

    columns = []
    for name, size in feature_sizes.items():
        for moment in moments:
            it = ((name, moment, '{:02d}'.format(i+1)) for i in range(size))
            columns.extend(it)
    #columns.extend('genres')

    names = ('feature', 'statistics', 'number')
    columns = pd.MultiIndex.from_tuples(columns, names=names)

    # More efficient to slice if indexes are sorted.
    return columns.sort_values()
```

**Total 9 features extracted from raw audio files with various dimensional feature vectors:**

```
x, sr = librosa.load(filename, sr=None, mono=True)  # kaiser_fast

f = librosa.feature.zero_crossing_rate(x, frame_length=2048, hop_length=512)
feature_stats('zcr', f)

cqt = np.abs(librosa.cqt(x, sr=sr, hop_length=512, bins_per_octave=12,
                         n_bins=7*12, tuning=None))
assert cqt.shape[0] == 7 * 12
assert np.ceil(len(x)/512) <= cqt.shape[1] <= np.ceil(len(x)/512)+1

f = librosa.feature.chroma_cqt(C=cqt, n_chroma=12, n_octaves=7)
feature_stats('chroma_cqt', f)
f = librosa.feature.chroma_cens(C=cqt, n_chroma=12, n_octaves=7)
feature_stats('chroma_cens', f)
f = librosa.feature.tonnetz(chroma=f)
feature_stats('tonnetz', f)

del cqt
stft = np.abs(librosa.stft(x, n_fft=2048, hop_length=512))
assert stft.shape[0] == 1 + 2048 // 2
assert np.ceil(len(x)/512) <= stft.shape[1] <= np.ceil(len(x)/512)+1
del x

f = librosa.feature.chroma_stft(S=stft**2, n_chroma=12)
feature_stats('chroma_stft', f)

f = librosa.feature.rmse(S=stft)
feature_stats('rmse', f)

f = librosa.feature.spectral_centroid(S=stft)
feature_stats('spectral_centroid', f)
f = librosa.feature.spectral_bandwidth(S=stft)
feature_stats('spectral_bandwidth', f)
f = librosa.feature.spectral_contrast(S=stft, n_bands=6)
feature_stats('spectral_contrast', f)
f = librosa.feature.spectral_rolloff(S=stft)
feature_stats('spectral_rolloff', f)

mel = librosa.feature.melspectrogram(sr=sr, S=stft**2)
del stft
f = librosa.feature.mfcc(S=librosa.power_to_db(mel), n_mfcc=20)
feature_stats('mfcc', f)
```

## 3.Single classifier model implemetation:



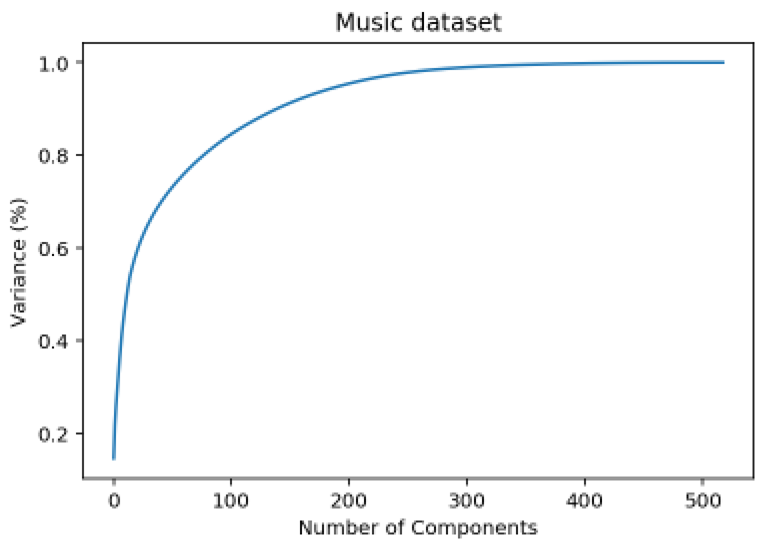
```
from sklearn.preprocessing import StandardScaler

X_train1, X_test1, y_train1, y_test1 = train_test_split(data, y, test_size=0.25, random_state=1)

scaler = StandardScaler().fit(X_train1)
X_train1 = scaler.transform(X_train1)

X_test1 = scaler.transform(X_test1)


# create the linear model classifier
from sklearn.linear_model import SGDClassifier
clf = SGDClassifier()
# fit (train) the classifier
clf.fit(X_train1, y_train1)


from sklearn import metrics
y_train_pred1 = clf.predict(X_train1)
print(metrics.accuracy_score(y_train1, y_train_pred1))

y_pred1 = clf.predict(X_test1)
print(metrics.accuracy_score(y_test1, y_pred1))
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/linear_model/stochastic_gradient.py:166: FutureWarning: max
_iter and tol parameters have been added in SGDClassifier in 0.19. If both are left unset, they default to
max_iter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=1000. From 0.21, default max_it
er will be 1000, and default tol will be 1e-3.
  FutureWarning)
0.894
0.8173333333333334
```

## 4.PCA implemetation:

```
from sklearn.preprocessing import StandardScaler

data_rescaled = StandardScaler().fit_transform(data)
from sklearn.decomposition import PCA
```

```
#Fitting the PCA algorithm with our Data
pca = PCA().fit(data_rescaled)
#Plotting the Cumulative Summation of the Explained Variance
plt.figure()
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('Number of Components')
plt.ylabel('Variance (%)') #for each component
plt.title('Music dataset')
plt.show()
```



## 5. Neural Networks Model Implementation:

```
X_train, X_test, y_train, y_test = train_test_split(dataset, y_binary, test_size=0.2, random_state=1,stra
ify=y_binary)
```

```
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
(4800, 150) (1200, 150) (4800, 10) (1200, 10)
```

```
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.2, random_state=1)
```

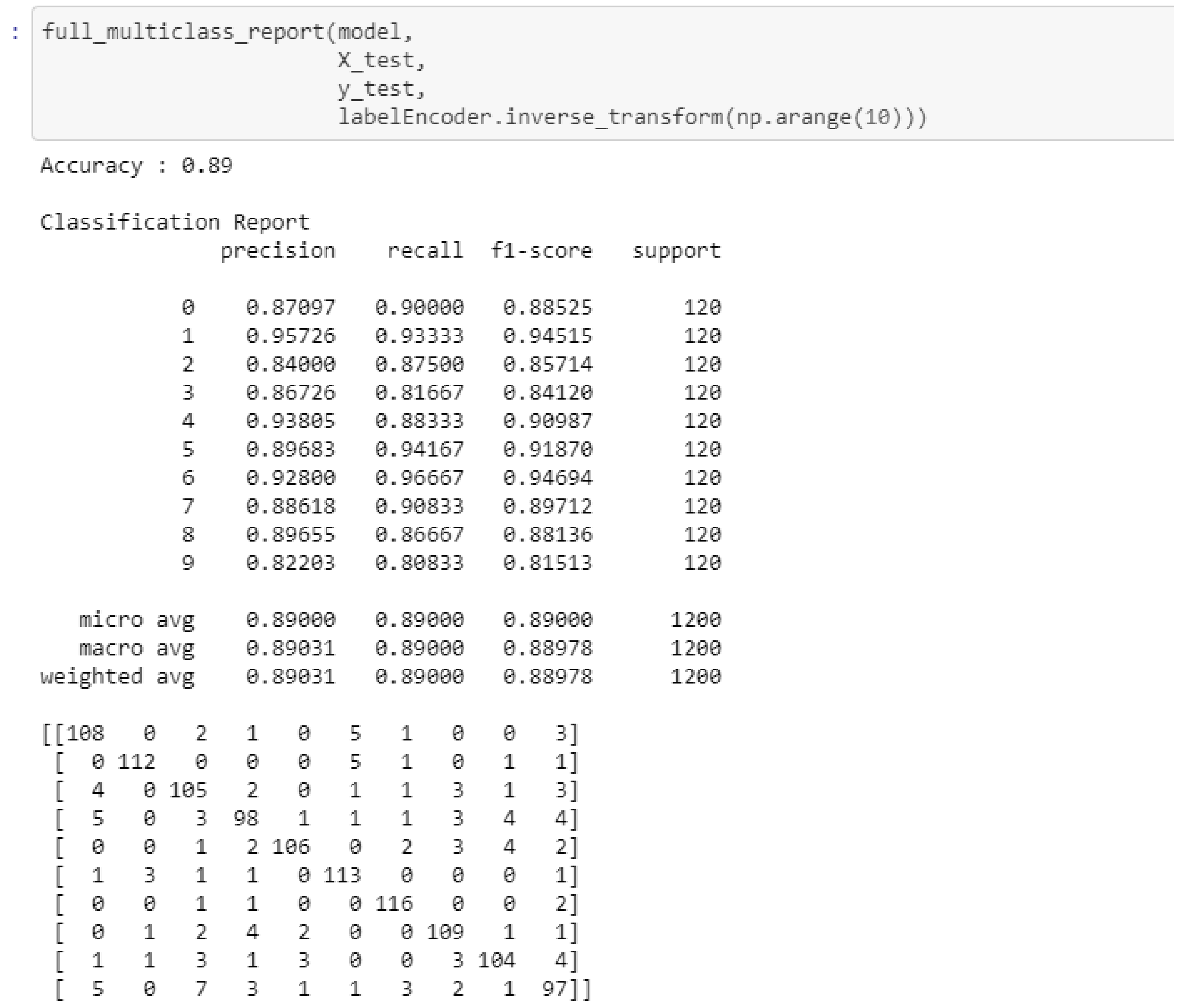
```
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
(3840, 150) (1200, 150) (3840, 10) (1200, 10)
```

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
```

```
def build_model():
    model = keras.Sequential([
        layers.Dense(128, activation=tf.nn.relu, input_shape=[150]),
        layers.Dense(128, activation=tf.nn.relu),
        layers.Dense(64, activation=tf.nn.relu),
        layers.Dense(10,activation='softmax')])

    optimizer = tf.keras.optimizers.RMSprop(0.001)

    model.compile(loss='categorical_crossentropy',
                  optimizer=optimizer,
                  metrics=['accuracy'])
    return model
```

6. Neural Networks Model Results:



```
: full_multiclass_report(model,
                         X_test,
                         y_test,
                         labelEncoder.inverse_transform(np.arange(10)))
```

```
Accuracy : 0.89

Classification Report
              precision    recall  f1-score   support

           0    0.87097   0.90000   0.88525       120
           1    0.95726   0.93333   0.94515       120
           2    0.84000   0.87500   0.85714       120
           3    0.86726   0.81667   0.84120       120
           4    0.93805   0.88333   0.90987       120
           5    0.89683   0.94167   0.91870       120
           6    0.92800   0.96667   0.94694       120
           7    0.88618   0.90833   0.89712       120
           8    0.89655   0.86667   0.88136       120
           9    0.82203   0.80833   0.81513       120

   micro avg    0.89000   0.89000   0.89000      1200
   macro avg    0.89031   0.89000   0.88978      1200
weighted avg    0.89031   0.89000   0.88978      1200

[[108   0   2   1   0   5   1   0   0   3]
 [  0 112   0   0   0   5   1   0   1   1]
 [  4   0 105   2   0   1   1   3   1   3]
 [  5   0   3  98   1   1   1   3   4   4]
 [  0   0   1   2 106   0   2   3   4   2]
 [  1   3   1   1   0 113   0   0   0   1]
 [  0   0   1   1   0   0 116   0   0   2]
 [  0   1   2   4   2   0   0 109   1   1]
 [  1   1   3   1   3   0   0   3 104   4]
 [  5   0   7   3   1   1   3   2   1  97]]
```