Capstone Project Report

Name: Sumit Patel

Course: AI and ML (Batch – AUG 2020)

Duration: 10 months

Music Genre Classification

Problem Statement:

Classify the GTZAN music dataset using multiple classifiers and compare their accuracy.

Prerequisites

What things you need to install the software and how to install them:

Python 3.6 This setup requires that your machine has latest version of python. The following url https://www.python.org/downloads/ can be referred to download python. Once you have python downloaded and installed, you will need to setup PATH variables (if you want to run python program directly, detail instructions are below in how to run software section). To do that check this: https://www.pythoncentral.io/add-python-to-path-python-is-not-recognized-as-an-internal-or-externalcommand/. Setting up PATH variable is optional as you can also run program without it and more instruction are given below on this topic. Second and easier option is to download anaconda and use its anaconda prompt to run the commands.

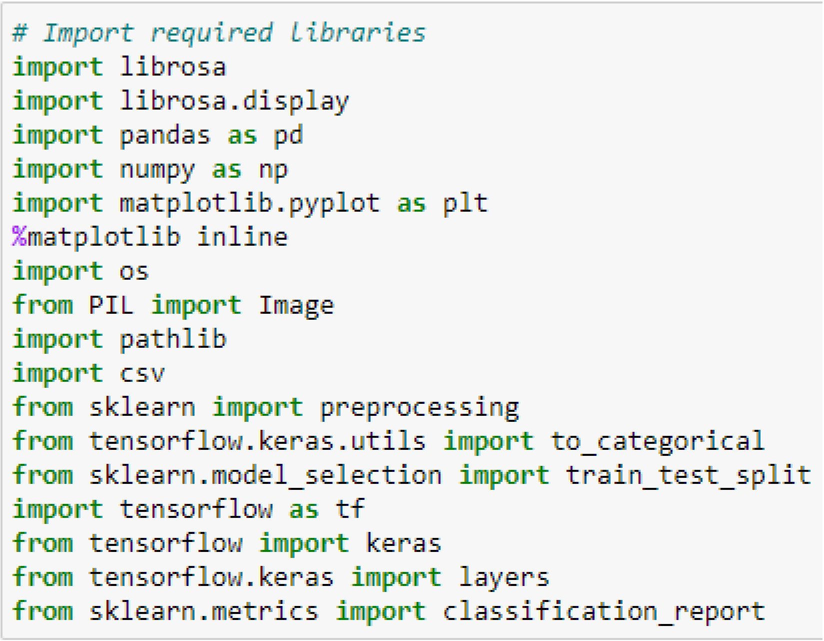
To install anaconda check this url https://www.anaconda.com/download/ You will also need to download and install below 3 packages after you install either python or anaconda from the steps above Sklearn (scikit-learn) numpy scipy if you have chosen to install python 3.6 then run below commands in command prompt/terminal to install these packages:

```
pip install numpy

pip install pandas

pip install sklearn

pip install tensorflow

pip install librosa
```

If you have chosen to install anaconda then run below commands in anaconda prompt to install these packages:

```
conda install -c anaconda numpy

conda install -c anaconda pandas

conda install -c anaconda sklearn

conda install -c anaconda tensorflow

conda install -c anaconda librosa
```

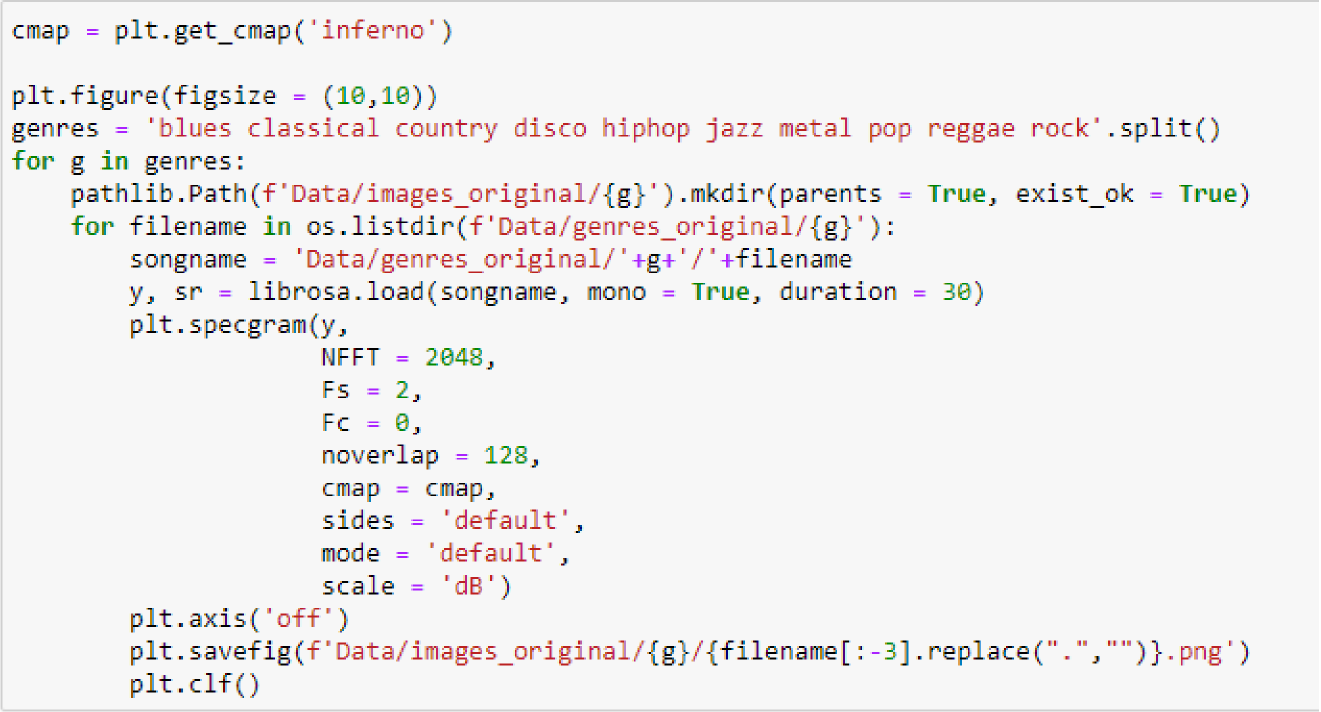
Importing the libraries and loading dataset.



```
# Import required libraries
import librosa
import librosa.display
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import os
from PIL import Image
import pathlib
import csv
from sklearn import preprocessing
from tensorflow.keras.utils import to_categorical
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from sklearn.metrics import classification_report
```

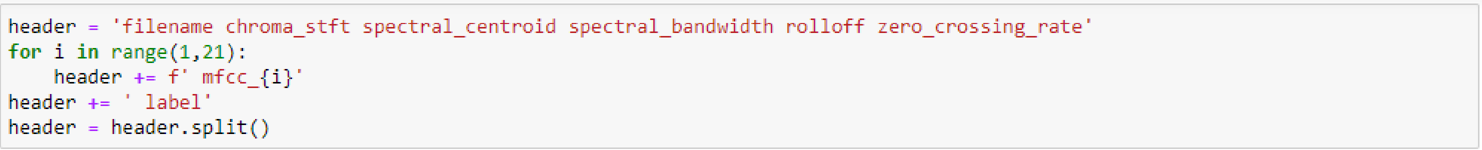
Generating the dataset

Images

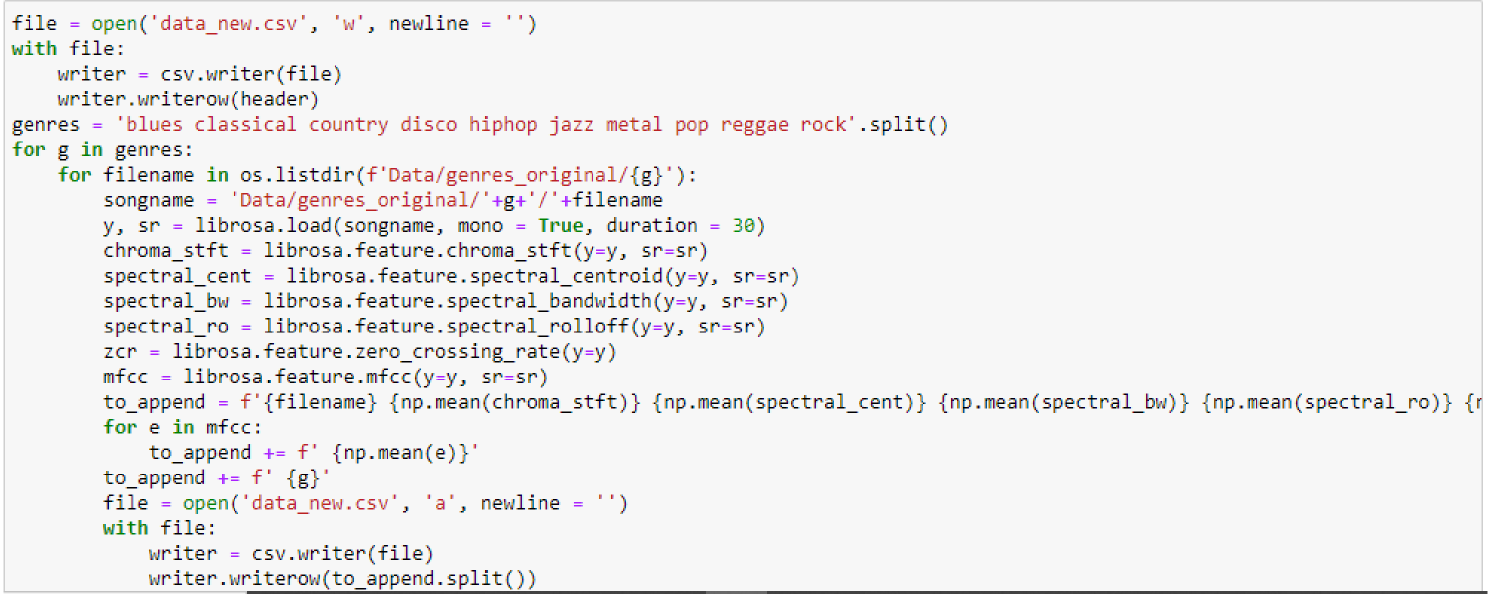


```
cmap = plt.get_cmap('inferno')

plt.figure(figsize = (10,10))
genres = 'blues classical country disco hiphop jazz metal pop reggae rock'.split()
for g in genres:
    pathlib.Path(f'Data/images_original/{g}').mkdir(parents = True, exist_ok = True)
    for filename in os.listdir(f'Data/genres_original/{g}'):
        songname = 'Data/genres_original/'+g+'/'+filename
        y, sr = librosa.load(songname, mono = True, duration = 30)
        plt.specgram(y,
                     NFFT = 2048,
                     Fs = 2,
                     Fc = 0,
                     noverlap = 128,
                     cmap = cmap,
                     sides = 'default',
                     mode = 'default',
                     scale = 'dB')
        plt.axis('off')
        plt.savefig(f'Data/images_original/{g}/{filename[:-3].replace(".","")}.png')
        plt.clf()
```

Audio features in csv

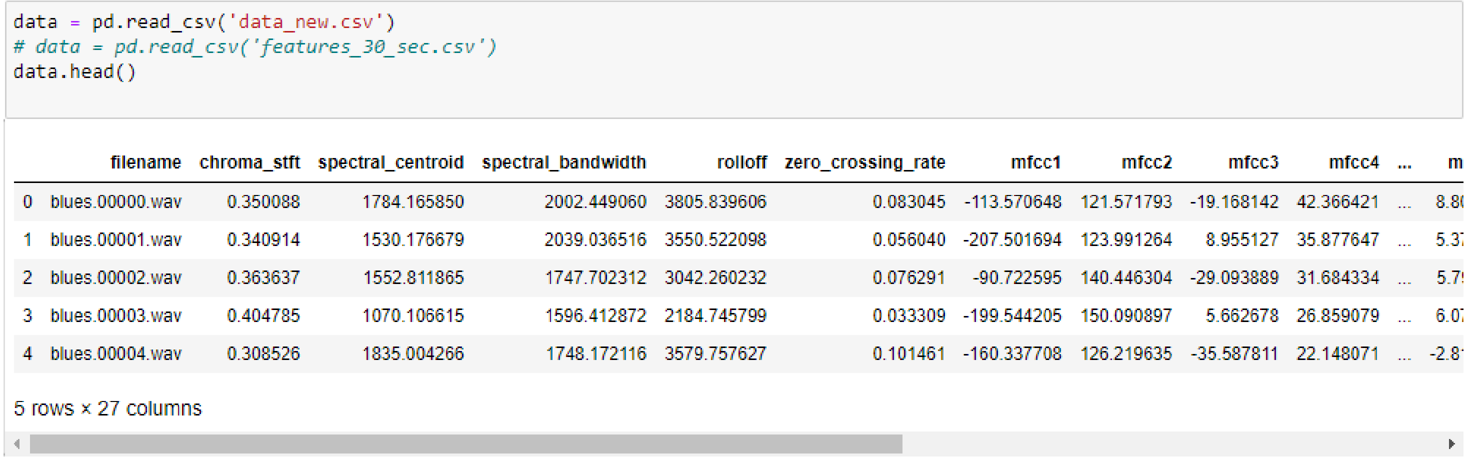


```
header = 'filename chroma_stft spectral_centroid spectral_bandwidth rolloff zero_crossing_rate'
for i in range(1,21):
    header += f' mfcc_{i}'
header += ' label'
header = header.split()
```



```
file = open('data_new.csv', 'w', newline = '')
with file:
    writer = csv.writer(file)
    writer.writerow(header)
genres = 'blues classical country disco hiphop jazz metal pop reggae rock'.split()
for g in genres:
    for filename in os.listdir(f'Data/genres_original/{g}'):
        songname = 'Data/genres_original/'+g+'/'+filename
        y, sr = librosa.load(songname, mono = True, duration = 30)
        chroma_stft = librosa.feature.chroma_stft(y=y, sr=sr)
        spectral_cent = librosa.feature.spectral_centroid(y=y, sr=sr)
        spectral_bw = librosa.feature.spectral_bandwidth(y=y, sr=sr)
        spectral_ro = librosa.feature.spectral_rolloff(y=y, sr=sr)
        zcr = librosa.feature.zero_crossing_rate(y=y)
        mfcc = librosa.feature.mfcc(y=y, sr=sr)
        to_append = f'{filename} {np.mean(chroma_stft)} {np.mean(spectral_cent)} {np.mean(spectral_bw)} {np.mean(spectral_ro)} {r
        for e in mfcc:
            to_append += f' {np.mean(e)}'
        to_append += f' {g}'
        file = open('data_new.csv', 'a', newline = '')
        with file:
            writer = csv.writer(file)
            writer.writerow(to_append.split())
```

Reading the data

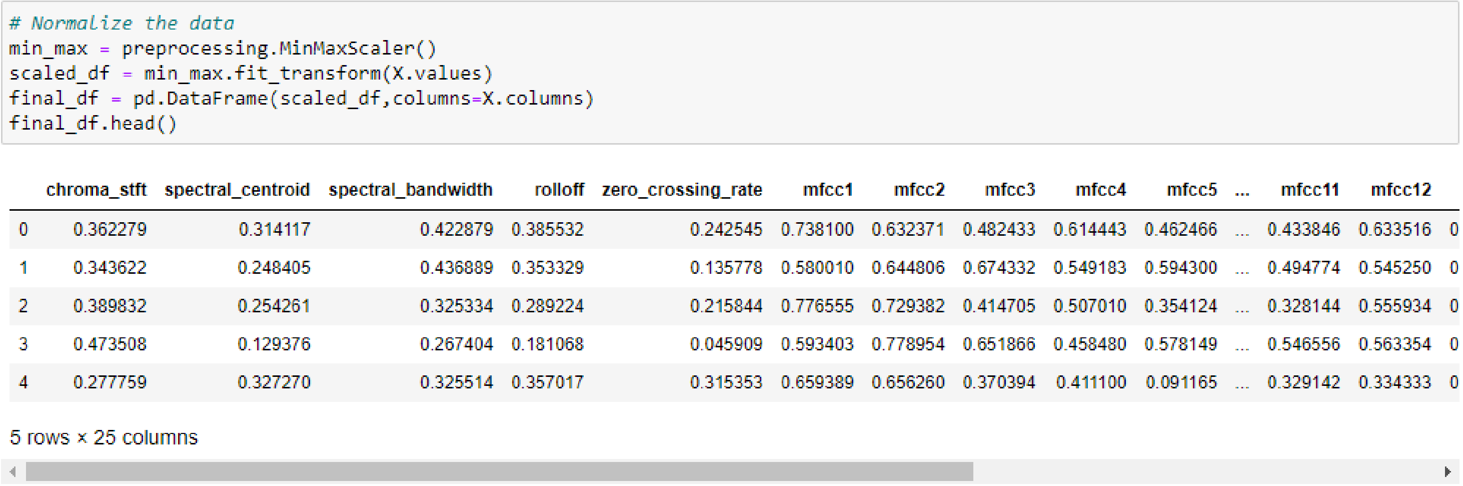


```
data = pd.read_csv('data_new.csv')
# data = pd.read_csv('features_30_sec.csv')
data.head()
```

	filename	chroma_stft	spectral_centroid	spectral_bandwidth	rolloff	zero_crossing_rate	mfcc1	mfcc2	mfcc3	mfcc4	...	m
0	blues.00000.wav	0.350088	1784.165850	2002.449060	3805.839606	0.083045	-113.570648	121.571793	-19.168142	42.366421	...	8.80
1	blues.00001.wav	0.340914	1530.176679	2039.036516	3550.522098	0.056040	-207.501694	123.991264	8.955127	35.877647	...	5.3
2	blues.00002.wav	0.363637	1552.811865	1747.702312	3042.260232	0.076291	-90.722595	140.446304	-29.093889	31.684334	...	5.7
3	blues.00003.wav	0.404785	1070.106615	1596.412872	2184.745799	0.033309	-199.544205	150.090897	5.662678	26.859079	...	6.0
4	blues.00004.wav	0.308526	1835.004266	1748.172116	3579.757627	0.101461	-160.337708	126.219635	-35.587811	22.148071	...	-2.8

5 rows × 27 columns

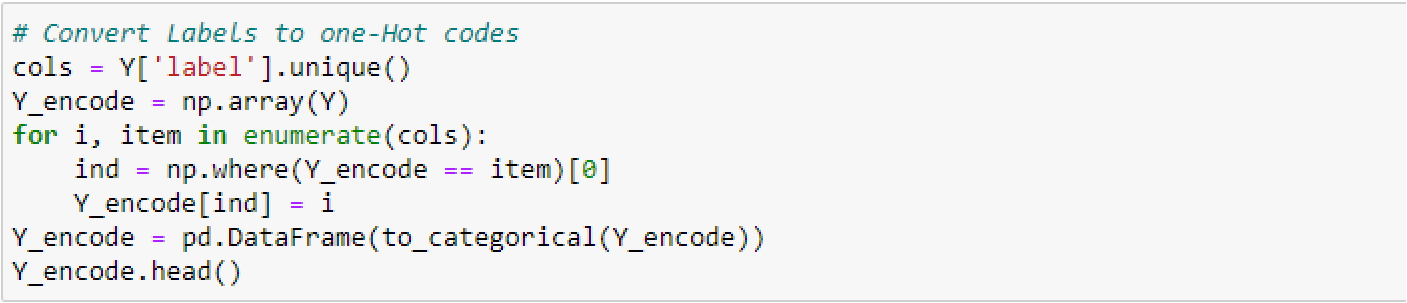
Data pre-processing



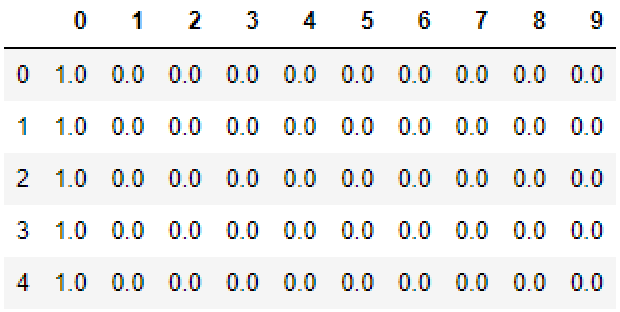
```
# Normalize the data
min_max = preprocessing.MinMaxScaler()
scaled_df = min_max.fit_transform(X.values)
final_df = pd.DataFrame(scaled_df,columns=X.columns)
final_df.head()
```

	chroma_stft	spectral_centroid	spectral_bandwidth	rolloff	zero_crossing_rate	mfcc1	mfcc2	mfcc3	mfcc4	mfcc5	...	mfcc11	mfcc12	
0	0.362279	0.314117	0.422879	0.385532	0.242545	0.738100	0.632371	0.482433	0.614443	0.462466	...	0.433846	0.633516	0
1	0.343622	0.248405	0.436889	0.353329	0.135778	0.580010	0.644806	0.674332	0.549183	0.594300	...	0.494774	0.545250	0
2	0.389832	0.254261	0.325334	0.289224	0.215844	0.776555	0.729382	0.414705	0.507010	0.354124	...	0.328144	0.555934	0
3	0.473508	0.129376	0.267404	0.181068	0.045909	0.593403	0.778954	0.651866	0.458480	0.578149	...	0.546556	0.563354	0
4	0.277759	0.327270	0.325514	0.357017	0.315353	0.659389	0.656260	0.370394	0.411100	0.091165	...	0.329142	0.334333	0

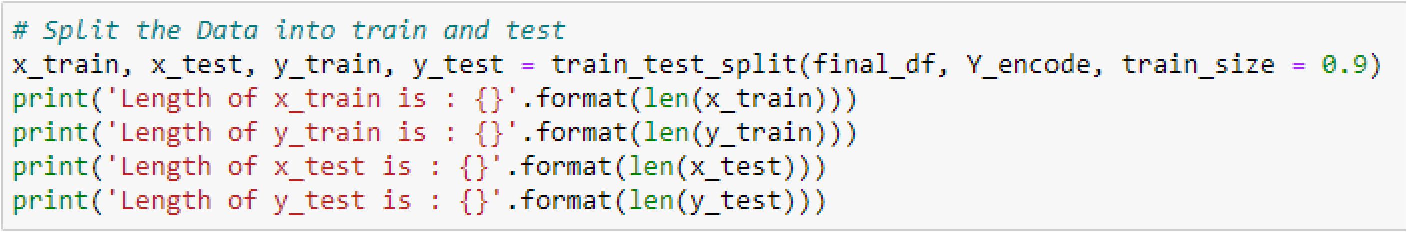
5 rows × 25 columns



```
# Convert Labels to one-Hot codes
cols = Y['label'].unique()
Y_encode = np.array(Y)
for i, item in enumerate(cols):
    ind = np.where(Y_encode == item)[0]
    Y_encode[ind] = i
Y_encode = pd.DataFrame(to_categorical(Y_encode))
Y_encode.head()
```

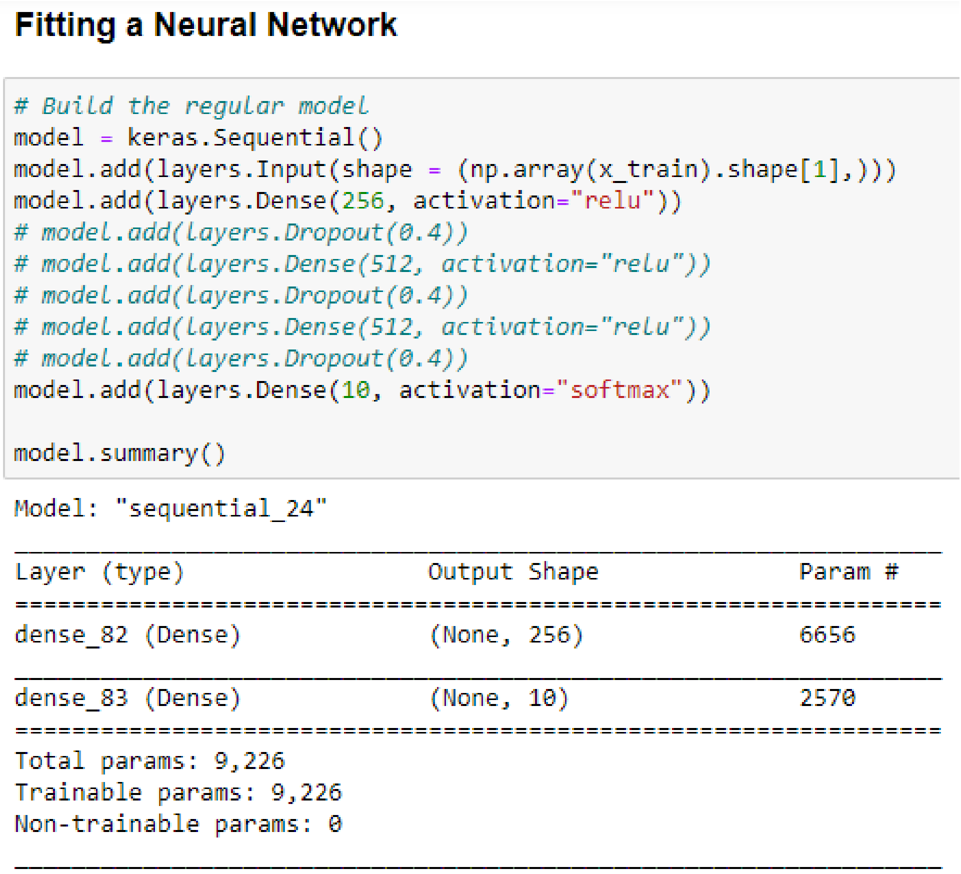


	0	1	2	3	4	5	6	7	8	9
0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0



```
# Split the Data into train and test
x_train, x_test, y_train, y_test = train_test_split(final_df, Y_encode, train_size = 0.9)
print('Length of x_train is : {}'.format(len(x_train)))
print('Length of y_train is : {}'.format(len(y_train)))
print('Length of x_test is : {}'.format(len(x_test)))
print('Length of y_test is : {}'.format(len(y_test)))
```

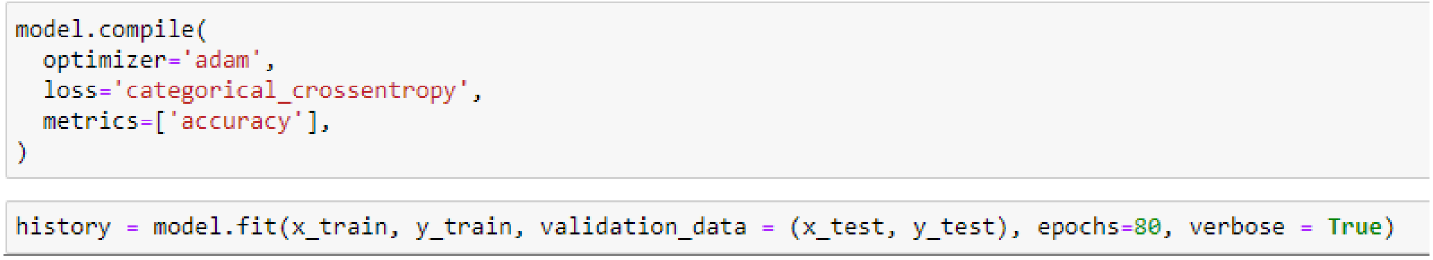
Training and evaluating the model



Fitting a Neural Network

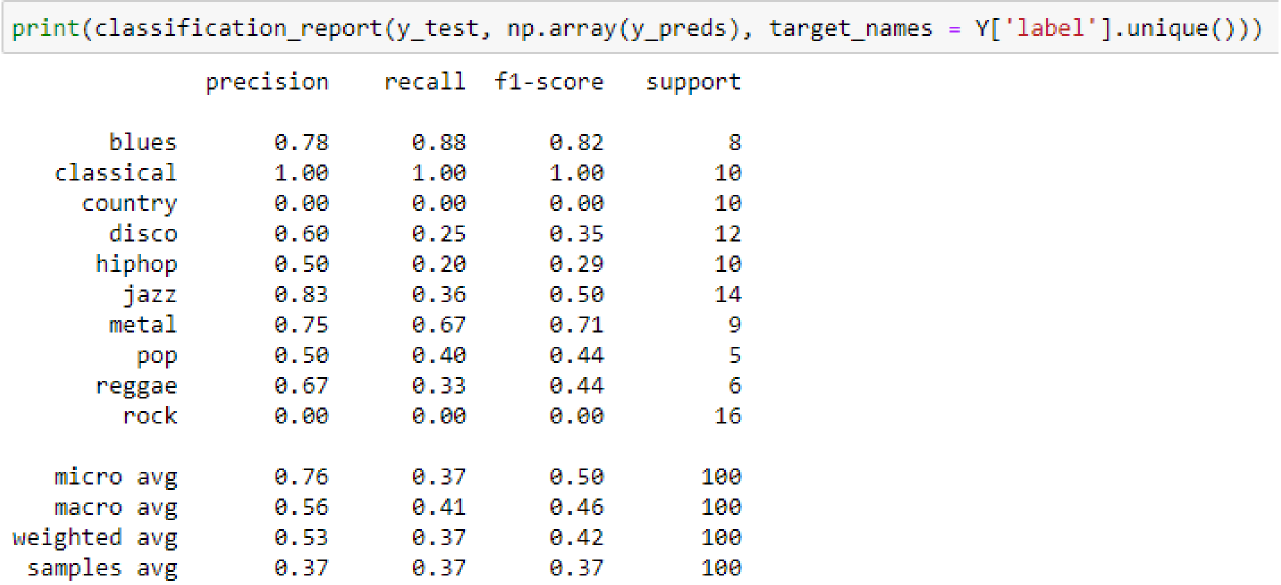
```
# Build the regular model
model = keras.Sequential()
model.add(layers.Input(shape = (np.array(x_train).shape[1],)))
model.add(layers.Dense(256, activation="relu"))
# model.add(layers.Dropout(0.4))
# model.add(layers.Dense(512, activation="relu"))
# model.add(layers.Dropout(0.4))
# model.add(layers.Dense(512, activation="relu"))
# model.add(layers.Dropout(0.4))
model.add(layers.Dense(10, activation="softmax"))

model.summary()
```

```
Model: "sequential_24"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_82 (Dense)             (None, 256)               6656
_________________________________________________________________
dense_83 (Dense)             (None, 10)                2570
=================================================================
Total params: 9,226
Trainable params: 9,226
Non-trainable params: 0
_________________________________________________________________
```



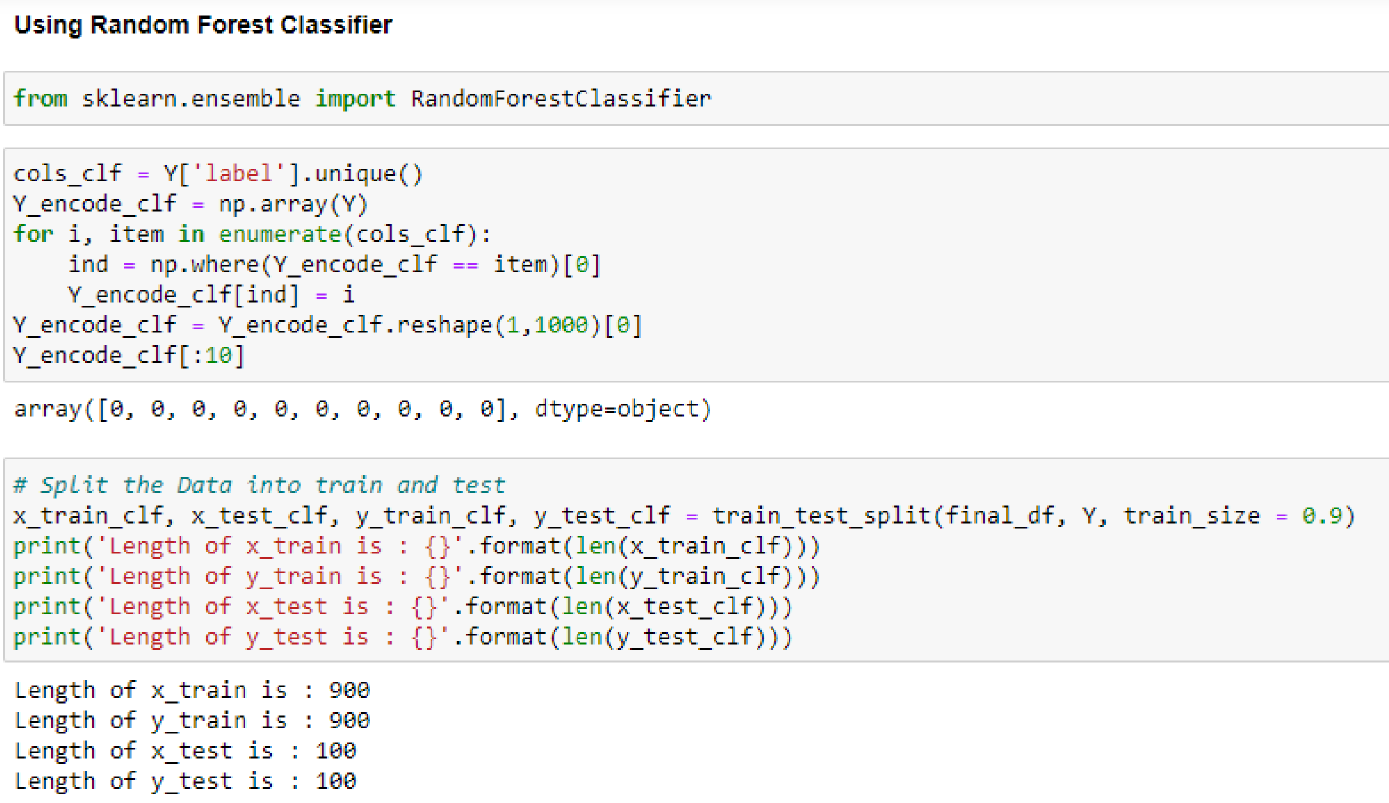
```
model.compile(
  optimizer='adam',
  loss='categorical_crossentropy',
  metrics=['accuracy'],
)
```

```
history = model.fit(x_train, y_train, validation_data = (x_test, y_test), epochs=80, verbose = True)
```



```
print(classification_report(y_test, np.array(y_preds), target_names = Y['label'].unique()))
```

```
              precision    recall  f1-score   support

       blues       0.78      0.88      0.82         8
   classical       1.00      1.00      1.00        10
     country       0.00      0.00      0.00        10
       disco       0.60      0.25      0.35        12
      hiphop       0.50      0.20      0.29        10
        jazz       0.83      0.36      0.50        14
       metal       0.75      0.67      0.71         9
         pop       0.50      0.40      0.44         5
      reggae       0.67      0.33      0.44         6
        rock       0.00      0.00      0.00        16

   micro avg       0.76      0.37      0.50       100
   macro avg       0.56      0.41      0.46       100
weighted avg       0.53      0.37      0.42       100
 samples avg       0.37      0.37      0.37       100
```



Using Random Forest Classifier

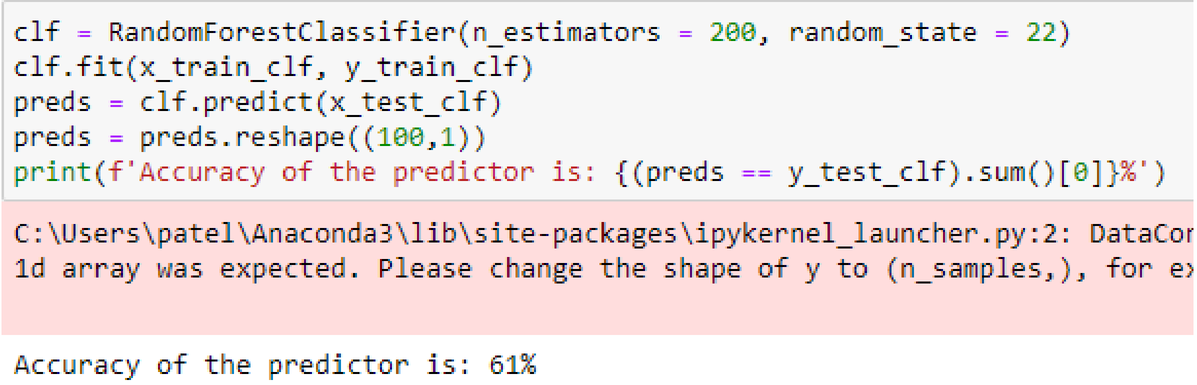
```
from sklearn.ensemble import RandomForestClassifier
```

```
cols_clf = Y['label'].unique()
Y_encode_clf = np.array(Y)
for i, item in enumerate(cols_clf):
    ind = np.where(Y_encode_clf == item)[0]
    Y_encode_clf[ind] = i
Y_encode_clf = Y_encode_clf.reshape(1,1000)[0]
Y_encode_clf[:10]
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=object)
```

```
# Split the Data into train and test
x_train_clf, x_test_clf, y_train_clf, y_test_clf = train_test_split(final_df, Y, train_size = 0.9)
print('Length of x_train is : {}'.format(len(x_train_clf)))
print('Length of y_train is : {}'.format(len(y_train_clf)))
print('Length of x_test is : {}'.format(len(x_test_clf)))
print('Length of y_test is : {}'.format(len(y_test_clf)))
```

```
Length of x_train is : 900
Length of y_train is : 900
Length of x_test is : 100
Length of y_test is : 100
```



```
clf = RandomForestClassifier(n_estimators = 200, random_state = 22)
clf.fit(x_train_clf, y_train_clf)
preds = clf.predict(x_test_clf)
preds = preds.reshape((100,1))
print(f'Accuracy of the predictor is: {(preds == y_test_clf).sum()[0]}%')
```

```
C:\Users\patel\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: DataCor
1d array was expected. Please change the shape of y to (n_samples,), for ex
```

```
Accuracy of the predictor is: 61%
```