Capstone Project

Music Genre Classification

Name : Dhivakar.R

Course: AI and ML(Aug2020)

Problem Statement: Sound/Audio signals can be represented in the form of various parameters such as frequency, bandwidth, roll-off and so on. Using various python libraries, we can perform feature extraction for these audio signals. These features can then be processed and further used to perform classification. In this project, we will use GTZAN dataset https://www.kaggle.com/andradaolteanu/gtzan-dataset-music-genreclassification

which consists of 10 genre with 100 songs each, all having a length of 30 seconds. Given this dataset, our task is to: Take two songs from each of the genre and visualize them and also find their spectrogram.

a) Create a dataset by extracting feature for each of the songs in GTZAN dataset. For our task, we will specifically use the following features: Mel-Frequency Cepstral Coefficients, Spectral Centroid, Zero Crossing Rate, Chroma Frequencies and Spectral Roll-off
b) Given total 1000 examples, perform K-Means-Clustering on the dataset to cross verify that the optimal number of clusters are 10 (one for each genre).
c) Divide the dataset into two parts: 90% train and 10% test i.e. for each genre use 90% of the dataset as train and the remaining as test dataset.
d) Perform classification using any of the four classification algorithms and compare the accuracy obtained. Study the architecture of the model used and describe the reason for the model with best accuracy.

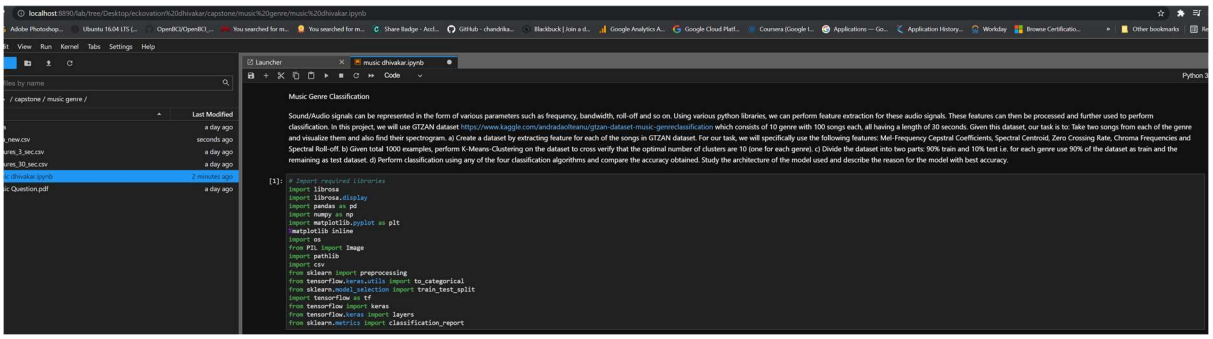
Prerequisites

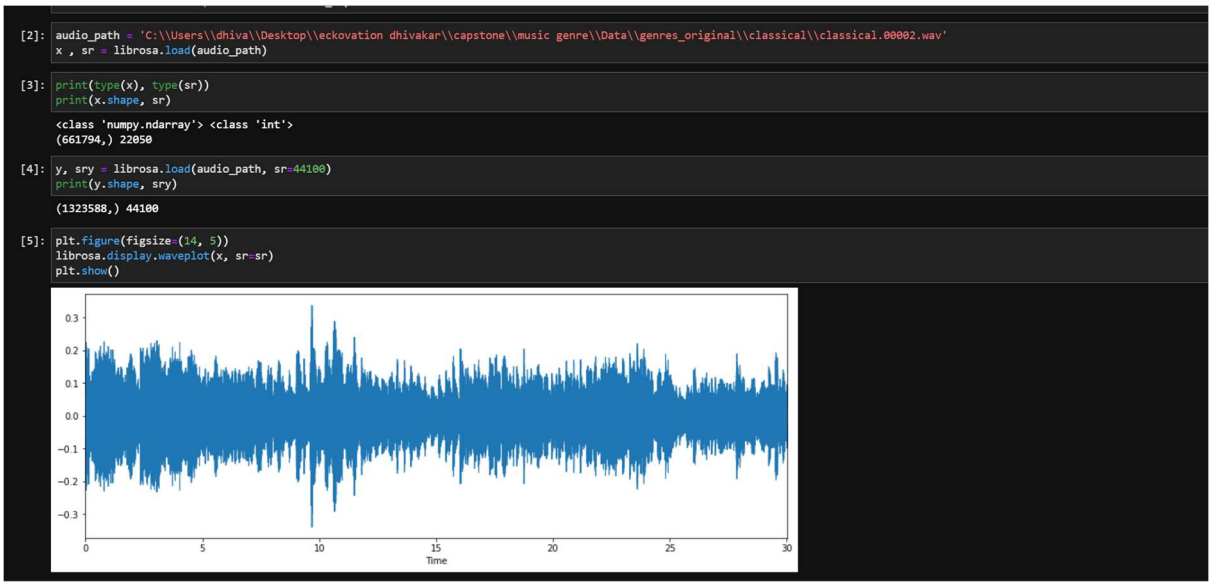
What things you need to install the software and how to install them:

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 -U scikit-learn pip install numpy pip install scipy if you have chosen to install anaconda then run below commands in anaconda prompt to install these packages conda install -c scikit-learn conda install -c anaconda numpy conda install -c anaconda scipy

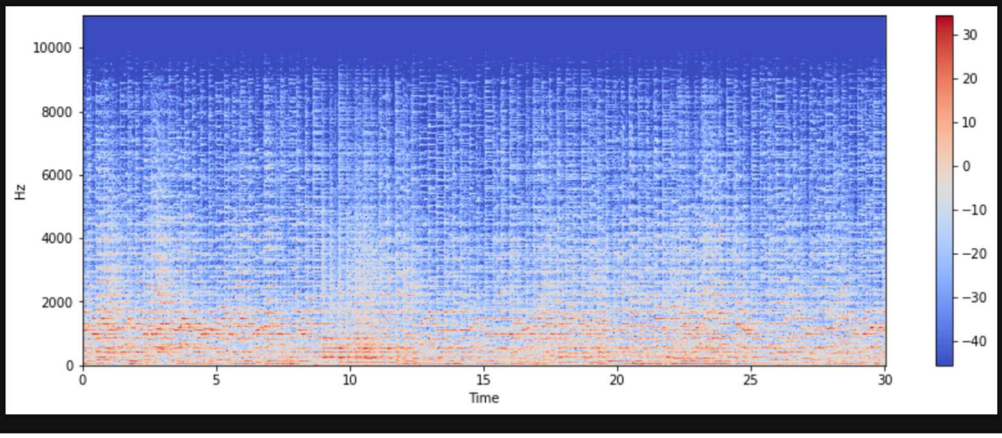
Importing the libraries and loading dataset :





```
[6]: X = librosa.stft(x)
     Xdb = librosa.amplitude_to_db(abs(X))
     plt.figure(figsize=(14, 5))
     librosa.display.specshow(Xdb, sr=sr, x_axis='time', y_axis='hz')
     plt.colorbar()
```

```
[6]: <matplotlib.colorbar.Colorbar at 0x22109a23a20>
```



Extracting Audio features in csv file

```
[7]: 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()
```

```
[12]: # data = pd.read_csv('data_new.csv')
      data = pd.read_csv('features_30_sec.csv')
      data.head()
```

[12]:

	filename	length	chroma_stft_mean	chroma_stft_var	rms_mean	rms_var	spectral_centroid_mean	spectral_centroid_var	spectral_bandwidth_mean	spectral_bandwidth_var	...	mfcc16_var	mfcc17_mean	mfcc17_var	m
0	blues.00000.wav	661794	0.350088	0.088757	0.130228	0.002827	1784.165850	129774.064525	2002.449060	85882.761315	...	52.420910	-1.690215	36.524071	
1	blues.00001.wav	661794	0.340914	0.094980	0.095948	0.002373	1530.176679	375850.073649	2039.036516	213843.755497	...	55.356403	-0.731125	60.314529	
2	blues.00002.wav	661794	0.363637	0.085275	0.175570	0.002746	1552.811865	156467.643368	1747.702312	76254.192257	...	40.598766	-7.729093	47.639427	
3	blues.00003.wav	661794	0.404785	0.093999	0.141093	0.006346	1070.106615	184355.942417	1596.412872	166441.494769	...	44.427753	-3.319597	50.206673	
4	blues.00004.wav	661794	0.308526	0.087841	0.091529	0.002303	1835.004266	343399.939274	1748.172116	88445.209036	...	86.099236	-5.454034	75.269707	

5 rows × 60 columns

```
[13]: # Remove Filename and Length. Save to X and Y
      data.pop('filename')
      data.pop('length')
      Y = pd.DataFrame(data.pop('label'))
      X = data
```

```
[14]: X.head()
```

[14]:

	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	...	mfcc16_mean	mfcc16_var	mfcc17_me
0	0.350088	0.088757	0.130228	0.002827	1784.165850	129774.064525	2002.449060	85882.761315	3805.839606	9.015054e+05	...	0.752740	52.420910	-1.690
1	0.340914	0.094980	0.095948	0.002373	1530.176679	375850.073649	2039.036516	213843.755497	3550.522098	2.977893e+06	...	0.927998	55.356403	-0.731
2	0.363637	0.085275	0.175570	0.002746	1552.811865	156467.643368	1747.702312	76254.192257	3042.260232	7.840345e+05	...	2.451690	40.598766	-7.729
3	0.404785	0.093999	0.141093	0.006346	1070.106615	184355.942417	1596.412872	166441.494769	2184.745799	1.493194e+06	...	0.780874	44.427753	-3.319
4	0.308526	0.087841	0.091529	0.002303	1835.004266	343399.939274	1748.172116	88445.209036	3579.757627	1.572978e+06	...	-4.520576	86.099236	-5.454

Normalize the data

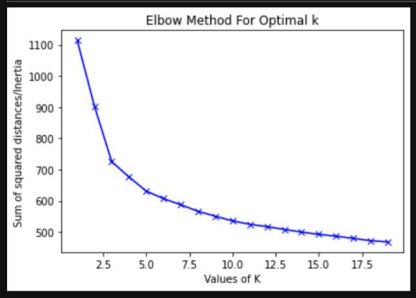
```
[16]: # Normalize the data
      min_max = preprocessing.MinMaxScaler()
      scaled_df = min_max.fit_transform(X.values)
      final_df = pd.DataFrame(scaled_df,columns=X.columns)
      final_df.head()
```

[16]:

	chroma_stft_mean	chroma_stft_var	rms_mean	rms_var	spectral_centroid_mean	spectral_centroid_var	spectral_bandwidth_mean	spectral_b
0	0.362279	0.695468	0.318188	0.101983	0.314117	0.040233	0.422879	
1	0.343622	0.793392	0.230894	0.085580	0.248405	0.121475	0.436889	
2	0.389832	0.640692	0.433652	0.099064	0.254261	0.049046	0.325334	
3	0.473508	0.777954	0.345856	0.229160	0.129376	0.058253	0.267404	
4	0.277759	0.681062	0.219641	0.083075	0.327270	0.110761	0.325514	

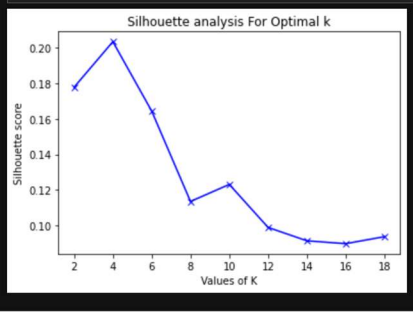
5 rows × 57 columns

[27]:
```
Sum_of_squared_distances = []
K = range(1,20)
for num_clusters in K :
    kmeans = KMeans(n_clusters=num_clusters)
    kmeans.fit(final_df)
    Sum_of_squared_distances.append(kmeans.inertia_)
plt.plot(K,Sum_of_squared_distances,'bx-')
plt.xlabel('Values of K')
plt.ylabel('Sum of squared distances/Inertia')
plt.title('Elbow Method For Optimal k')
plt.show()
```



[28]:
```
range_n_clusters = np.arange(2,20,2)
silhouette_avg = []
for num_clusters in range_n_clusters:
    kmeans = KMeans(n_clusters=num_clusters)
    kmeans.fit(final_df)
    cluster_labels = kmeans.labels_

    # silhouette score
    silhouette_avg.append(silhouette_score(final_df, cluster_labels))
plt.plot(range_n_clusters,silhouette_avg,'bx-')
plt.xlabel('Values of K')
plt.ylabel('Silhouette score')
plt.title('Silhouette analysis For Optimal k')
plt.show()
```



Modelling

Fitting a Neural Network

[31]:
```
# 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()
```

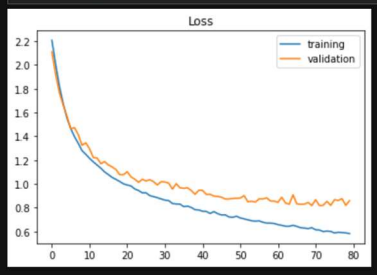
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 256)               14848
_________________________________________________________________
dense_1 (Dense)              (None, 10)                2570
=================================================================
Total params: 17,418
Trainable params: 17,418
Non-trainable params: 0
_________________________________________________________________
```

[32]:
```
model.compile(
  optimizer='adam',
  loss='categorical_crossentropy',
  metrics=['accuracy'],
)
```

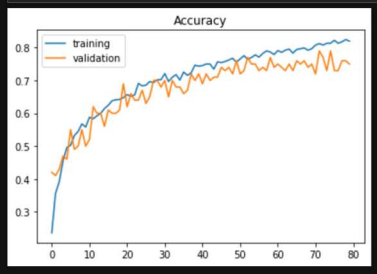
[33]:
```
history = model.fit(x_train, y_train, validation_data = (x_test, y_test), epochs=80, verbose = True)
```

```
Epoch 1/80
29/29 [==============================] - 0s 5ms/step - loss: 2.2060 - accuracy: 0.2356 - val_loss: 2.1089 - val_accuracy: 0.4200
Epoch 2/80
29/29 [==============================] - 0s 1ms/step - loss: 2.0062 - accuracy: 0.3556 - val_loss: 1.9230 - val_accuracy: 0.4100
Epoch 3/80
29/29 [==============================] - 0s 1ms/step - loss: 1.8204 - accuracy: 0.3922 - val_loss: 1.7644 - val_accuracy: 0.4300
Epoch 4/80
29/29 [==============================] - 0s 1ms/step - loss: 1.6753 - accuracy: 0.4556 - val_loss: 1.6629 - val_accuracy: 0.4700
Epoch 5/80
```

[34]:
```
records = history.history
plt.plot(records['loss'], label="training")
plt.plot(records['val_loss'], label="validation")
plt.legend()
plt.title("Loss");
```



[35]:
```
plt.plot(records['accuracy'], label="training")
plt.plot(records['val_accuracy'], label="validation")
plt.legend()
plt.title('Accuracy');
```



[36]:
```
y_preds = np.round(model.predict(x_test))
y_preds[:5]
```

[36]:
```
array([[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., 1., 0., 0., 0.],
       [0., 0., 0., 1., 0., 0., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0., 0., 0., 0., 0., 0.]], dtype=float32)
```

```
       [0., 1., 0., 0., 0., 0., 0., 0., 0.]], dtype=float32)
```

[37]:
```
print(classification_report(y_test, np.array(y_preds), target_names = Y['label'].unique()))
```

```
              precision    recall  f1-score   support

       blues       0.50      0.71      0.59         7
   classical       1.00      0.89      0.94         9
     country       0.91      0.62      0.74        16
       disco       0.71      0.71      0.71         7
      hiphop       1.00      0.50      0.67        10
        jazz       1.00      0.92      0.96        13
       metal       0.91      0.91      0.91        11
         pop       1.00      0.86      0.92         7
      reggae       0.67      0.67      0.67         9
        rock       0.33      0.09      0.14        11

   micro avg       0.83      0.68      0.75       100
   macro avg       0.80      0.69      0.73       100
weighted avg       0.82      0.68      0.73       100
 samples avg       0.68      0.68      0.68       100

C:\Users\dhiva\Anaconda3\envs\tensorflow\lib\site-packages\sklearn\metrics\_classification.py:1248: UndefinedMetricWarni
ntrol this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Using Random Forest Classifier :

[38]:
```
from sklearn.ensemble import RandomForestClassifier
```

[39]:
```
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]
```

[39]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=object)
```

[40]:
```
# 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
```

Predictor accuracy:

[42]:
```
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\dhiva\Anaconda3\envs\tensorflow\lib\site-packages\ipykernel_launcher.py:2: DataConversionWarning: A

Accuracy of the predictor is: 76%
```