In [4]:

```
%load_ext nb_black
```

Import Section

In [30]:

```
import os
import json
import numpy as np
import matplotlib.pyplot as plt
import re
import pandas as pd
import librosa
import tensorflow
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Input
from sklearn.model_selection import train_test_split
import IPython.display as ipd
from tensorflow.keras.models import load_model
from sklearn.preprocessing import LabelEncoder
```

In [5]:

```
DATASET_PATH = "COVID-19"
JSON_PATH = "metadata.json"
```

In [6]:

```
files = []

# r=root, d=directories, f = files

for r, d, f in os.walk(DATASET_PATH):
    for file in f:
        if ".wav" in file:
            files.append(os.path.join(r, file))
```

In [7]:

```
files[:9]
```

Out[7]:

```
['COVID-19/7DfMFXPDu3W2Fxjs8w0OsLIY8em1/vowel-o.wav',
 'COVID-19/7DfMFXPDu3W2Fxjs8w0OsLIY8em1/vowel-a.wav',
 'COVID-19/7DfMFXPDu3W2Fxjs8w0OsLIY8em1/breathing-shallow.wav',
 'COVID-19/7DfMFXPDu3W2Fxjs8w0OsLIY8em1/cough-shallow.wav',
 'COVID-19/7DfMFXPDu3W2Fxjs8w0OsLIY8em1/vowel-e.wav',
 'COVID-19/7DfMFXPDu3W2Fxjs8w0OsLIY8em1/cough-heavy.wav',
 'COVID-19/7DfMFXPDu3W2Fxjs8w0OsLIY8em1/breathing-deep.wav',
 'COVID-19/7DfMFXPDu3W2Fxjs8w0OsLIY8em1/counting-normal.wav',
 'COVID-19/7DfMFXPDu3W2Fxjs8w0OsLIY8em1/counting-fast.wav']
```

Feature Extraction

In [9]:

```
user_id = []
mfcc = []

for f in files:

    y, sr = librosa.load(f, sr=None)

    if y is None or len(y) == 0:
        continue

    else:
        user_id.append(f.split("/")[1])

        mfccs = librosa.feature.mfcc(y=y, sr=sr)
        m = []

        for e in mfccs:
            m.append(np.mean(e))
        mfcc.append(m)
```

In [10]:

```
mfcc = np.array(mfcc)
```

In [11]:

```
mfcc.shape
```

Out[11]:

```
(3798, 20)
```

In [12]:

```
len(files)
```

Out[12]:

3798

In [15]:

```
cols = ["mfcc_" + str(i) for i in range(1, 20 + 1)]
```

In [69]:

```
data = pd.DataFrame(mfcc, columns=cols)
```

In [70]:

```
# data["user_id"] = user_id
data["filepath"] = files
```

In [71]:

```
data.head()
```

Out[71]:

mfcc_16	mfcc_17	mfcc_18	mfcc_19	mfcc_20	filep
8.871928	-11.130017	-10.805934	-3.902681	-4.770747	COV 19/7DfMFXPDu3W2Fxjs8w0OsLIY8em1/vov o.v
5.872452	-1.326357	-1.137605	-7.114715	-6.406201	COV 19/7DfMFXPDu3W2Fxjs8w0OsLIY8em1/vov a.v
2.045947	0.254025	-0.853397	-0.025326	-0.848093	COV 19/7DfMFXPDu3W2Fxjs8w0OsLIY8em1/breath
0.439062	-1.520346	-1.996574	0.463630	-0.533586	COV 19/7DfMFXPDu3W2Fxjs8w0OsLIY8em1/cou s
7.020574	-10.257932	-9.524367	-6.419607	-11.246635	COV 19/7DfMFXPDu3W2Fxjs8w0OsLIY8em1/vov e.v

In [72]:

```
data["filename"] = data["filepath"].apply(lambda x: x.split("/")[2])
```

In [73]:

```
data.drop(["filepath"], inplace=True, axis=1)
```

In [74]:

```
data.head()
```

Out[74]:

	mfcc_1	mfcc_2	mfcc_3	mfcc_4	mfcc_5	mfcc_6	mfcc_7	mfcc_8
0	-363.377258	144.358337	27.980934	15.131627	-4.168990	-31.025049	-20.511789	-20.623474
1	-345.390533	124.867485	-4.307336	19.490623	7.525982	-17.402979	6.819692	-16.135691
2	-860.161987	97.513298	-21.191671	8.036111	6.223122	-6.836332	8.026308	2.952522
3	-525.321655	51.241253	-15.734446	-2.986843	5.281782	-3.511705	3.455106	-0.633038
4	-406.449951	121.712585	16.063465	29.269457	18.560997	-3.910539	12.661272	-5.819675

5 rows × 21 columns

In [75]:

```
file_le = LabelEncoder()
file_le.classes_ = np.load("file_le.npy", allow_pickle=True)
data["filename"] = file_le.transform(data["filename"])
```

In [76]:

```
# target data
cs_le = LabelEncoder()
cs_le.classes_ = np.load("cs_le.npy", allow_pickle=True)

target_data = np.load("target_data.npy")
```

In [77]:

```
data.values.shape
```

Out[77]:

```
(3798, 21)
```

In [78]:

```
# shape = (no. of samples , audio files , mfcc features + user_id + filen

mfcc_data = data.values.reshape((422, 9, 21))
```

In [79]:

```
mfcc_data[0][0]
```

Out[79]:

```
array([-363.3772583 ,  144.3583374 ,   27.98093414,   15.13162708,
         -4.16899014,  -31.02504921,  -20.51178932,  -20.62347412,
        -23.28576088,  -15.3525877 ,  -21.60122299,  -14.31004906,
          1.83776784,   -9.35709286,  -16.90119362,   -8.87192822,
        -11.13001728,  -10.80593395,   -3.90268111,   -4.77074718,
          8.        ])
```

In [80]:

```
data.iloc[0]
```

Out[80]:

```
mfcc_1     -363.377258
mfcc_2      144.358337
mfcc_3       27.980934
mfcc_4       15.131627
mfcc_5       -4.168990
mfcc_6      -31.025049
mfcc_7      -20.511789
mfcc_8      -20.623474
mfcc_9      -23.285761
mfcc_10     -15.352588
mfcc_11     -21.601223
mfcc_12     -14.310049
mfcc_13       1.837768
mfcc_14      -9.357093
mfcc_15     -16.901194
mfcc_16      -8.871928
mfcc_17     -11.130017
mfcc_18     -10.805934
mfcc_19      -3.902681
mfcc_20      -4.770747
filename      8.000000
Name: 0, dtype: float64
```

In [81]:

```
np.save("mfcc_data.npy", mfcc_data)
```

In [82]:

```
# features data and target data
mfcc_data.shape, target_data.shape
```

Out[82]:

```
((422, 9, 21), (422,))
```

Data Preparation

- Order of Features Must Be Same
  - mfcc (1 to 20)
  - user_id
  - Filename (filename => file_le.fit_transform)
- Target:
  - covid_status (covid_status => cs_le.fit_transform)

File Information

- **mfcc_data.npy** : our final feature array of shape (422 ,9 ,22)
- **targegt_data.npy** : final target array of shape (422 ,1)
- **cs_le.npy** : covid_status label encoder classes
- **file_le.npy** : filename label encoder classes

Deep Learning Model

In [83]:

```
n_classes = len(cs_le.classes_)
```

In [84]:

```
mfcc_data[0].shape
```

Out[84]:

```
(9, 21)
```

In [92]:

```
model = Sequential()

model.add(Input(shape=mfcc_data[0].shape))
model.add(Flatten())

model.add(Dense(units=16, activation="relu"))
model.add(Dense(units=32, activation="relu"))
model.add(Dense(units=64, activation="relu"))
model.add(Dense(units=64, activation="relu"))
# model.add(Dense(units=128, activation="relu"))
model.add(Dense(units=n_classes, activation="softmax"))
```

In [93]:

```
model.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_4 (Flatten)          (None, 189)               0
_________________________________________________________________
dense_22 (Dense)             (None, 16)                3040
_________________________________________________________________
dense_23 (Dense)             (None, 32)                544
_________________________________________________________________
dense_24 (Dense)             (None, 64)                2112
_________________________________________________________________
dense_25 (Dense)             (None, 64)                4160
_________________________________________________________________
dense_26 (Dense)             (None, 7)                 455
=================================================================
Total params: 10,311
Trainable params: 10,311
Non-trainable params: 0
_________________________________________________________________
```

In [94]:

```
model.compile(
    optimizer="adam",
    loss=tensorflow.keras.losses.SparseCategoricalCrossentropy(),
    metrics=["accuracy"],
)
```

Train Test Split

In [95]:

```
X_train, X_test, y_train, y_test = train_test_split(
    mfcc_data, target_data, test_size=0.20, random_state=0
)
```

In [96]:

```
X_train.shape, X_test.shape
```

Out[96]:

```
((337, 9, 21), (85, 9, 21))
```

In [97]:

```
y_train = np.array(y_train)
y_test = np.array(y_test)
y_train.shape, y_test.shape
```

Out[97]:

```
((337,), (85,))
```

Model Training

In [98]:

```
model.fit(X_train, y_train, batch_size=1, epochs=15, validation_data=(X_t
```

```
Epoch 1/15
337/337 [==============================] - 0s 733us/step - loss: 3.438
6 - accuracy: 0.5490 - val_loss: 1.3378 - val_accuracy: 0.5294
Epoch 2/15
337/337 [==============================] - 0s 515us/step - loss: 1.532
9 - accuracy: 0.6380 - val_loss: 1.2116 - val_accuracy: 0.7176
Epoch 3/15
337/337 [==============================] - 0s 525us/step - loss: 1.385
6 - accuracy: 0.6291 - val_loss: 1.1218 - val_accuracy: 0.7176
Epoch 4/15
337/337 [==============================] - 0s 522us/step - loss: 1.287
8 - accuracy: 0.6528 - val_loss: 1.0893 - val_accuracy: 0.7176
Epoch 5/15
337/337 [==============================] - 0s 517us/step - loss: 1.317
6 - accuracy: 0.6469 - val_loss: 1.0433 - val_accuracy: 0.7176
Epoch 6/15
337/337 [==============================] - 0s 526us/step - loss: 1.196
1 - accuracy: 0.6588 - val_loss: 1.0582 - val_accuracy: 0.7176
Epoch 7/15
337/337 [==============================] - 0s 523us/step - loss: 1.230
4 - accuracy: 0.6617 - val_loss: 1.0457 - val_accuracy: 0.7176
Epoch 8/15
337/337 [==============================] - 0s 518us/step - loss: 1.186
9 - accuracy: 0.6617 - val_loss: 1.0468 - val_accuracy: 0.7176
Epoch 9/15
337/337 [==============================] - 0s 515us/step - loss: 1.171
1 - accuracy: 0.6677 - val_loss: 1.0507 - val_accuracy: 0.7176
Epoch 10/15
337/337 [==============================] - 0s 501us/step - loss: 1.190
7 - accuracy: 0.6647 - val_loss: 1.0516 - val_accuracy: 0.7176
Epoch 11/15
337/337 [==============================] - 0s 522us/step - loss: 1.192
0 - accuracy: 0.6647 - val_loss: 1.0551 - val_accuracy: 0.7176
Epoch 12/15
337/337 [==============================] - 0s 523us/step - loss: 1.183
8 - accuracy: 0.6617 - val_loss: 1.0615 - val_accuracy: 0.7176
Epoch 13/15
337/337 [==============================] - 0s 538us/step - loss: 1.187
3 - accuracy: 0.6677 - val_loss: 1.0458 - val_accuracy: 0.7176
Epoch 14/15
337/337 [==============================] - 0s 566us/step - loss: 1.183
9 - accuracy: 0.6677 - val_loss: 1.0457 - val_accuracy: 0.7176
Epoch 15/15
337/337 [==============================] - 0s 591us/step - loss: 1.176
0 - accuracy: 0.6677 - val_loss: 1.0417 - val_accuracy: 0.7176
```

Out[98]:

```
<tensorflow.python.keras.callbacks.History at 0x7f4d944cf040>
```

In [99]:

```
model.evaluate(X_test, y_test)
# [loss , accuracy ]
```

```
3/3 [==============================] - 0s 768us/step - loss: 1.0417 -
accuracy: 0.7176
```

Out[99]:

```
[1.041690468788147, 0.7176470756530762]
```

In [67]:

```
model.save("model_2_71_mfcc.h5")
```

Model Acccuracy Comparision:

- Simple Feature Extraction : 0.7058823704719543
- MFCCs Feature Extraction : 0.7176470756530762

Next we will combine both features and then check for accuracy...