NAME:R.RAHINI ROLL NO:215229128

Lab8. Audio corpus creation and binary classification using DNN

1. Create a Dataset

Dataset is created

2. Read the Audio

In [2]:

```
import warnings
warnings.filterwarnings('ignore')

import os

import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
%matplotlib inline

import glob

from IPython.display import Audio
import IPython.display as ipd

import librosa #pip install librosa
import librosa.display

from tqdm import tqdm

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten
from tensorflow.keras.optimizers import Adam, RMSprop
from tensorflow.keras.utils import to_categorical

from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import *
```

In [3]:

```
paths = '1fly.wav'

Audio(paths)
```

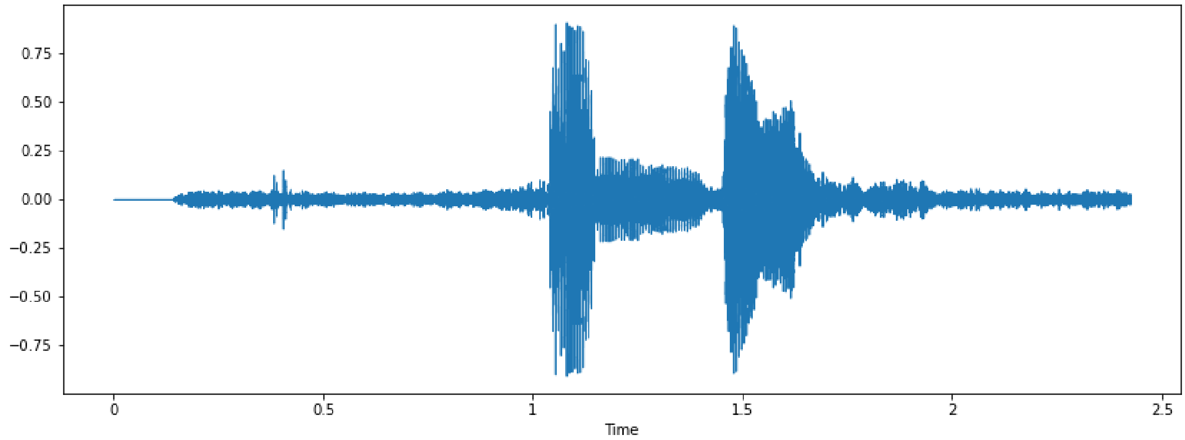
Out[3]:

0:00 / 0:00

In [4]:

```
data, sample_rate = librosa.load(paths)
plt.figure(figsize=(14,5))

librosa.display.waveshow(data, sr=sample_rate)
plt.show()
```



In [5]:

```
sample_rate
```

Out[5]:

```
22050
```

In [6]:

```
data
```

Out[6]:

```
array([0.        , 0.        , 0.        , ..., 0.01374257, 0.01699779,
       0.        ], dtype=float32)
```

In [7]:

```
stftt = librosa.feature.chroma_stft(y=data, sr=sample_rate)
stftt.shape
```

Out[7]:

```
(12, 105)
```

In [8]:

```
stftt
```

Out[8]:

```
array([[0.        , 0.        , 0.        , ..., 0.802695  , 1.        ,
        1.        ],
       [0.        , 0.        , 0.        , ..., 0.7765452 , 0.9367392 ,
        0.87225854],
       [0.        , 0.        , 0.        , ..., 0.64823526, 0.67778885,
        0.70276004],
       ...,
       [0.        , 0.        , 0.        , ..., 0.62063694, 0.42907584,
        0.4647396 ],
       [0.        , 0.        , 0.        , ..., 0.5343908 , 0.31281394,
        0.3764422 ],
       [0.        , 0.        , 0.        , ..., 0.7833488 , 0.6150779 ,
        0.60883486]], dtype=float32)
```

In [10]:

```
def features_extractor(file):
    audio, sample_rate = librosa.load(file_name)
    stftt_features = librosa.feature.chroma_stft(y=audio, sr=sample_rate)
    stftt_scaled_features = np.mean(stftt_features.T, axis=0)
    return stftt_scaled_features
```

In [11]:

```
extracted_features=[]

for index_num, row in tqdm(df.iterrows()):
    file_name = row[0]
    final_class_labels = row[1]
    data = features_extractor(file_name)
    extracted_features.append([data,final_class_labels])
```

```
20it [00:05,  3.52it/s]
```

In [12]:

```
extracted_features_df = pd.DataFrame(extracted_features, columns=['feature','class'])
extracted_features_df.head()
```

Out[12]:

	feature	class
0	[0.36663052, 0.38500747, 0.46057516, 0.5782328...	0
1	[0.33804783, 0.36386037, 0.39901736, 0.4628456...	0
2	[0.525747, 0.41264907, 0.22864214, 0.3536954, ...	0
3	[0.34989068, 0.30691823, 0.37625653, 0.2727835...	0
4	[0.33986202, 0.34667534, 0.28150865, 0.2368082...	0

3. Split the dataset

In [13]:

```
X = np.array(extracted_features_df['feature'].tolist())
y = np.array(extracted_features_df['class'])
```

In [14]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=
```

In [15]:

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

```
(15, 12)
(15,)
(5, 12)
(5,)
```

4. Train a Neural Network Model

In [16]:

```
batch_size=132
num_labels = y.shape[0]
```

In [18]:

```
model=Sequential()
model.add(Dense(128, activation='tanh', input_shape=(12,)))
model.add(Dense(64, activation='tanh'))
model.add(Dense(32, activation='tanh'))
model.add(Dense(16, activation='tanh'))
model.add(Dense(8, activation='tanh'))
model.add(Dense(1, activation='sigmoid'))
model.summary()
model.compile(loss='mean_squared_error',metrics=['accuracy'],optimizer='adam')
history=model.fit(X_train, y_train, batch_size=batch_size, epochs=50 , verbose=2,validat
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 dense (Dense)               (None, 128)               1664      
                                                                 
 dense_1 (Dense)             (None, 64)                8256      
                                                                 
 dense_2 (Dense)             (None, 32)                2080      
                                                                 
 dense_3 (Dense)             (None, 16)                528       
                                                                 
 dense_4 (Dense)             (None, 8)                 136       
                                                                 
 dense_5 (Dense)             (None, 1)                 9         
                                                                 
=================================================================
Total params: 12,673
Trainable params: 12,673
```

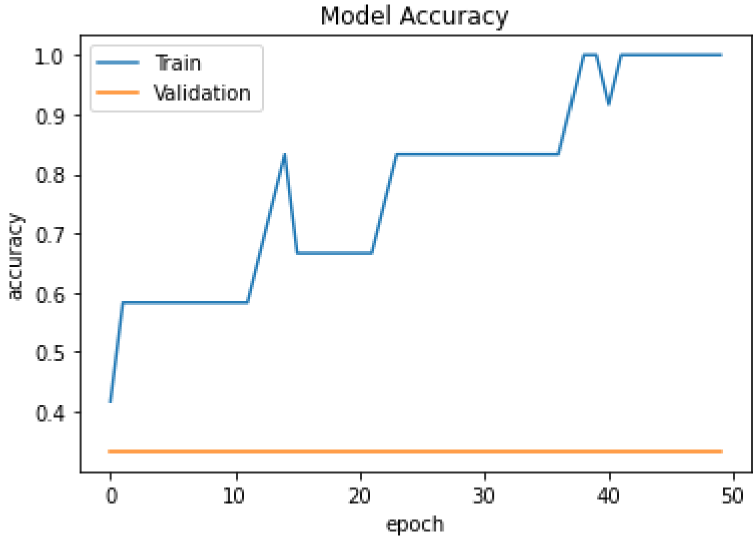
In [20]:

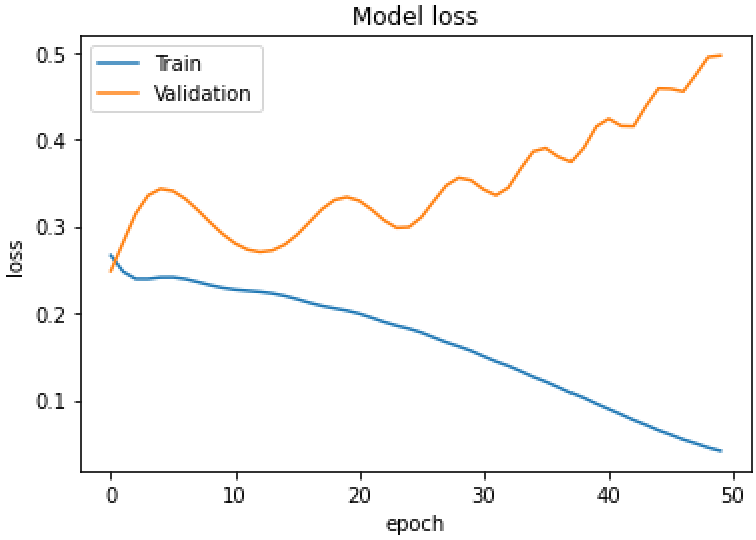
```
score=model.evaluate(X_test,y_test,verbose=0)
print("Loss :", score[0])
print("Accuracy :",score[1])
```

```
Loss : 0.19610339403152466
Accuracy : 0.800000011920929
```

In [21]:

```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['Train', 'Validation'])
plt.show()
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['Train', 'Validation'])
plt.show()
```





In [22]:

```
y_pred=model.predict(X_test)
y_pred=(y_pred>0.5)*1
y_pred
```

```
1/1 [==============================] - 0s 234ms/step
```

Out[22]:

```
array([[1],
       [0],
       [0],
       [0],
       [1]])
```

In [23]:

```
y_pred = model.predict(X_test).round()
y_pred
```

```
1/1 [==============================] - 0s 56ms/step
```

Out[23]:

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

In [24]:

```
print(" Accuracy ",accuracy_score(y_test,y_pred))
print(" Precision ",precision_score(y_test,y_pred))
print(" Recall ",recall_score(y_test,y_pred))
print(" AUC ",roc_auc_score(y_test,y_pred))
```

```
 Accuracy  0.8
 Precision  1.0
 Recall  0.6666666666666666
 AUC  0.8333333333333333
```

5. Run different Neural Network models

In [25]:

```
def c_model(node):
    model=Sequential()
    model.add(Dense(128, activation='tanh', input_shape=(12,)))
    model.add(Dense(node, activation='tanh'))
    model.add(Dense(node, activation='tanh'))
    model.add(Dense(node, activation='tanh'))
    model.add(Dense(8, activation='tanh'))
    model.add(Dense(1, activation='sigmoid'))
    model.summary()
    model.compile(loss='mean_squared_error',metrics=['accuracy'],optimizer='adam')
    history=model.fit(X_train, y_train, batch_size=batch_size, epochs=50 , verbose=2,va
    score=model.evaluate(X_test,y_test,verbose=0)
    print("loss ", score[0])
    print("accuracy ",score[1])
    plt.plot(history.history['accuracy'])
    plt.plot(history.history['val_accuracy'])
    plt.title('Model Accuracy')
    plt.ylabel('accuracy')
    plt.xlabel('epoch')
    plt.legend(['Train', 'Validation'])
    plt.show()
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.title('Model loss')
    plt.ylabel('loss')
    plt.xlabel('epoch')
    plt.legend(['Train', 'Validation'])
    plt.show()
```

In [26]:

```
c_model(8)
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 dense_6 (Dense)             (None, 128)               1664      
                                                                 
 dense_7 (Dense)             (None, 8)                 1032      
                                                                 
 dense_8 (Dense)             (None, 8)                 72        
                                                                 
 dense_9 (Dense)             (None, 8)                 72        
                                                                 
 dense_10 (Dense)            (None, 8)                 72        
                                                                 
 dense_11 (Dense)            (None, 1)                 9         
                                                                 
=================================================================
Total params: 2,921
Trainable params: 2,921
Non-trainable params: 0
```

In [27]:

```
c_model(16)
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 dense_12 (Dense)            (None, 128)               1664      
                                                                 
 dense_13 (Dense)            (None, 16)                2064      
                                                                 
 dense_14 (Dense)            (None, 16)                272       
                                                                 
 dense_15 (Dense)            (None, 16)                272       
                                                                 
 dense_16 (Dense)            (None, 8)                 136       
                                                                 
 dense_17 (Dense)            (None, 1)                 9         
                                                                 
=================================================================
Total params: 4,417
Trainable params: 4,417
Non-trainable params: 0
```

In [28]:

```
c_model(32)
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 dense_18 (Dense)            (None, 128)               1664      
                                                                 
 dense_19 (Dense)            (None, 32)                4128      
                                                                 
 dense_20 (Dense)            (None, 32)                1056      
                                                                 
 dense_21 (Dense)            (None, 32)                1056      
                                                                 
 dense_22 (Dense)            (None, 8)                 264       
                                                                 
 dense_23 (Dense)            (None, 1)                 9         
                                                                 
=================================================================
Total params: 8,177
Trainable params: 8,177
Non-trainable params: 0
```

In [29]:

```
c_model(64)
```

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 dense_24 (Dense)            (None, 128)               1664      
                                                                 
 dense_25 (Dense)            (None, 64)                8256      
                                                                 
 dense_26 (Dense)            (None, 64)                4160      
                                                                 
 dense_27 (Dense)            (None, 64)                4160      
                                                                 
 dense_28 (Dense)            (None, 8)                 520       
                                                                 
 dense_29 (Dense)            (None, 1)                 9         
                                                                 
=================================================================
Total params: 18,769
Trainable params: 18,769
```

In [33]:

```
def c_layer(n):
    model=Sequential()
    model.add(Dense(128, activation='tanh', input_shape=(12,)))
    for i in range(0,n):
        model.add(Dense(32, activation='tanh'))

    model.add(Dense(1, activation='sigmoid'))
    model.summary()
    model.compile(loss='mean_squared_error',metrics=['accuracy'],optimizer='adam')
    history=model.fit(X_train, y_train, batch_size=batch_size, epochs=50 , verbose=2,va
    score=model.evaluate(X_test,y_test,verbose=0)
    print("loss ", score[0])
    print("accuracy ",score[1])
    plt.plot(history.history['accuracy'])
    plt.plot(history.history['val_accuracy'])
    plt.title('Model Accuracy')
    plt.ylabel('accuracy')
    plt.xlabel('epoch')
    plt.legend(['Train', 'Validation'])
    plt.show()
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.title('Model loss')
    plt.ylabel('loss')
    plt.xlabel('epoch')
    plt.legend(['Train', 'Validation'])
    plt.show()
```

In [34]:

```
c_layer(2)
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 dense_30 (Dense)            (None, 128)               1664      
                                                                 
 dense_31 (Dense)            (None, 32)                4128      
                                                                 
 dense_32 (Dense)            (None, 32)                1056      
                                                                 
 dense_33 (Dense)            (None, 1)                 33        
                                                                 
=================================================================
Total params: 6,881
Trainable params: 6,881
Non-trainable params: 0
_________________________________________________________________
Epoch 1/50
1/1 - 1s - loss: 0.2464 - accuracy: 0.5833 - val_loss: 0.2624 - val_accu
```

In [35]:

```
c_layer(3)
```

```
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 dense_34 (Dense)            (None, 128)               1664      
                                                                 
 dense_35 (Dense)            (None, 32)                4128      
                                                                 
 dense_36 (Dense)            (None, 32)                1056      
                                                                 
 dense_37 (Dense)            (None, 32)                1056      
                                                                 
 dense_38 (Dense)            (None, 1)                 33        
                                                                 
=================================================================
Total params: 7,937
Trainable params: 7,937
Non-trainable params: 0
_________________________________________________________________
Epoch 1/50
```

In [36]:

```
c_layer(5)
```

```
Model: "sequential_7"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 dense_39 (Dense)            (None, 128)               1664      
                                                                 
 dense_40 (Dense)            (None, 32)                4128      
                                                                 
 dense_41 (Dense)            (None, 32)                1056      
                                                                 
 dense_42 (Dense)            (None, 32)                1056      
                                                                 
 dense_43 (Dense)            (None, 32)                1056      
                                                                 
 dense_44 (Dense)            (None, 32)                1056      
                                                                 
 dense_45 (Dense)            (None, 1)                 33        
                                                                 
=================================================================
```