```
from google.colab import drive

drive._mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mour
```

```
!zip -r '/content/drive/MyDrive/ml_cp/G_13".zip"' '/content/drive/MyDrive/ml_cp/G_13"'
```

```
        zip warning: name not matched: /content/drive/MyDrive/ml_cp/G_13"

zip error: Nothing to do! (try: zip -r /content/drive/MyDrive/ml_cp/G_13".zip" . -i /cor
```

```
!zip "/content/drive/MyDrive/ml_cp/G_13.zip" "/content/drive/MyDrive/ml_cp/G_13/Extracted"
```

```
        zip warning: name not matched: /content/drive/MyDrive/ml_cp/G_13/Extracted

zip error: Nothing to do! (/content/drive/MyDrive/ml_cp/G_13.zip)
```

```
!pip install unrar
!unrar x "/content/drive/MyDrive/ml_cp/G_13.rar" "/content/drive/MyDrive/ml_cp/G_13"
```

```
Requirement already satisfied: unrar in /usr/local/lib/python3.7/dist-packages (0.4)

UNRAR 5.50 freeware      Copyright (c) 1993-2017 Alexander Roshal

Cannot open /content/drive/MyDrive/ml_cp/G_13.rar
No such file or directory
No files to extract
```

```
import librosa
import soundfile
import os, glob, pickle
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score
```

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
```

```
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
import tensorflow as tf
from tensorflow import keras


def extract_feature(file_name, mfcc, chroma, mel):
    with soundfile.SoundFile(file_name) as sound_file:
        X = sound_file.read(dtype="float32")
        sample_rate=sound_file.samplerate
        if chroma:
            stft=np.abs(librosa.stft(X))
        result=np.array([])
        if mfcc:
            mfccs=np.mean(librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=40).T, axis=0)
            result=np.hstack((result, mfccs))
        if chroma:
            chroma=np.mean(librosa.feature.chroma_stft(S=stft, sr=sample_rate).T,axis=0)
            result=np.hstack((result, chroma))
        if mel:
            mel=np.mean(librosa.feature.melspectrogram(X, sr=sample_rate).T,axis=0)
            result=np.hstack((result, mel))
    return result


emotions={
  'N':'Neutral',
  'H':'Happy',
  'S':'Sad',
  'A':'Angry'

}
a={
    '01':'Angry',
   '02':'Happy',
   '03':'Neutral',
   '04':'Sad'
}
observed_emotions=['Angry', 'Happy', 'Neutral', 'Sad']


!pip install pyunpack
!pip install patool
from pyunpack import Archive
Archive('/content/drive/MyDrive/ml_cp/G_13.zip').extractall('/content/drive/MyDrive/G13')
```

```
Requirement already satisfied: pyunpack in /usr/local/lib/python3.7/dist-packages (0.2.2
Requirement already satisfied: easyprocess in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: entrypoint2 in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: patool in /usr/local/lib/python3.7/dist-packages (1.12)
```

```
def load_data(test_size=0.3):
```

```
    x,y=[],[]
    for file in glob.glob("/content/drive/MyDrive/G13/Extracted/Emotion_*/*.wav"):
        file_name=os.path.basename(file)
        print(file)
        emotion=emotions[file_name.split("_")[3]]
        if emotion not in observed_emotions:
            print(emotion)
            continue
        feature=extract_feature(file, mfcc=True, chroma=True, mel=True)
        x.append(feature)
        y.append(emotion)
    return train_test_split(np.array(x), y, test_size=test_size, random_state=9)


import pathlib
data_dir = pathlib.Path("/content/drive/MyDrive/G13/Extracted")
image_count = len(list(data_dir.glob('*/*.wav')))
print(image_count)
```

```
1342
```

```
x_train,x_test,y_train,y_test=load_data(test_size=0.25)
```

```
x_train
```

```
array([[-4.01113556e+02,  1.74065155e+02, -3.21291351e+01, ...,
         5.86960425e-07,  6.45651767e-07,  6.74027888e-07],
       [-5.03315338e+02,  1.58175858e+02,  1.69029484e+01, ...,
         7.65015429e-08,  3.05694350e-08,  2.67238569e-08],
       [-4.34067017e+02,  1.99254303e+02, -3.26265678e+01, ...,
         8.04445222e-09,  7.90137022e-09,  7.84838061e-09],
       ...,
       [-2.91048859e+02,  1.61659973e+02, -1.64385185e+01, ...,
         8.50290682e-09,  8.68795080e-09,  8.82453399e-09],
       [-4.49952606e+02,  1.79362015e+02, -2.17644119e+01, ...,
         7.86225129e-09,  7.93160826e-09,  8.04218381e-09],
       [-3.13023621e+02,  1.60178955e+02, -3.96102371e+01, ...,
         5.43542313e-08,  5.27078932e-08,  4.77053455e-08]])
```

```
print((x_train.shape[0], x_test.shape[0]))
```

```
(1006, 336)
```

```
print(f'Features extracted: {x_train.shape[1]}')
```

```
Features extracted: 180
```

```
print(f'Features extracted: {x_train.shape[1]}')
```

```
Features extracted: 180
```

```
acc_svm = []
clf_svm = SVC(kernel='linear')
clf_svm.fit(x_train, y_train)
acc_svm.append(clf_svm.score(x_test, y_test))
print("The average accuracy for SVM classifier is {}%".format(np.average(acc_svm)*100))
```

```
The average accuracy for SVM classifier is 82.73809523809523%
```

```
feature=extract_feature("/content/drive/MyDrive/G13/Extracted/Emotion_Sad/SP001_M_R001_S_S01.
feature=feature.reshape(1,-1)
out=clf_svm.predict(feature)
print(out)
```

```
['Sad']
```

```
clf=RandomForestClassifier(n_estimators = 200, criterion = 'entropy',max_depth=200,random_sta
clf.fit(x_train,y_train)
y_pred1=clf.predict(x_test)
print("Accuracy for RF",accuracy_score(y_test,y_pred1)*100)
```

```
Accuracy for RF 88.98809523809523
```

```
feature=extract_feature("/content/drive/MyDrive/G13/Extracted/Emotion_Sad/SP001_M_R001_S_S01.
feature=feature.reshape(1,-1)
out=clf.predict(feature)
print(out)
```

```
['Sad']
```

```
acc_lr = []
clf_lr = make_pipeline(StandardScaler(), LogisticRegression(solver='saga', max_iter=200, rand
clf_lr.fit(x_train, y_train)
acc_lr.append(clf_lr.score(x_test, y_test))
print("The average accuracy for Logistic Regression is {}%".format(np.average(acc_lr)*100))
```

```
The average accuracy for Logistic Regression is 82.14285714285714%
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_sag.py:354: ConvergenceWarn
  ConvergenceWarning,
```

```
feature=extract_feature("/content/drive/MyDrive/G13/Extracted/Emotion_Sad/SP001_M_R001_S_S01.
feature=feature.reshape(1,-1)
out=clf_lr.predict(feature)
print(out)
```

```
['Sad']
```

✓ 14s completed at 11:03 PM