# به نام خدا

دانشگاه تهران

پردیس دانشکده های فنی

دانشکده مهندسی برق و کامپیوتر

---

# تمرین شماره‌ی 2

## بخش عملی

**استاد درس:**

دکتر فدائی

دکتر یعقوب‌زاده

**نگارش:**

فاطمه محمدی 810199489

# پردازش سیگنال

## Hidden Markov Models

## اهداف:

توسعه یک سیستم تشخیص گفتار اعداد براساس HMM میباشد.

## مقدمه:

Hidden Markov Models (HMM) یکی از ابزارهای مهم در زمینه پردازش سیگنال های زمانی و تشخیص الگو هستند. در این مدل ها هر حالت به یک وضعیت مشخص مرتبط می شود و احتمال حرکت از یک وضعیت به وضعیت دیگر با توجه به وضعیت فعلی تعیین می شود. از این رو، HMM به عنوان یک ابزار قدتمند برای مدلسازی سیستم های پویا و تصمیم گیری در شرایط عدم قطعیت به شمار می آید. در زمینه تشخیص گفتار، HMM معمولاً برای مدل سازی دقیق تر وضعیت های مختلف گفتار، مانند حروف، کلمات یا فریم های زمانی کوتاه از گفتار، استفاده می شود.

## تعریف پروژه:

همانطور که اشاره شد هدف اصلی این پروژه توسعه یک سیستم تشخیص گفتار اعداد براساس HMM میباشد. برای اینکار مجموعه داده صوتی ای استفاده میشود و باید مراحل پیش پردازش، استخراج ویژگی، پیاده سازی مدل و تحلیل آن در طی پروژه انجام داد.

## داده‌ها و پارامترهای مسئله:

در این پروژه که یک پروژه داده محور است، مجموعه داده‌ای مورد نیاز، شامل ضبط های گفتاری از 6 گوینده مختلف است که هر کدام 50 بار ارقام 0 تا 9 را به زبان انگلیسی تلفظ می کنند. نام هر فایل صوتی به صورت {digitLabel}_{speakerName}_{index}.wav است که به ترتیب عدد گفته شده، نام گوینده و شماره نمونه را نشان میدهد.

# پیاده سازی گام به گام پروژه:

در ادامه به توضیح هر گام از پروژه به همراه علت انجام آن و تحلیل خروجی توابع صدا زده در هر گام میپردازیم.

## بخش صفر: مروری بر ورودی ها، کتابخانه ها و داده های ضروری:

### تعریف کتابخانه‌های مورد نیاز:

در این پروژه به کتابخانه های زیر نیاز داریم:

```
import librosa
import numpy as np
import os
import soundfile as sf
import matplotlib.pyplot as plt
import noisereduce as nr
from scipy.spatial.distance import euclidean
from sklearn.preprocessing import MinMaxScaler, StandardScaler
from hmmlearn import hmm
from sklearn.model_selection import train_test_split
from colorama import Fore, Style
from collections import defaultdict
import scipy.stats
```

در ادامه علت نیاز به هریک از کتابخانه ها را بررسی خواهیم کرد.

### تعریف پارامترها:

در قدم اول پارامترهایی که نیاز داریم را مشخص میکنیم،(برخی از آنها در مراحل جلو تر اضافه میشوند):

```
initial_recordings_folder = 'recordings'
preprocessed_recordings_folder = 'preprocessed_recordings'
mfcc_features_folder = 'mfcc_features'
heatmaps_folder = "heatmaps"

DIGITS = []
SPEAKERS = []

TARGET_SAMPLING_RATE = 16000

N_MFCC = 13
N_FRAME_MFCC = 30

NUM_REPEATED_RECORDING = 50

TRAIN_PERCENT = 0.3
TEST_PERCENT = 1 - TRAIN_PERCENT

NUM_STATE = 13
NUM_ITERATION = 10
```

- در ابتدا پوشه هایی که نیاز داریم داریم را مشخص میکنیم، که به ترتیب شامل رکورد ها اولیه، رکورد های پیش پردازش شده، MFCCها و Heatmap های MFCC میباشند.
- در ادامه دو set، داریم که مقادیر یکتای از اعداد و یا افرادی که باید تشخیص داده شوند، در آنها ذخیره خواهد شد.
- برای یکنواخت شدن نرخ خواندن رکورد ها و feature mfcc و بهتر بررسی کردن آنها چند معیار باید مشخص کرد، که در ادامه نیز آنها را بیشتر بررسی میکنیم، این معیار ها شامل نرخ نمونه برداری از داده، N_MFCC و N_FRAME_MFCC است که در ادامه به بررسی این دو خواهیم پرداخت.
- در ادامه تعداد رکورد ها به ازای هر رقم که فردگفته را مشخص میکنیم.
- برای تقسیم داده ها به دو دسته Train و Test نیز باید درصد هر کدام را مشخص کنیم.
- در اخر برای مدل های HMM تعداد Hidden State ها و تعداد تکرار (برای Train) را مشخص میکنیم.

# بخش اول: پیش پردازش و استخراج ویژگی:

## پیش پردازش:

پیش پردازش داده ها اولین گام در هر پروژه داده محور است. هدف از پیش پردازش داده ها، بهبود کیفیت داده ها و آماده سازی آنها برای آموزش و آزمایش سیستم تشخیص گفتار است. مراحل معمول پیش پردازش داده ها می تواند شامل افزایش داده، حذف نویز، تقویت سیگنال، نرمال سازی و قطعه بندی باشد.

### پرسش 1:

آیا قطعه بندی داده ها برای این دیتاست مفید است؟ چرا؟

پاسخ:

- در صورتی که منظور قطعه بندی هر وویس باشد، پاسخ خیر است، چرا که در هر فایل ضبط شده تنها یک رقم توسط یک نفر گفته شده است. درواقع قطعه بندی برای آن است که درصورتی که در یک فایل چندین فرد صحبت کرده باشند و یا چندین رقم بیان شده باشد، آنها را جدا کنیم، که در این دیتاست این کار از قبل انجام شده است.
- درصورتی که منظور دسته بندی وویس ها به چند گروه براساس گوینده یا عدد گفته شده باشد، میتوانیم این دسته بندی را داشته باشیم تا سریع تر داده ها را را بررسی کنیم، البته توجه شود با توجه به این که در این پروژه به دو هدف از داده ها استفاده میکنیم و همچنین داده هایی که برای Train هستند را فقط میتوان دسته بندی کرد، این عمل را در فاز های بعدی انجام میدهیم. (توجه شود در واقعیت مواردی هستند که داده های مورد تست ناشناخته هستند و نمیتوان دسته بندی کرد.)
  البته این کار را میتوانستیم در این فاز هم انجام بدهیم، اما فعلا آن را بعد از دسته بندی داده ها به دو گروه Train , Test انجام میدهیم.
  دلیل آنکه قبل از این فاز دسته بندی به دو گروه Testو Train را انجام نداده ایم ساده تر شدن برنامه است چرا که تمامی مراحل این فاز باید روی همه داده ها (هم Testو هم Train) انجام شوند.

❖ دلایل اهمیت قطعه بندی:
- کاهش پیچیدگی داد ها
- تجزیه تحلیل سریع و ساده تر
- بهبود دقت
- بهبود سرعت پردازش
- بهنیه سازی منابع
- سازگاری بیشتر

در این بخش همانطور که اشاره شده ما ابتدا برخی از پیش پردازش ها و استخراج ویژگی ها را انجام میدهیم، اما قطعه بندی را فعلا انجام نمیدهیم:

1. در گام اول یک تابع تعریف میکینم که ابتدا یک تابع تعریف میکنیم که برای یک رکورد پیش پردازشی شامل 1. کاهش نویز، 2. حذف بخش silent 3. نرمال سازی را انجام میدهد:

```python
def preprocess_audio(input_path, output_path, target_sampling_rate = TARGET_SAMPLING_RATE):
    # Load the audio file
    audio, sampling_rate = librosa.load(input_path, sr = target_sampling_rate)

    # Noise reduction
    reduced_noise_audio = nr.reduce_noise(y=audio, sr = sampling_rate)

    # Silence removal
    non_silent_audio, _ = librosa.effects.trim(audio)

    # Normalize the audio to a standard volume level
    normalized_audio = librosa.util.normalize(non_silent_audio)

    # Save the processed audio file
    sf.write(output_path, normalized_audio, target_sampling_rate)
```

2. سپس تابعی مینویم که این عمل  را برای تمام رکورد های اولیه انجام دهد و آنها را در پوشه دیگری ذخیره کند:

```python
def preprocess_all_audio(input_folder, output_folder):
    if not os.path.exists(output_folder):
        os.makedirs(output_folder)

    for file_name in os.listdir(input_folder):
        if file_name.endswith('.wav'):
            file_path = os.path.join(input_folder, file_name)
            output_path = os.path.join(output_folder, file_name)
            preprocess_audio(file_path, output_path)
            #print(f"Processed {file_name}")
```

3. در اخرین قدم این تابع را فراخوانی میکنیم:

```python
preprocess_all_audio(initial_recordings_folder, preprocessed_recordings_folder)
```

## استخراج ویژگی:

استخراج و انتخاب ویژگی از مهم ترین مراحل هر پروژه هوش مصنوعی و یادگیری ماشین می باشند. استفاده از ویژگی مناسب می تواند تاثیر بالایی در خروجی مدل نهایی داشته باشد.

همچنین ویژگی های بسیار متفاوتی را می توان از یک محتوای صوتی استخراج کرد که هر کدام اطلاعات گوناگونی را به ما می دهند. از جمله این ویژگی ها می توان mel-spectrogram، Zero Crossing Rate، mfcc و chroma features را نام برد.

### پرسش 2:

در مورد هر کدام از این ویژگی ها تحقیق کنید و روابط بین آن ها را توضیح دهید.

پاسخ:

- همانطور که قبلتر اشاره کرده ایم، استخراج ویژگی یک مرحله مهم در پردازش صدا و وظایف تشخیص گفتار است که داده‌های صوتی خام را به نمایشی آموزنده‌تر و فشرده‌تر تبدیل می‌کند. که از جمله این ویژگی ها می توان mel-spectrogram، Zero Crossing Rate، mfcc و chroma featuresرا نام برد.
  در ادامه به توضیح مختصری درباره ی هریک از این ویژگی ها میپردازیم:

**1. Mel-Spectrogram**

mel-spectrogram یک نمایش بصری از طیف فرکانس‌های یک سیگنال صوتی است که با زمان تغییر می‌کنند و یک مقیاس mel در محور فرکانس دارد. مقیاس mel به گونه ای طراحی شده است که پاسخ گوش انسان را بیشتر از باندهای فرکانسی با فاصله خطی مورد استفاده در طیف سنجی استاندارد تقلید کند. این باعث می‌شود که

طیف‌نگار mel به ویژه برای پردازش گفتار و موسیقی مفید باشد زیرا روی فرکانس‌هایی که برای درک شنوایی انسان مهم هستند تأکید می‌کند..

**2. Zero Crossing Rate (ZCR)**

ZCR اندازه گیری تعداد دفعاتی است که سیگنال صوتی از محور صفر در یک بازه زمانی معین عبور می کند. این یک ویژگی ساده و در عین حال موثر برای تجزیه و تحلیل ویژگی های زمانی صدا است. مقادیر بالای ZCR معمولاً اجزای پر سر و صدا یا با فرکانس بالا را نشان می دهد، در حالی که مقادیر پایین تر نشان دهنده صداهای تونال یا هارمونیک هستند. ZCR می‌تواند در تمایز بخش‌های گفتاری صدادار از بی‌صدا، در میان سایر برنامه‌ها مفید باشد.

**3. Mel-Frequency Cepstral Coefficients (MFCCs)**

MFCC ها ضرایبی هستند که در مجموع یک MFC را تشکیل می دهند. آنها از نمایش واقعی مغزی کلیپ صوتی مشتق شده اند، که در آن باندهای فرکانس خطی نیستند اما بر اساس مقیاس mel توزیع می شوند. این تقریباً پاسخ سیستم شنوایی انسان را دقیق‌تر از هر طیف‌نگار فرکانس خطی تقریب می‌کند. MFCC به طور گسترده ای در تشخیص گفتار و طبقه بندی ژانر موسیقی استفاده می شود، زیرا آنها به طور موثر ویژگی های صدا و متن سیگنال های صوتی را ضبط می کنند.

**4. Chroma Features**

ویژگی‌های Chroma ابزاری قدرتمند برای تجزیه و تحلیل موسیقی است که شدت هر یک از ۱۲ کلاس مختلف را بدون توجه به اکتاو(octave) نشان می‌دهد. برای تشخیص گفتار، ویژگی‌های کروما کمتر به طور مستقیم مورد استفاده قرار می‌گیرند، اما می‌توانند بینش‌هایی را در مورد میزان آهنگ و محتوای هارمونیک در سیگنال‌های صوتی که ملودی و هارمونی مرتبط هستند، ارائه دهند.

## روابط و کاربردها

- **Mel-Spectrogram و MFCCs:**

هر دو بر اساس مقیاس mel هستند که برای بازتاب شنوایی انسان طراحی شده است.

در حالی که mel-spectrogram ها نمایش متراکمی از سیگنال مفید برای تحلیل بصری و مدل های یادگیری عمیق را ارائه می دهند، MFCC ها این اطلاعات را در مجموعه فشرده ای از ویژگی ها خلاصه می کنند که مهم ترین جنبه های صدا را همانطور که توسط انسان درک می شود، به تصویر می کشد، و آنها را برای مدلهای یادگیری ماشین سنتی بسیار کارآمد می کند.

- **ZCR and Chroma Features:**

ZCR اندازه گیری ساده ای از تغییرات محتوای فرکانس سیگنال را ارائه می دهد که برای شناسایی بخش های گفتار یا تشخیص ریتم در موسیقی مفید است. Chroma Features ، اگرچه پیچیده‌تر هستند، اما نمای دقیق‌تری از محتوای هارمونیک ارائه می‌دهند. هر دو می توانند مکمل یکدیگر در کاربردهایی باشند که از درک هر دو جنبه ریتمیک و هارمونیک صدا، مانند سیستم های آنالیز موسیقی پیشرفته بهره می برند.

## تشخیص گفتار:

برای تشخیص گفتار، MFCCها به دلیل اثربخشی آنها در گرفتن ویژگی های کلیدی گفتار، اغلب انتخاب اصلی هستند. می توان از Mel-spectrograms نیز استفاده کرد، به خصوص با deep learning models که می توانند از نمایش متراکم آنها یاد بگیرند. ZCR می‌تواند به تقسیم‌بندی گفتار از سکوت یا نویز کمک کند، در حالی که ویژگی‌های chroma کمتر در برنامه‌های گفتاری خالص استفاده می‌شوند، اما می‌توانند زمینه اضافی را در وظایف مربوط به آواز خواندن یا جایی که اطلاعات زیر و بم مرتبط هستند، فراهم کند.

به طور خلاصه، هر یک از این ویژگی ها نقاط قوت و ضعف خود را دارند و برای جنبه های خاصی از تجزیه و تحلیل صدا مناسب هستند. MFCCها و mel-spectrograms ارتباط نزدیکی دارند و برای تشخیص گفتار بسیار مرتبط هستند، در حالی که ZCR در تشخیص و تقسیم‌بندی گفتار ارزش می‌افزاید. ویژگی‌های Chroma، در حالی که در زمینه گفتار جا افتاده‌تر هستند، بینش‌هایی را در مورد محتوای هارمونیک ارائه می‌دهند که می‌توانند سایر ویژگی‌ها را در برنامه‌های دارای تمایل به موسیقی تکمیل کنند.

## استخراج MFCC:

در این پروژه از ضرایب Mel Frequency Cepstral Coefficient (MFCC) برای دسته بندی record ها استفاده میکنیم.

ضرایب MFCC به طور گسترده در زمینه های مرتبط با دسته بندی موسیقی و تشخیص گفتار استفاده شده اند.

در ادامه مجموعه ضرایب MFCC را برای هر کدام از نمونه های داده شده استخراج میکنیم.

1. یکی از اقدام مهم برای داشتن MFCCs هایی با shape های یکسان، حفظ ساختار زمانی است که میتوان به صورت های متفاوت انجام شود که در اینجا ما از روش paddingاستفاده میکنیم.
padding نه فقط برای دستیابی به طول ثابت، بلکه برای حفظ ساختار زمانی سیگنال صوتی در آرایه ویژگی استفاده می شود. در اینجا ما از N_FRAME_MFCC که به صورت تجربی برابر با 30 قرار داده ایم استفاده میکنیم تا تمام MFCC هایی خود را به صورتی یکدست تولید کنیم، به این منظور تابع زیر را تعریف میکنیم:

```
def pad_mfcc(raw_mfcc, n_frame_mfcc = N_FRAME_MFCC):
    temp = np.tile(raw_mfcc, (1 ,int(np.ceil(n_frame_mfcc / raw_mfcc.shape[1])) ))
    padded_mfcc_features= temp[:,:n_frame_mfcc]
    return padded_mfcc_features
```

2. در ادامه تابعی تعریف میکنیم که به ازای هر فایل رکورد، ابتدا با کمک کتابخانه librosa، mfcc فایل را به صورت خام استخراج میکند و سپس به کمک تابع pad_mfcc آن را به فرمت خواسته شده در می اورد.

```
def extract_mfcc(file_path, target_sampling_rate = TARGET_SAMPLING_RATE, n_mfcc = N_MFCC, n_frame_mfcc = N_FRAME_MFCC):
    audio, _ = librosa.load(file_path)
    raw_mfcc = librosa.feature.mfcc(y = audio, sr = target_sampling_rate, n_mfcc = n_mfcc)
    mfcc = pad_mfcc(raw_mfcc, n_frame_mfcc)
    return mfcc
```

3. در ادامه نیز یک تابع نوشته ایم که با ازای تمامی رکورد های موجود در یک فولدر mfcc ها را استخراج کند و آنها را در فولدر دیگری ذخیره کند.

```
def extract_features_for_all_files(recordings_folder, mfcc_folder):
    if not os.path.exists(mfcc_folder):
        os.makedirs(mfcc_folder)

    for file_name in os.listdir(recordings_folder):
        if file_name.endswith('.wav'):
            file_path = os.path.join(recordings_folder, file_name)
            mfcc_feature = extract_mfcc(file_path)
            # Save mfcc_feature as file
            output_file_path = os.path.join(mfcc_folder, file_name.replace('.wav', '.npy'))
            np.save(output_file_path, mfcc_feature)
            #print(f"MFCC features extracted and saved for {file_name} , shape = {mfcc_feature.shape}")
```

4. سپس با فراخوانی تابع بالا، mfcc ها را استخراج میکنیم.

```
extract_features_for_all_files(preprocessed_recordings_folder, mfcc_features_folder)
```

5. میتوانسیتم در همان تابع بالا در زمان ذخیره mfcc آنها در یک set نیز ذخیره کنیم، که در کل سرعت برنامه را شاید بیشتر میکرد، اما به دلیل تک کاربردی بودن هر تابع، این کار را در ادامه به صورت زیر با خواندن مجدد فایل هایی که تولید شده اند انجام میدهیم:

```
mfcc_files= [f for f in os.listdir(mfcc_features_folder) if f.endswith('.npy')]
mfcc_features = {file: np.load(os.path.join(mfcc_features_folder, file)) for file in mfcc_files}
```

## رسم Heat Map:

پس از استخراج ویژگی ها، نمودار Heat Map مربوط به به ضرایب برای یک نمونه از هرکدام از دسته ها را رسم میکنیم.

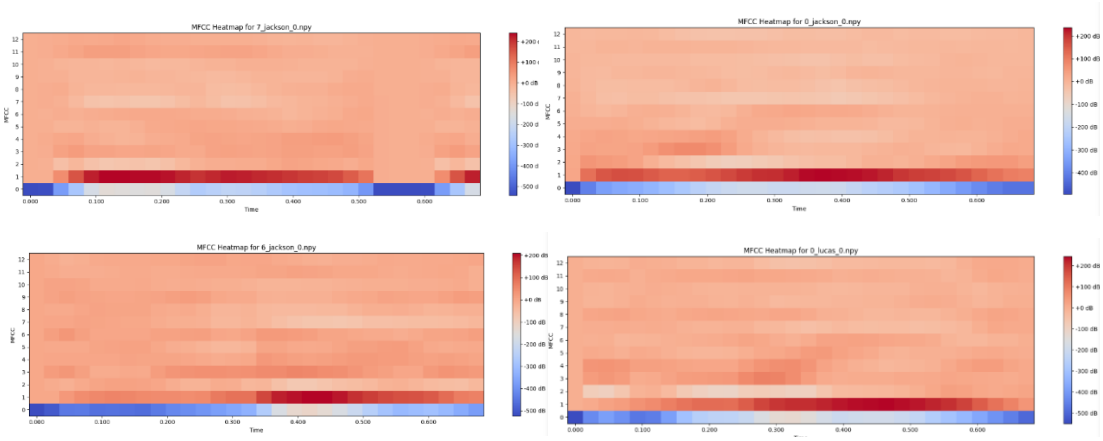
1. به منظور رسم نمودار های Heat Map، تابع زیر را تعریف کرده ایم که به کمک کتابخانه matplotlib میتوان Heat Map را برای تمامی mfcc ها در یک mfcc set رسم کرد:

```
def plot_mfcc_heatmaps(mfccs, n_mfcc = N_MFCC):
    for file_name, mfcc in mfccs.items():
        plt.figure(figsize=(15, 5))
        plt.title(f'MFCC Heatmap for {file_name}')
        librosa.display.specshow(mfcc, x_axis='time')
        plt.yticks(range(0, n_mfcc))
        plt.ylabel('MFCC')
        plt.colorbar(format='%+2.0f dB')
        plt.tight_layout()
        plt.show()
```

2. در ادامه برای آنکه تنها یک فایل به ازای هر رقم گفته شده توسط هر فرد را رسم کنیم، mfcc ها را براساس index دسته بندی میکنیم و فقط برای آنهایی که index آنها 0 هستند را رسم میکنیم.

```
files_with_index_0 = [f for f in os.listdir(mfcc_features_folder) if f.endswith('_0.npy')]
mfcc_features_index_0 = {file: np.load(os.path.join(mfcc_features_folder, file)) for file in files_with_index_0}
plot_mfcc_heatmaps(mfcc_features_index_0)
```

در ادامه برخی از نمودار ها را اورده ایم:




در ادامه برای آنکه بتوان تمامی نمودار ها را دریافت و ذخیره کرد تابع زیر را نوشته ایم، که به دلیل جلوگیری از کند شدن برنامه فعلا به اجرای آن نمیپردازیم:

```python
def save_plot_mfcc_heatmaps(mfccs, folder_name, n_mfcc = N_MFCC):
    if not os.path.exists(folder_name):
        os.makedirs(folder_name)
    for file_name, mfcc in mfccs.items():
        plt.figure(figsize=(15, 5))
        plt.title(f'MFCC Heatmap for {file_name}')
        librosa.display.specshow(mfcc, x_axis='time')
        plt.yticks(range(0, n_mfcc))
        plt.ylabel('MFCC')
        plt.colorbar(format='%+2.0f dB')
        plt.tight_layout()
        full_path = os.path.join(folder_name, file_name.replace('.npy', ''))
        plt.savefig(full_path)
        plt.close()
```

```python
#save_plot_mfcc_heatmaps(mfcc_features, heatmaps_folder)
```

**پرسش 3:**

Robustness و حساسیت ویژگی های MFCCs را نسبت به تغیرات در سیگنال های صوتی بررسی کنید:

پاسخ:

- Robustness (استحکام):

MFCCها در برابر نویز تقریبا مقاوم هستند، به خصوص درصورتی که قبل از استخراج MFCCs ها، در پیش پردازی به کاهش نویز بپردازیم، MFCC ها به خوبی عمل میکنند، اما به طور کلی برای تشخیص گفتار حتی در محیط های پر سرو صدا نیز مناسبند.

- حساسیت:
  - تغیرات زمانی:

MFCC ها نمیتوانند تغیرات سریع صدا (در واحد زمان) را به خوبی ثبت کنند و درنتیجه برای application هایی که نیاز به تجزیه تحیلی دقیق زمانی دارند، مناسب نیستند.

  - MFCC ها نسبت به بمی، لحن و ریتم حساس هستند که میتواند هم نقطه ضعف و هم قوت آنها محسوب شود، اما به طور کلی مزیتی که دارد این است میتواند بین دو صدا تفاوت قائل شود و در مواردی که هدف تشخیص گوینده است این ویژگی یکی از مزیت های آن به حساب می آید.

موارد دیگری نیز وجود دارد اما تا همین حد و نسبت به این پروژه میتوان گفت که، MFCCs ها با مقاومت در برابر نویز و حساسیت به حس و لحن گفتار میتوانند بسیار در تشخیص گفتار موثر عمل کنند.

**پرسش 4:**

آیا موارد خاصی وجود دارند که ضرایبMFCC کارایی کمتری داشته باشند؟

پاسخ:

بله بدون شک موقعیت هایی وجود دارد که MFCCs ها کمتر موثرند و یا کارایی دارند.

1. ملودی و هارمونی: MFCCها در capture کردن ویژگی های timbral بسیار عالی هستند اما مستقیماً برای استخراج ویژگی های موسیقی سطح بالا مانند ملودی، هارمونی یا ریتم مناسب نیستند. برای کارهایی که نیاز به تجزیه و تحلیل محتوای موسیقایی دقیق دارند، مانند رونویسی یا تشخیص ملودی، ویژگی‌ها یا تکنیک‌های دیگری اغلب همراه یا به جای MFCC مورد نیاز است.

2. تشخیص صدای محیطی: در حالی که MFCC ها تا حدی در برابر نویز مقاوم هستند، کارایی آنها می تواند در محیط های صوتی بسیار متغیر یا پیچیده کاهش یابد. تشخیص صدای محیطی، که اغلب شامل تجزیه و تحلیل صداها از چندین منبع با نویز پس‌زمینه قابل توجه است، ممکن است به ویژگی‌های اضافی یا جایگزین برای بهبود دقت نیاز داشته باشد.
3. MFCC :Fine-Grained Temporal Resolutionها ممکن است تغییرات زمانی بسیار سریع را به دلیل فرآیند windowing و میانگین‌گیری که در محاسبه آن‌ها دخیل هستند، به طور موثر ثبت نکنند. برای صداهایی با اطلاعات قابل توجه در حالت های گذرا یا نوسانات سریع، مانند صداهای ضربی خاص یا رویدادهای صوتی غیرگفتاری بسیار پویا، MFCC ممکن است اطلاعات حیاتی را از دست بدهد.

به طور خلاصه، در حالی که MFCC ها مجموعه ای از ویژگی های قدرتمند و همه کاره برای بسیاری از وظایف آنالیز صدا هستند، کاربرد و کارایی آنها می تواند در موقعیت هایی که نیاز به تجزیه و تحلیل دقیق گام، وضوح زمانی سریع، یا استخراج ویژگی های موسیقی سطح بالا دارند، محدود شود.

**پرسش 5:**

چرا در محاسبه MFCC فریم های استفاده شده با یکدیگر هم پوشانی دارند؟

پاسخ:

در فرآیند محاسبه MFCCs، فریم ها اغلب با یکدیگر همپوشانی دارند تا از انتقال هموار بین فریم ها اطمینان حاصل شود و از از دست رفتن اطلاعات در مرزها جلوگیری شود. این همپوشانی جنبه مهمی از تکنیک پردازش سیگنال است که برای تجزیه و تحلیل سیگنال‌های متغیر با زمان، مانند صدا، استفاده می‌شود. در اینجا دلایل کلیدی همپوشانی فریم ها در محاسبه MFCC آورده شده است:

1. **تداوم و همواری (اجتناب از ناپیوستگی ها)**
فریم های همپوشانی به ایجاد یک انتقال هموار بین فریم های مجاور کمک می کند. بدون همپوشانی، شروع و پایان ناگهانی فریم ها می تواند ناپیوستگی های مصنوعی در سیگنال ایجاد کند، که به طور بالقوه بر تحلیل طیفی تأثیر می گذارد و منجر به عدم دقت در ویژگی های استخراج شده می شود.
2. **وضوح زمانی:**
با همپوشانی فریم ها، با آنالیز وضوح زمانی دقیق تری به دست می آوریم. این امر به ویژه برای ثبت ویژگی‌ها و تفاوت‌های گذرا در گفتار و موسیقی که ممکن است بین فریم‌های on-overlapping رخ دهد، مهم است. همپوشانی تضمین می کند که این ویژگی های گذرا نادیده گرفته نمی شوند و به اندازه کافی در تحلیل نشان داده می شوند.
3. **جلوگیری از از دست دادن اطلاعات:**
سیگنال های صوتی می توانند حاوی اطلاعات مهمی در لبه های فریم باشند. بدون همپوشانی ممکن است این اطلاعات را از دست بدهیم. فریم های همپوشانی تضمین می کنند که اطلاعات در لبه ها به طور کامل در فریم های متعدد مورد استفاده قرار می گیرد و خطر از دست دادن ویژگی های سیگنال بحرانی را کاهش می دهد.

به طور خلاصه، همپوشانی فریم ها در محاسبه MFCC ها یک انتخاب عمدی برای بهبود کیفیت و قابلیت اطمینان فرآیند استخراج ویژگی است. این کمک می کند تا اطمینان حاصل شود که تجزیه و تحلیل تفاوت های ظریف سیگنال صوتی را به طور موثر ضبط می کند.

**پرسش 6:**

چرا در اکثر پروژه های مرتبط با صوت تنها از ۱۲ یا ۱۳ ضریب ابتدایی MFCC استفاده میشود؟

پاسخ:

1. **کاهش بازده در ضرایب بالاتر**

MFCC های lower-order، مهم ترین ویژگی های طیفی سیگنال صوتی را که مربوط به شکل مجرای صوتی است، می گیرند. این ویژگی ها برای تمایز بین واج(phoneme) های مختلف در گفتار بسیار مهم هستند. ضرایب مرتبه بالاتر، از سوی

دیگر، تمایل به نمایش جزئیات دقیق تری در طیف دارند، که اغلب شامل نویز و اطلاعات بحرانی کمتر برای بسیاری از برنامه ها می شود. سهم این ضرایب مرتبه بالاتر در بهبود عملکرد در کارهایی مانند تشخیص گفتار یا شناسایی گوینده اغلب افزایش پیچیدگی محاسباتی را توجیه نمی کند.

2. **کاهش پیچیدگی محاسباتی:**
   محدود کردن تعداد ضرایب مورد استفاده، پیچیدگی محاسباتی مراحل پردازش بعدی، مانند machine Train learning را کاهش میدهد.
3. **مقاومت در برابر نویز:**
   MFCC های مرتبه بالاتر بیشتر مستعد نویز هستند زیرا اجزای نویز در سیگنال های صوتی می توانند محتوای فرکانس بالاتری داشته باشند، که ضرایب مرتبه بالاتر به احتمال زیاد نشان دهنده آن هستند. با تمرکز بر 12 یا 13 ضریب اول، سیستم ها می توانند در برابر نویز پس زمینه و سایر اشکال تداخل قوی تر باشند.

به این دلایل و بسیاری از دلایل دیگر، انتخاب 12 یا 13 MFCC اول یک روش معمول در پروژه های پردازش صدا است که تعادلی بین ثبت ویژگی های اساسی سیگنال صوتی و حفظ نیازهای محاسباتی قابل مدیریت ایجاد می کند.

## بخش دوم: آشنایی با HMM:

### پرسش 1:

توضیح دهید منظور از State ها و Observation چیست؟

پاسخ:

1. **State:**

در یک HMM، یک حالت یک نمایش از شرایط یا فازی است که در مدل قرار دارد ولی قابل مشاهده نیست، مدل بین این حالت ها در طول زمان، تغییر میکند، هر حالت با یک توزیع احتمال همراه است که احتمال مشاهدات مختلف ناشی از آن حالت را مدل میکند.

2. **Observation:(مشاهدات)**

مشاهدات، نقاط داده ای هستند که ما در واقع اندازه گیری می کنیم یا می بینیم. در زمینه HMM ها، مشاهدات توسط حالت ها بر اساس توزیع های احتمال مرتبط با آن حالت ها تولید می شوند. با این حال، به دلیل ماهیت احتمالی مدل، نمی‌توانیم مستقیماً وضعیت را از مشاهده استنتاج کنیم. در عوض، ما از الگوریتم‌هایی مانند الگوریتم Viterbi برای استنتاج محتمل‌ترین توالی حالت‌ها با توجه به دنباله‌ای از مشاهدات استفاده می‌کنیم.

در این تمرین State ها کدامند و Observation چگونه بدست می‌آید؟

پاسخ:

1. **State:**

برای هر یک موارد تشخیص گوینده/عدد، حالت ها میتوانند متفاوت باشند. در زمینه تشخیص اعداد، هر حالت می تواند بخشی از گفتار مربوط به بیان یک عدد خاص یا یک واج را در آن عدد نشان دهد. برای تشخیص گوینده، حالت ها ممکن است نشان دهنده ویژگی های آوایی یا خاص گوینده باشد که یک گوینده را از دیگری متمایز می کند.

ما در اینجا تعداد حالت را برای هردو یکسان در نظر گرفته ایم، اما میتوانست متغیر باشد.

2. **Observation:(مشاهدات)**

مشاهدات MFCCهای استخراج شده از ضبط های گفتار خواهند بود. MFCC ها نمایشی از طیف توان short-term صدا هستند که بر اساس تبدیل کسینوس خطی طیف توان لگاریتم بر روی یک مقیاس فرکانس mel غیر خطی است. آنها در گرفتن جنبه های صدای صدا موثر هستند، به همین دلیل است که آنها به طور گسترده در وظایف تشخیص گفتار و گوینده استفاده می شوند.

نحوه بدست اوردن مشاهدات:

1. پیش پردازش
2. استخراج MFCCs به کمک کتاب خانه

این مراحل به صورت دقیق قبلتر بررسی شده اند.

**پرسش 2:**

مدل های HMM را میتوان بر اساس میزان وابستگی میان حالت های پنهان دسته بندی کرد، مدلی که در این تمرین به پیاده سازی آن می پردازید یک مدل First-Order HMM است. دلیل نامگذاری آن و همچنین ویژگی های آن را بررسی کنید و تفاوت آن با مدل های دیگر در این دسته بندی را بیان کنید.

پاسخ:

HMMها را می‌توانیم بر اساس وابستگی‌های بین حالت های پنهان آن‌ها طبقه‌بندی کرد، که HMM مرتبه اول یکی از رایج‌ترین انواع آن است. این طبقه بندی تا حد زیادی مشخص می کند که چگونه مدل بر اساس وضعیت فعلی خود، وضعیت های آینده را پیش بینی می کند و بر پیچیدگی و منابع محاسباتی مورد نیاز برای آموزش و استنتاج تأثیر می گذارد.

یک HMM مرتبه اول فرض می کند که احتمال انتقال به حالتبعدی فقط به حالت الان بستگی دارد. این به عنوان ویژگی مارکوف(Markov property) شناخته می شود. در یک HMM مرتبه اول، حالتهای اینده به شرط دانستن حالت حال از حالت های گذشته مستقل میشوند.

HMM مرتبه اول با موارد زیر مشخص می شود:

- Transition probabilities بین حالت ها نشان دهنده احتمال انتقال از یک حالت به حالت دیگر است.

- Emission probabilities که نشان دهنده احتمال راه اندازی سیستم در هر حالت معین است.

مدل های دیگر HMM ها وابستگی های متفاوتی را بین حالت ها و مشاهدات معرفی می کنند. تفاوت های اصلی بین HMM مرتبه اول و انواع دیگر عبارتند از:

- HMM های مرتبه دوم: این مدل ها فرض می کنند که احتمال انتقال به حالت بعدی به حالت فعلی و قبل از آن بستگی دارد. و میتواند این به مرتبه‌های بالاتر گسترش می‌یابد که در آن وابستگی می‌تواند به چندین حالت برگردد. تفاوت اصلی در حافظه(memory) مدل است. HMM های مرتبه اول فقط وضعیت فعلی را "به یاد می آورند"، در حالی که HMM های مرتبه بالاتر بیشتر گذشته را به یاد می آورند، و آنها را به طور بالقوه دقیق تر می کند، اما همچنین پیچیده تر و با منابع فشرده تر هستند.
- Coupled HMMs and Factorial HMMs:
این مدل‌ها شامل زنجیره‌های چندگانه وابسته به هم از حالت‌های پنهان هستند، جایی که انتقال یک حالت در یک زنجیره ممکن است به حالت‌های زنجیره‌های دیگر بستگی داشته باشد.
- آنها به طور قابل توجهی پیچیده تر از HMM های مرتبه اول هستند، که امکان مدل سازی وابستگی های پیچیده تر را فراهم می کند اما این موضوع همراه با افزایش پیچیدگی محاسباتی و دشواری در تخمین پارامتر است.
- semi-Markov HMMs:
در این مدل‌ها، مدت‌زمان‌های حالت (یعنی زمان صرف شده در هر حالت) به صراحت مدل‌سازی می‌شوند، برخلاف HMMهای مرتبه اول که مدت زمان به طور ضمنی از یک توزیع geometric پیروی می‌کند.
HSMM ها می توانند سیستم هایی را مدل کنند که در آن مدت حالت ها مهم است و به طور قابل توجهی متفاوت است و انعطاف پذیری بیشتری نسبت به HMM های مرتبه اول در نمایش مدت زمان حالت ارائه می دهد.

## پرسش 3:

درباره HMMتحقیق کنید و توضیح دهید که که این مدل برای بررسی و تحلیل چه پدیده هایی مناسب است؟ چرایی این موضوع را توضیح دهید.

پاسخ:

HMMها ابزارهای همه کاره برای مدل‌سازی داده‌های متوالی هستند، به‌ویژه زمانی که سیستم در حال مدل‌سازی دارای حالت‌های اساسی است که مستقیماً قابل مشاهده نیستند. قدرت آنها در توانایی آنها برای مدل سازی انتقال احتمالی بین حالت های پنهان و تولید داده های قابل مشاهده از این حالت های نهفته است. این امر HMM ها را برای طیف گسترده ای از مشکلات در دامنه های مختلف مناسب می کند:

1. **تشخیص گفتار**

HMM ها از نظر تاریخی در تشخیص گفتار مهم هستند، جایی که هر حالت می تواند یک واج یا بخشی از یک واج را نشان دهد و مشاهدات سیگنال های صوتی هستند. این مدل انتقال‌های زمانی بین صداهای مختلف در گفتار را ضبط می‌کند و تشخیص کلمات یا عبارات گفتاری را از داده‌های صوتی خام ممکن می‌سازد.

2. **پردازش زبان طبیعی (NLP)**

در NLP، HMM ها برای کارهایی مانند برچسب گذاری قسمتی از گفتار استفاده شده است، که در آن هر کلمه در یک جمله با بخش متناظر گفتار خود (اسم، فعل، صفت و غیره) برچسب گذاری می شود و حالت ها قسمت هایی از گفتار هستند.

آنها همچنین در تشخیص نهادهای نامگذاری شده برای شناسایی و طبقه بندی عناصر کلیدی در متن به دسته های از پیش تعریف شده مانند نام افراد، سازمان ها، مکان ها، بیان زمان ها، مقادیر، مقادیر پولی، درصد و غیره استفاده می شوند.

3. **بیوانفورماتیک(Bioinformatics)**

در بیوانفورماتیک، HMM ها برای مدل سازی توالی های بیولوژیکی استفاده می شوند. برای مثال، آنها در پیش‌بینی ژن برای شناسایی مناطق از DNA ژنومی که ژن‌ها را رمزگذاری می‌کنند، استفاده می‌شوند. آنها همچنین می‌توانند برای پیش‌بینی ساختار پروتئین، تراز کردن توالی‌ها و شناسایی ژن‌های همولوگ استفاده شوند.

4. **امور مالی و اقتصادی**

HMM ها می توانند وضعیت های اقتصادی پنهان مانند سازمان های بازار (مثلا بازارهای نزولی) را از داده های قابل مشاهده مانند قیمت سهام یا حجم معاملات مدل کنند.

5. **رباتیک**

HMM ها در رباتیک برای کارهایی مانند برنامه ریزی مسیر و نقشه برداری محیط، کاربرد پیدا می کنند، جایی که ربات باید موقعیت خود (حالت پنهان) را بر اساس مشاهدات سنسورها یا دوربین ها در یک محیط نامشخص استنتاج کند.

6. **مدلسازی محیطی**

آنها در مدل‌سازی سیستم‌های محیطی، مانند پیش‌بینی الگوهای بارش یا دما، که در آن حالت‌های پنهان می‌توانند شرایط اقلیمی مختلفی را نشان دهند که بر داده‌های آب‌وهوای مشاهده‌شده تأثیر می‌گذارند، استفاده می‌شوند.

مناسب بودن HMM ها برای این مشکلات ناشی از انعطاف پذیری آنها در مدل سازی فرآیندهای تصادفی حاکم بر داده های متوالی، عدم قطعیت مرتبط با حالت های پنهان، و انتقال بین این حالت ها است. با این حال، اثربخشی HMM ها در این کاربردها به انتخاب مناسب پارامترهای مدل، کیفیت داده های آموزشی و پیچیدگی مشکل مدل سازی شده بستگی دارد. از آنجایی که تکنیک‌های جدیدتر یادگیری ماشین، مانند یادگیری عمیق، برجسته‌تر شده‌اند، گاهی اوقات برای کاربردهای خاص با HMM ترکیب می‌شوند یا به جای آن‌ها استفاده می‌شوند، به‌ویژه هنگام مدیریت مجموعه‌های داده بسیار بزرگ یا گرفتن الگوهای پیچیده که فراتر از وابستگی‌های مرتبه اول هستند.

**پرسش 4:**

مدل HMM نیز مانند هر مدل دیگری دارای مزایا و معایبی است که آن را ویژه می کند. مزایا و معایب این مدل را بررسی کرده و هر کدام را مختصرا توضیح دهید.

پاسخ:

## ❖ مزایای HMM

1. **قابلیت مدل‌سازی داده‌های متوالی زمانی:**

HMM ها برای مدیریت داده‌های متوالی طراحی شده‌اند، و آنها را به ویژه برای تجزیه و تحلیل سری‌های زمانی، گفتار، زبان و توالی‌های بیولوژیکی مناسب می‌سازد، جایی که جنبه زمانی بسیار مهم است.

2. **مدیریت وضعیت های پنهان:**

HMM قادر به مدل سازی سیستم هایی با حالت های غیرقابل مشاهده یا پنهان با استفاده از داده های قابل مشاهده هستند. این در بسیاری از سناریوهای دنیای واقعی که فرآیندی که به آن علاقه داریم مستقیماً قابل اندازه گیری نیست مفید است..

3. **نعطاف پذیری در فرآیندهای مدل سازی:**

HMM ها را می توان برای طیف وسیعی از کاربردها، از تشخیص گفتار گرفته تا بیوانفورماتیک، با تنظیم ساختار و پارامترهای آنها، استفاده شوند.

4. **تکنیک های استنتاج ب توسعه یافته:**

وجود داشتن الگوریتم های قوی ای مانند: الگوریتم هایی برای یادگیری پارامترهای HMM (مانند الگوریتم Baum-Welch)، رمزگشایی محتمل ترین توالی حالت های پنهان مشاهدات داده شده (مانند الگوریتم Viterbi) و محاسبه احتمال مشاهدات (به عنوان مثال، الگوریتم Forward-Backward).

## ❖ معایب HMM ها

1. **فرض Markov Property:**

این فرض که حالت آینده فقط به وضعیت فعلی بستگی دارد و نه به حالات گذشته (مخصوصا در مدل درجه اول) می تواند برای سیستم هایی که تاریخ یا وابستگی های طولانی تر اهمیت دارند، بسیار ساده انگارانه باشد. در حالی که HMM های مرتبه بالاتر وجود دارند، پیچیدگی مدل و الزامات محاسباتی را به طور قابل توجهی افزایش می دهند.

2. **فرض های ایستایی و همگنی:**

HMM ها احتمال های انتقال و انتشار ثابت را فرض می کنند، که ممکن است برای فرآیندهایی که این احتمالات در طول زمان یا در پاسخ به شرایط خارجی تغییر می کنند، واقعی نباشد.

3. **مقیاس پذیری و پیچیدگی محاسباتی:**

با افزایش تعداد حالت ها، پیچیدگی محاسباتی آموزش و رمزگشایی با HMM ها می تواند به طور قابل توجهی افزایش یابد و آنها را برای کاربردهای بسیار بزرگ یا برای پردازش بلادرنگ بدون محاسبات قابل توجه کمتر عملی کند.

4. **مشکل در مدل‌سازی وابستگی‌های Long-Range:**

HMMها، به‌ویژه مدل مرتبه اول، در گرفتن وابستگی‌های دوربرد در داده‌ها مشکل دارند، محدودیتی که می‌تواند بر عملکرد در کارهایی مانند مدل‌سازی زبان یا پیش‌بینی سری‌های زمانی پیچیده تأثیر بگذارد.

5. **چالش های تخمین پارامتر:**

فرآیند تخمین انتقال، انتشار، و احتمالات حالت اولیه (به ویژه در سناریوهای بدون نظارت که توالی وضعیت ناشناخته است) می تواند چالش برانگیز باشد.

اما علی‌رغم این محدودیت‌ها، HMMها با موفقیت در زمینه‌های متعددی به کار گرفته شده‌اند و همچنان ابزاری ارزشمند در جعبه ابزار دانشمندان داده هستند.

**پرسش 5:**

انواع مختلفی از مدل های HMM وجود دارد، درباره آنها تحقیق کنید و چند مورد را بطور مختصر بررسی کنید.

پاسخ:

انواع مختلف HMM ها برای کاربردهای خاص، از تشخیص گفتار تا تجزیه و تحلیل توالی بیولوژیکی، طراحی شده اند. در واقع این این تنوع به HMM ها اجازه می دهد تا برای طیف گسترده ای از کاربردها، از تشخیص گفتار و پردازش زبان طبیعی گرفته تا بیوانفورماتیک و فراتر از آن، طراحی شوند. در زیر برخی از انواع مختلف HMM وجود دارد که هر کدام برای چالش‌های مدل‌سازی خاص سازگار شده‌اند:

1. Discrete HMMs (HMM های گسسته):

HMM های گسسته دارای مجموعه محدودی از حالات قابل مشاهده هستند و مشاهدات از یک مجموعه گسسته گرفته می شوند. آنها به ویژه برای کاربردهایی مناسب هستند که مشاهدات را می توان به طور طبیعی به تعداد محدودی از نمادها طبقه بندی کرد، مانند برچسب گذاری قسمتی از گفتار در NLP، که در آن هر کلمه با یکی از مجموعه محدودی از دسته های دستوری برچسب گذاری می شود.

2. Continuous HMMs

بر خلاف HMM های گسسته، HMM های پیوسته برای رسیدگی به مشاهدات با ارزش واقعی طراحی شده اند. آنها معمولاً فرض می‌کنند که مشاهدات برای هر حالت از توزیع پیوسته خاصی پیروی میکنند. HMM های پیوسته به طور گسترده در تشخیص گفتار استفاده می شوند.

3. Hidden Semi-Markov Models (HSMMs)

در حالی که HMMهای استاندارد فرض می‌کنند که انتقال بین حالت‌ها در هر مرحله زمانی اتفاق می‌افتد، HSMM به حالت‌ها اجازه می‌دهد که مدت‌زمان مدل‌سازی صریح داشته باشند. این بدان معناست که مدل می‌تواند برای تعداد متغیری از مراحل زمانی در همان حالت باقی بماند، که در برنامه‌هایی مانند تشخیص حرکت، که در آن اقدامات دارای طول‌های متغیر هستند، مفید است.

4. Hierarchical HMMs(HMMهای سلسله مراتبی):

HMM های سلسله مراتبی لایه های متعددی از HMM ها را در خود جای می دهند، که در آن حالت های یک HMM سطح بالاتر، خود HMM های سطح پایین تر هستند. این ساختار امکان مدل‌سازی فرآیندهای پیچیده با سطوح مختلف انتزاع را فراهم می‌کنند.

5. Factorial HMMs
6. Coupled HMMs
7. Autoregressive HMMs
8. Input-Output HMMs (IOHMMs)

## اماده سازی داده ها:

در ابتدا برای ورود به فاز modeling و پیاده سازی، لازم است، داد ها را به دو دسته برای یادگیری (Train) و آزمایش (Test) تقسیم کنیم.

یک تابع مینویسم تا داده ها را به دو گروه برای Trainکردن یا همان فاز یادگیری و برای Test تقسیم کنیم:

```
def prepare_data(mfccs, num_repeated_recoding = NUM_REPEATED_RECORDING):
    train_features, test_features = [], []
    train_labels, test_labels = [], []

    split_index = num_repeated_recoding * TRAIN_PERCENT
    for file_name, mfcc in mfccs.items():
        digit_label, speaker_name, index_str =  file_name[:-4].split('_')
        index = int(index_str)
        if index < split_index:
            train_features.append(mfcc.T)
            train_labels.append(file_name)
        else:
            test_features.append(mfcc.T)
            test_labels.append(file_name)

    return train_features, train_labels, test_features, test_labels
```

```
train_features, train_labels, test_features, test_labels = prepare_data(mfcc_features, NUM_REPEATED_RECORDING)
```

```
len(train_labels)
```

```
900
```

```
len(test_labels)
```

```
2100
```

## بخش سوم: پیاده سازی مسئله:

### توابع اولیه و  مرتب سازی داده ها:

در این بخش به ارزیابی و تحلیل داده ها نمیپردازیم ولی تنها جهت مطمئن شدن از درستی عملکرد پیاده سازی‌ها، یک تابع ساده تعریف میکنیم که خروجی پیاده سازی و خروجی مورد نظر را پرینت کند:

```
def print_detailed_report(arr1, arr2):
    t = 0
    f = 0
    for i in range(len(arr1)):
        if arr1[i] != arr2[i]:
            f = f + 1
        else:
            t = t + 1
    print(f"num_true = {t}, num_false = {f}")
    for i in range(len(arr1)):
        print(i)
        if arr1[i] != arr2[i]:
            print(Fore.RED + f"\033[1m{arr1[i], arr2[i]}\033[0m"+ Style.RESET_ALL)
        else:
            print(Fore.BLUE  + f"{arr1[i], arr2[i]}"+ Style.RESET_ALL)
```

در ادامه دو تابع تعریف میکنیم تا براساس label داده‌ها مقدار واقعی digit یا speaker را  استخراج کند:

```
def extract_digit(label):
    digit_label = label.split('_')[0]
    return digit_label

def extract_speaker(label):
    speaker_label = label.split('_')[1]
    return speaker_label
```

در ادامه برای مرتب سازی داده های  Trainو دسترسی راحت تر هم به MFCC ها  و  هم  Label مورد نظر ( digit یاspeaker) آنها را باهم Zip کرده و داده ها را براساس  Labelمورد نظر دسته بندی میکنیم که حکم همان "قطعه بندی" را دارد که ما به اینجا موکول کرده بودیمش.

```
# Organize training data by digit
training_data_by_digit = defaultdict(list)
for mfcc, label in zip(train_features, train_labels):
    digit = extract_digit(label)
    training_data_by_digit[digit].append(mfcc)

# Organize training data by speaker
training_data_by_speaker = defaultdict(list)
for mfcc, label in zip(train_features, train_labels):
    speaker = extract_speaker(label)
    training_data_by_speaker[speaker].append(mfcc)
```

لازم است برای بررسی های این قسمت و هم قسمت های بعد مقدار واقعی داده ها را هم برای digit و  هم برای speaker داشته باشیم:

```
test_true_digits = [extract_digit(label) for label in test_labels]
test_true_speakers = [extract_speaker(label) for label in test_labels]
```

## پارت اول: پیاده سازی به کمک کتابخانه ها:

برای پیاده سازی مدل و تست آن باید دو کار انجام دهیم 1. یادگیری و ایجاد مدل hmm برای هر مقدار/گوینده 2.تست هر یک mfcc ها که observation های ما هستند و نسبت دادن یک مقدار(امتیاز) به آنها به ازای هر مدل و در نهایت انتخاب بهترین جواب.

1. یادگیری

در این قسمت کافی است با استفاده از توابع GuassianHMM وfit از کتابخانه hmmlearn به یادگیری بپردازیم

2. تست:

در این بخش به ازای تمام observation ها یا همان MFCC ها امتیاز به دست امده از هر مدل را محسابه میکنیم و براساس مدلی که بیشترین امتیاز داشت، مقدار پیشبینی شده را انتخاب میکنیم.

### برای اعداد:

```
hmm_models_digit_1 = {}

for digit, data in training_data_by_digit.items():
    lengths = [len(sequence) for sequence in data]
    X = np.concatenate(data)
    model = hmm.GaussianHMM(n_components = NUM_STATE)
    model.fit(X, lengths)
    hmm_models_digit_1[digit] = model
```

```
test_predictions_digits_1 = []

for mfcc in test_features:
    best_score, best_digit = float("-inf"), None
    for digit, model in hmm_models_digit_1.items():
        score = model.score(mfcc)
        if score > best_score:
            best_score, best_digit = score, digit
    test_predictions_digits_1.append(best_digit)
```

```
print_detailed_report(test_true_digits, test_predictions_digits_1)
```

### برای گوینده:

```
hmm_models_speaker_1 = {}

for speaker, data in training_data_by_speaker.items():
    lengths = [len(sequence) for sequence in data]
    X = np.concatenate(data)
    model = hmm.GaussianHMM(n_components = NUM_STATE)
    model.fit(X, lengths)
    hmm_models_speaker_1[speaker] = model
```

```
test_predictions_speakers_1 = []

for mfcc in test_features:
    best_score, best_speaker = float("-inf"), None
    for speaker, model in hmm_models_speaker_1.items():
        score = model.score(mfcc)
        if score > best_score:
            best_score, best_speaker = score, speaker
    test_predictions_speakers_1.append(best_speaker)
```

```
print_detailed_report(test_true_speakers, test_predictions_speakers_1)
```

## پارت دوم: پیاده سازی به کمک Scratch:

ابتدا کلاس HMM را تکمیل میکنیم که به علت ظولانی بودن، آن را در گزارش نیاورده ام.

سپس همانند پارت قبل دو بخش را پیاده سازی میکنیم:

1. یادگیری
2. تست

### برای اعداد:

```
hmm_models_digit_2 = {}

for digit, data in training_data_by_digit.items():
    model = HMM(num_hidden_states = NUM_STATE)
    X = np.concatenate(data)
    model.train(X.T, NUM_ITERATION)
    hmm_models_digit_2[digit] = model
```

```
test_predictions_digits_2 = []

for mfcc in test_features:
    best_score, best_digit = float("-inf"), None
    for digit, model in hmm_models_digit_2.items():
        score = model.score(mfcc.T)
        if score > best_score:
            best_score, best_digit = score, digit
    test_predictions_digits_2.append(best_digit)
```

```
print_detailed_report(test_true_digits, test_predictions_digits_2)
```

### برای گوینده:

```
hmm_models_speaker_2 = {}

for speaker, data in training_data_by_speaker.items():
    model = HMM(num_hidden_states = NUM_STATE)
    X = np.concatenate(data)
    model.train(X.T, NUM_ITERATION)
    hmm_models_speaker_2[speaker] = model
```

```
test_predictions_speakers_2 = []

for mfcc in test_features:
    best_score, best_speker = float("-inf"), None
    for speaker, model in hmm_models_speaker_2.items():
        score = model.score(mfcc.T)
        if score > best_score:
            best_score, best_speker = score, speaker
    test_predictions_speakers_2.append(best_speker)
```

```
print_detailed_report(test_true_speakers, test_predictions_speakers_2)
```

**پرسش بخش سه:**

نکته: ممکن است نتایج شما در بخش اول و دوم فرق کند و مدل آماده (که در بخش اول از آن استفاده کردید) نتایج متفاوت و دقت بالاتری نسبت به مدل طراحی شده توسط شما داشته باشد. این اختلاف ممکن است چه دلایلی داشته باشد؟ درباره عوامل تاثیرگذار بر روی این اختلاف دقت تحقیق کنید.

راهنمایی: این تفاوت می تواند در ساختار مدل ها یا پیش پردازش داده ها باشد.

پاسخ:

همانظور که انتظار میرفت با توجه به اینکه هم برای اعداد و هم برای گویندگان، تعداد خطا در مدل اول (به کمک کتابخانه) کمتر است، میتوان گفت این مدل، دقت بیشتری دارد که میتواند دلایل مختلفی داشته باشد:

1. Optimized numerical computations:

"hmmlearn" از کتابخانه های محاسباتی عددی بهینه شده مانند NumPy و SciPy استفاده می کند. این کتابخانه ها برای عملیات عددی بسیار بهینه و کارآمد هستند، که می تواند به نتایج دقیق تری منجر شود، به ویژه در سناریوهایی که شامل عملیات ماتریس پیچیده و محاسبات احتمال است.

2.

پیاده سازی الگوریتم ها برای HMM ها شامل عملیات هایی با احتمالات و ماتریس است که در صورت عدم مدیریت صحیح می تواند از نظر عددی ناپایدار شوند. کتابخانه‌هایی مانند «hmmlearn» معمولاً با استفاده از تکنیک‌هایی مانند log transformations یا scaling factors ، مسائل مربوط به ثبات عددی، مانند underflow و overflow را مدیریت می‌کنند.

3. Fine-tuning and parameter optimization:

در "hmmlearn" ممکن است پارامترهای مختلف برای الگوریتم های زیربنایی را جهت رسیدن به عملکرد بهتر در طیف گسترده ای از مجموعه داده ها و برنامه های کاربردی دقیق تر تنظیم شده باشد

4. تست فراوان جهت صحت کد:

کتابخانه های اماده مانند "hmmlearn" به طور گسترده توسط developer ها مورد آزمایش و تایید قرار گرفته اند. در مقایسه با یک پیاده سازی ساده که ما داشته ایم ، که به اندازه کافی تست نشده است، احتمال کمتری وجود دارد که دارای اشکالات یا خطاهای پیاده سازی باشند.

5. Handling of edge cases and corner scenarios:

و بسیاری موارد دیگر که ممکن است در کلاس ساده شده ما، رعایت نشده باشد.

*

## بخش چهارم: ارزیابی و تحلیل

معیارهای زیادی برای سنجش و ارزیابی عملکرد مدل ها وجود دارد؛ مثل: Accuracy Score، Precision، Recall، F1

قبل از پاسخ به پرسش های زیر لازم است با چند اصطلاح اشنا شویم:

- correctly predicted positives = True Positives = TP
- incorrectly predicted positives = False Positives = FP
- correctly predicted negatives = True Negatives = TN
- incorrectly predicted negatives = False Negatives= FN

در ادامه دو مورد زیر را نیز تعریف میکنیم که به ترتیب تعداد پیشبینی های درست و غلط در کل هستند.

- T = TN + TP
- F = FN + FP

**پرسش 1:**

درباره هر کدام از معیارهای بالا تحقیق کنید و نحوه محاسبه هر یک را توضیح دهید.

پاسخ:

- **Accuracy:**

این معیار ساده ترین و شهودی ترین معیار عملکرد است. نسبت نتایج واقعی (هم مثبت و هم منفی واقعی) را در بین تعداد کل موارد بررسی شده اندازه گیری می کند.

برای ارزیابی کلی بهترین است اما در مجموعه داده های نامتعادل می تواند فریبنده باشد.

$$Accracy = \frac{T}{F+T} = \frac{TP+TN}{\text{TN} + \text{TP} + \text{FN} + \text{FP}}$$

- **Precision:**

در این معیار تنها به پیشبینی های مثبت (Positive) توجه میشود و تعداد آنها در تعداد کل موارد مثبت واقعی بررسی میکنیم.

استفاده از این معیار وقتی هزینه مثبت کاذب زیاد است پیشنهاد میشود (به عنوان مثال، ارسال ایمیل های تبلیغاتی به کاربران غیر علاقه مند)

$$Precision = \frac{TP}{TP+FP}$$

- **Recall (Sensitivity or True Positive Rate):**

این معیار در میان تمام مواردی که مثبت در نظر گرفته شده اند، درصدی که به درستی در نظر گرفته شده اند را بدست می اورد. یا به عبارتی ه این سوال پاسخ می دهد: "از بین تمام موارد مثبت واقعی، چند مورد را به درستی پیش بینی کردیم؟"

این معیار زمانی که هزینه منفی کاذب بالا باشد بسیار مهم است. (به عنوان مثال، عدم تشخیص یک بیماری جدی)

$$Recall = \frac{TP}{TP+FN}$$

- **F1 Score:**

در این معیار میانگین هارمونیک Precision و Recall محاسبه میشود.

زمانی این معیار برای ما اهمیت ویژه دارد که توزیع ناهمواری داشته باشیم. در واقع این معیار تلاش دارد تعادلی میان دو معیار قبلتر ایجاد کند.

$$F1\ score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

**پرسش 2:**

آیا محاسبه معیارهای ذکر شده برای این پروژه که multi-class(چند کلاسه) است، چالشی دارد؟ اگر بله چه راه حلی برایش دارید؟

پاسخ:

بله بی شک مشکلات و پیچیدگی هایی به همراه داشت چرا که حتی فرمول هایی که قبلتر تعریف کرده ایم هم برای کلاس های binary است که در ادامه به برخی از این موارد میپردازیم.

- وجود نداشتن تعادل در بین class ها
- ابهام در TN و FP:

در یک طبقه بندی باینری، TN و FP واضح هستند زیرا فقط یک کلاس دیگر وجود دارد. با این حال، در تنظیمات چند کلاسه، وقتی یک کلاس را در نظر می گیریم، بقیه کلاس ها فقط یک کلاس "منفی" نیستند بلکه چندین کلاس مجزا هستند. این باعث می شود محاسبه معیارها پیچیده تر شود.

برخی از راه حل های ممکن:

- Macro-Averaging: (از این روش استفاده کرده ایم)

متریک مورد نظرمان را را به طور مستقل برای هر کلاس محاسبه میکنیم و سپس میانگین را میگیریم. این روش بدون در نظر گرفتن فراوانی آنها در مجموعه داده، با تمام کلاس ها به طور یکسان رفتار می کند.

N را با تعداد کلاس ها درظر میگیریم:

$$\boldsymbol{Metric_{macro} = \frac{1}{N}\sum_{i=1}^{N} Metric_{class\ i}}$$

برخی از دیگر روش ها و راه حل های ممکن:

- Micro-Averaging
- Weighted-Averaging
- One-vs-All (OvA) or One-vs-Rest (OvR) Strategy

**پرسش 3:**

توضیح دهید که هر کدام از معیارها چگونه مدل را ارزیابی میکنند.

پاسخ:

- **Accuracy:**
  - چه چیزی را اندازی گیری میکند:

صحت کلی مدل را اندازه گیری می کند، به عنوان مثال، هر چند وقت یکبار مدل خروجی صحیح را پیش بینی می کند، خواه یک کلاس مثبت یا منفی باشد.

  - جنبه ارزیابی:

اثربخشی کلی مدل را در تمام پیش بینی ها ارزیابی می کند.

- **Precision:**

- چه چیزی را اندازی گیری میکند:
توانایی مدل را برای پیش بینی صحیح نتایج مثبت از همه نتایج مثبت پیش بینی شده اندازه گیری می کند. این نسبت پیش بینی های مثبت واقعی در کل پیش بینی های مثبت است.
- جنبه ارزیابی:
مدل را در پیش بینی موارد مثبت ارزیابی می کند. دقت بالا نشان دهنده نرخ پایین مثبت کاذب است.

- **Recall (Sensitivity or True Positive Rate):**
- چه چیزی را اندازی گیری میکند:
توانایی مدل را برای شناسایی صحیح نکات مثبت واقعی از همه مثبت های واقعی اندازه گیری می کند. این نسبت پیش بینی های مثبت واقعی در تمام موارد مثبت واقعی است.
- جنبه ارزیابی:
توانایی مدل را در تشخیص موارد مثبت ارزیابی می کند. recall بالا نشان دهنده نرخ پایین منفی کاذب است.

- **F1 Score:**
- چه چیزی را اندازی گیری میکند:

- جنبه ارزیابی:

تعادل بین Precision و Recallرا ارزیابی می کند. بالا بودن این معیار نشان می دهد که مدل دارای تعادل قوی بین Precision و Recall است و به طور موثر موارد مثبت را بدون تعداد بالایی از مثبت کاذب یا منفی کاذب شناسایی می کند.

**پرسش 4:**

تفاوت میان Precision و Recall بیان کنید و توضیح دهید چرا هر کدام به تنهایی برای ارزیابی مدل کافی نیست؟ برای هر یک مثالی بیاورید که در آن، این معیار مقدار بالایی دارد اما مدل عملکرد خوبی ندارد.

پاسخ:

recall و Precision هر دو معیارهای مهمی هستند که برای ارزیابی عملکرد مدل‌های classification مورد استفاده قرار می‌گیرند، به‌ویژه در سناریوهایی که توزیع کلاس‌ها نامتعادل است یا زمانی که هزینه‌های مثبت کاذب و منفی کاذب به طور قابل‌توجهی متفاوت است.

- تفاوت بین recall و Precision

- recall نسبت موارد مثبت واقعی را که به درستی توسط مدل شناسایی شده اند اندازه گیری می کند. این به توانایی مدل برای یافتن تمام موارد مرتبط در یک مجموعه داده مربوط می شود. recall بالا به این معنی است که مدل در گرفتن نمونه های مثبت خوب است، اما لزوماً نشان نمی دهد که چند نمونه منفی به اشتباه به عنوان مثبت (مثبت نادرست) برچسب گذاری شده اند.

-Precision پیش بینی های مثبت انجام شده توسط مدل را اندازه گیری می کند. یعنی از بین تمام مواردی که مدل به عنوان مثبت شناسایی کرد، چند مورد در واقع مثبت هستند؟ Precision بالا به این معنی است که مدل در طبقه بندی های مثبت خود قابل اعتماد است اما نشان نمی دهد که چند نمونه مثبت از دست می دهد (منفی های کاذب).

چرا هر کدام به تنهایی برای ارزیابی مدل کافی نیست؟

برای پاسخ به این سوال دو سناریو را مورد بررسی قرار میدهیم:

- سناریوی با recall بالا، Precision کم: یک مدل تشخیص پزشکی طراحی شده برای شناسایی یک بیماری نادر را در نظر بگیرید. اگر این مدل برای اولویت‌بندی recall پیکربندی شده باشد، ممکن است بسیاری از افراد را به عنوان مبتلا به این بیماری برچسب‌گذاری کند (برای اطمینان از اینکه هیچ مورد واقعی را از دست ندهد)، از جمله بسیاری از افراد که این بیماری

را ندارند. در اینجا، recall زیاد است زیرا مدل تمام موارد واقعی بیماری را شناسایی می‌کند، اما Precision پایین است چرا که تعداد زیادی از موارد مثبت کاذبند. این سناریو می‌تواند منجر به آزمایش‌های بیشتر و درمان برای کسانی شود که به اشتباه تشخیص داده شده‌اند.

- Precision بالا، سناریوی recall کم: سیستمی را تصور کنید که هرزنامه را شناسایی کند که بسیار محتاط است و فقط در صورتی که کاملاً مطمئن باشد، ایمیل را به عنوان هرزنامه برچسب گذاری می کند. چنین مدلی ممکن است Precision بسیار بالایی داشته باشد زیرا تقریباً هر ایمیلی که به‌عنوان هرزنامه علامت‌گذاری می‌کند در واقع هرزنامه است، اما اگر بسیاری از ایمیل‌های هرزنامه را که شناسایی آن‌ها کمی سخت‌تر است از دست بدهد، می‌تواند recall کمی داشته باشد. در این مورد، صندوق ورودی ممکن است همچنان مملو از هرزنامه باشد، که نشان دهنده عملکرد کلی ضعیف با وجود Precision بالا است.

برای هر یک مثالی بیاورید که در آن، این معیار مقدار بالایی دارد اما مدل عملکرد خوبی ندارد.

علاوه بر دو سناریو و مثال بالا در ادامه دو مثال دیگر نیز اورده ایم:

1. recall بالا، عملکرد نامناسب:

مدلی که برای شناسایی تقلب در تراکنش های مالی طراحی شده است، ممکن است فراخوانی را در اولویت قرار دهد تا بیشترین تعداد تراکنش های متقلبانه را به دست آورد. اگر مدل بیش از حد حساس باشد و تعداد زیادی از تراکنش ها را به عنوان جعلی اعلام کند (از جمله بسیاری از تراکنش های قانونی)، recall بالا ولی Precision پایینی خواهد داشت. پیامد آن می‌تواند باعث ناراحتی تعداد قابل توجهی از مشتریان با پرچم‌گذاری تراکنش‌های قانونی آن‌ها شود که به طور بالقوه به تجربه کاربر و اعتماد به سیستم آسیب می‌رساند.

2. Precision بالا، عملکرد نامناسب:

یک سیستم recommendation محتوا که فقط زمانی محتوا را به کاربر توصیه می کند که بسیار مطمئن باشد کاربر از آن خوشش می آید ممکن است Precision بسیار بالایی را نشان دهد (کاربر بیشتر محتوای توصیه شده را دوست دارد) . با این حال، اگر این رویکرد محتاطانه منجر به توصیه های بسیار کمی شود (از دست دادن محتوایی که کاربر ممکن است دوست داشته باشد)، فراخوان کم خواهد بود. با وجود Precision بالا، عملکرد کلی سیستم خوب نیست زیرا نمی تواند کاربر را با محتوای کافی درگیر کند.

در نتیجه، تکیه صرف بر recall یا Precision می تواند تصویر گمراه کننده ای از عملکرد یک مدل ارائه دهد. متعادل کردن هر دو، اغلب از طریق معیارهایی مانند معیار F1، دید جامع تری از اثربخشی یک مدل ارائه می دهد، به خصوص در برنامه هایی که هر دو مثبت کاذب و منفی کاذب پیامدهای مهمی دارند.

**پرسش 5:**

معیار F1 از چه نوع میانگین گیری استفاده میکند؟ تفاوت این نوع میانگین گیری با میانگین گیری عادی چیست و در اینجا چرا اهمیت دارد؟

پاسخ:

معیار F1 از میانگین هارمونیک برای محاسبه میانگین Precision و Recallاستفاده می کند. میانگین هارمونیک نوعی میانگین است که مخصوصاً هنگام rates با ratioها یا نسبت ها مفید است و دیدگاهی متفاوت از میانگین حسابی (میانگین معمولی) که معمولاً در بسیاری از محاسبات استفاده می شود ارائه می دهد.

- میانگین هارمونیک

فرمول میانگین هارمونیک دو عدد x و yبه صورت زیر بدست می آید:

$$\frac{2xy}{x+y}$$

- میانگین حسابی

در مقابل، میانگین حسابی (یا میانگین معمولی) دو عدد x و به صورت زیر محاسبه می شود:

$$\frac{x+y}{2}$$

**تفاوت و اهمیت (برای معیار F1)**

حساسیت میانگین هارمونیک به مقادیر کم -> متعادل کردن Precision و Recall

میانگین هارمونیک به مقادیر پایین حساس تر از میانگین حسابی است. اگر Precision یا Recall کم باشد، میانگین هارمونیک (معیار F1) به طور قابل توجهی کمتر از میانگین حسابی آن دو خواهد بود. این ویژگی باعث می‌شود که میانگین هارمونیک به ویژه برای موقعیت‌هایی مفید باشد که می‌خواهیم اطمینان حاصل کنیم که هر دو اعداد متوسط به طور معقولی بالا هستند و یک مقدار فوق‌العاده بالا نتیجه را بیش از حد منحرف نمی‌کند. این دقیقاً همان سناریویی است که ما به آن به دنبال تعادلیم و یکی از دو معیار نباید کمبود شدید دیگری را جبران کند.

به طور خلاصه، حساسیت میانگین هارمونیک به مقادیر پایین، آن را برای محاسبه معیار F1 ایده‌آل می‌کند، زیرا تضمین می‌کند که Precision و Recallبه طور منصفانه در نظر گرفته شده و مدل راا به سمت تعادل این دو سوق میدهد. این در بسیاری از کاربردهای عملی که هم اجتناب از مثبت کاذب و هم شناسایی همه موارد مثبت واقعی مهم است، بسیار پرکاربردست.

## پیاده سازی توابع برای ارزیابی:

### 1. Confusion Matrix:

1. ابتدا یک تابع نیاز داریم تا matrix را بسازیم:

```
def create_confusion_matrix(y_true, y_pred):
    classes = sorted(set(y_true))
    class_indices = {cls: i for i, cls in enumerate(classes)}
    matrix = [[0 for _ in classes] for _ in classes]
    for actual, predicted in zip(y_true, y_pred):
        i = class_indices[actual]
        j = class_indices[predicted]
        matrix[i][j] += 1

    return matrix
```

2. سپس یک تابع تعریف میکنیم تا heat map را برای یک matrix رسم کنید تا ارزیابی راحت تر شود:

```
def plot_confusion_matrix(matrix, classes):
    fig, ax = plt.subplots()
    cax = ax.matshow(matrix, cmap=plt.cm.Spectral)
    fig.colorbar(cax)

    plt.xticks(np.arange(len(classes)), classes, rotation=45)
    plt.yticks(np.arange(len(classes)), classes)

    ax.set_xlabel('Predicted')
    ax.xaxis.set_label_position('top')
    ax.set_ylabel('True')

    for (i, j), val in np.ndenumerate(matrix):
        ax.text(j, i, f'{val}', ha='center', va='center', color='black')

    plt.show()
```

3. در اخر به کمک تابع زیر matrix را ساخته و رسم میکنیم:

```
def create_plot_confusion_matrix(true_vals, predict_vals, vals, type, part):
    print(f"for {part}:")
    cm = create_confusion_matrix(true_vals, predict_vals)
    plot_confusion_matrix(cm, vals)
    return np.array(cm)
```

**برای محاسبه باقی متریک ها ابتدا تابع زیر را تعریف کرده ایم تا تمام مقادیر مورد نیاز برای محاسبه متیرک ها برای یک کلاس را بدست بیاوریم:**

```
def calculate_metrics(confusion_matrix, class_index):
    TP = confusion_matrix[class_index, class_index]
    TN = np.sum(confusion_matrix) - np.sum(confusion_matrix[class_index, :]) - np.sum(confusion_matrix[:, class_index]) + TP
    FP = np.sum(confusion_matrix[:, class_index]) - TP
    FN = np.sum(confusion_matrix[class_index, :]) - TP
    return TP, TN, FP, FN
```

**سپس با توجه به توضیحات پرسش اول این بخش، توبع زیر را پیاده سازی میکینم:**

### 2. Accuracy:

```
def print_accuracy(confusion_matrix, vals):
    accs = []
    for i in range(len(vals)):
        TP, TN, FP, FN = calculate_metrics(confusion_matrix, i)
        ALL = TP + TN + FP + FN
        T = TP + TN
        acc = T/ALL
        accs.append(acc)
        print(f"  {vals[i]} : {acc* 100:.2f}")
    print(f"macro avg accuracy = {sum(accs)/len(accs) * 100:.2f}%")
    print("--" * 15)
```

```
def print_accuracies(cm_1, cm_2, vals, type):
    print("accuracies for " + type + "-" + "part 1:")
    print_accuracy(cm_1, vals)
    print("accuracies for " + type + "-" + "part 2:")
    print_accuracy(cm_2, vals)
```

### 3. Precision:

```
def print_precision(confusion_matrix, vals):
    precs = []
    for i in range(len(vals)):
        TP, TN, FP, FN = calculate_metrics(confusion_matrix, i)
        precision = TP / (TP + FP)
        precs.append(precision)
        print(f"  {vals[i]} : {precision* 100:.2f}")
    print(f"macro avg precision = {sum(precs)/len(precs) * 100:.2f}%")
    print("--" * 15)
```

```
def print_precisions(cm_1, cm_2, vals, type):
    print("precision for " + type + "-" + "part 1:")
    print_precision(cm_1, vals)
    print("precision for " + type + "-" + "part 2:")
    print_precision(cm_2, vals)
```

**4. Recall:**

```
def print_recall(confusion_matrix, vals):
    recs = []
    for i in range(len(vals)):
        TP, TN, FP, FN = calculate_metrics(confusion_matrix, i)
        recall = TP / (TP + FN)
        recs.append(recall)
        print(f"  {vals[i]} : {recall* 100:.2f}")
    print(f"macro avg recall = {sum(recs)/len(recs) * 100:.2f}%")
    print("--" * 15)
```

```
def print_recalls(cm_1, cm_2, vals, type):
    print("recall for " + type + "-" + "part 1:")
    print_recall(cm_1, vals)
    print("recall for " + type + "-" + "part 2:")
    print_recall(cm_2, vals)
```

**5. F1 Score:**

```
def print_f1(confusion_matrix, vals):
    f1s = []
    for i in range(len(vals)):
        TP, TN, FP, FN = calculate_metrics(confusion_matrix, i)
        p =  TP / (TP + FP)
        r = TP / (TP + FN)
        f1 = 2 * r * p / (r + p)
        f1s.append(f1)
        print(f"  {vals[i]} : {f1* 100:.2f}")
    print(f"macro avg F1 score = {sum(f1s)/len(f1s) * 100:.2f}%")
    print("--" * 15)
```

```
def print_f1_scores(cm_1, cm_2, vals, type):
    print("F1 score for " + type + "-" + "part 1:")
    print_f1(cm_1, vals)
    print("F1 score for " + type + "-" + "part 2:")
    print_f1(cm_2, vals)
```

**6.** برای مقایسه راحت تر از توابع زیر و نمودار استفاده کرده ایم:

```
def all_metrics(confusion_matrix, vals):
    f1s = []
    recs = []
    accs = []
    precs = []
    for i in range(len(vals)):
        TP, TN, FP, FN = calculate_metrics(confusion_matrix, i)
        ALL = TP + TN + FP + FN
        T = TP + TN
        acc = T/ALL
        pre =  TP / (TP + FP)
        recall = TP / (TP + FN)
        f1 = 2 * recall * pre / (recall + pre)
        f1s.append(f1)
        recs.append(recall)
        accs.append(acc)
        precs.append(pre)
    return  accs, precs, recs, f1s
```

```python
def all_metric_2_approch(cm_1, cm_2, vals, type):
    accs1, precs1, recs1, f1s1 = all_metrics(cm_1, vals)
    accs2, precs2, recs2, f1s2 = all_metrics(cm_2, vals)

    x = np.arange(len(vals))
    width = 0.35
    fig, axs = plt.subplots(2, 2, figsize=(12, 10))

    # Plotting accuracy
    rects1 = axs[0, 0].bar(x - width/2, accs1, width, label='Accuracy - part 1')
    rects2 = axs[0, 0].bar(x + width/2, accs2, width, label='Accuracy - part 2')
    axs[0, 0].set_title('Accuracy Comparison')
    axs[0, 0].set_xticks(x)
    axs[0, 0].set_xticklabels(vals)
    axs[0, 0].legend()

    avg_acc1 = np.mean(accs1)
    avg_acc2 = np.mean(accs2)
    var_acc1 = np.var(accs1)
    var_acc2 = np.var(accs2)
    axs[0, 0].annotate(f'Avg 1: {avg_acc1:.2f}, Var 1: {var_acc1:.2f}\nAvg 2: {avg_acc2:.2f}, Var 2: {var_acc2:.2f}',
                       xy=(0.5, 0.95), xycoords='axes fraction', ha='center', va='center')

    # Plotting precision
    rects1 = axs[0, 1].bar(x - width/2, precs1, width, label='Precision - part 1')
    rects2 = axs[0, 1].bar(x + width/2, precs2, width, label='Precision - part 2')
    axs[0, 1].set_title('Precision Comparison')
    axs[0, 1].set_xticks(x)
    axs[0, 1].set_xticklabels(vals)
    axs[0, 1].legend()

    avg_prec1 = np.mean(precs1)
    avg_prec2 = np.mean(precs2)
    var_prec1 = np.var(precs1)
    var_prec2 = np.var(precs2)
    axs[0, 1].annotate(f'Avg 1: {avg_prec1:.2f}, Var 1: {var_prec1:.2f}\nAvg 2: {avg_prec2:.2f}, Var 2: {var_prec2:.2f}',
                       xy=(0.5, 0.95), xycoords='axes fraction', ha='center', va='center')

    # Plotting recall
    rects1 = axs[1, 0].bar(x - width/2, recs1, width, label='Recall - part 1')
    rects2 = axs[1, 0].bar(x + width/2, recs2, width, label='Recall - part 2')
    axs[1, 0].set_title('Recall Comparison')
    axs[1, 0].set_xticks(x)
    axs[1, 0].set_xticklabels(vals)
    axs[1, 0].legend()

    avg_recall1 = np.mean(recs1)
    avg_recall2 = np.mean(recs2)
    var_recall1 = np.var(recs1)
    var_recall2 = np.var(recs2)
    axs[1, 0].annotate(f'Avg 1: {avg_recall1:.2f}, Var 1: {var_recall1:.2f}\nAvg 2: {avg_recall2:.2f}, Var 2: {var_recall2:.2f}',
                       xy=(0.5, 0.95), xycoords='axes fraction', ha='center', va='center')

    # Plotting F1 score
    rects1 = axs[1, 1].bar(x - width/2, f1s1, width, label='F1 Score - part 1')
    rects2 = axs[1, 1].bar(x + width/2, f1s2, width, label='F1 Score - part 2')
    axs[1, 1].set_title('F1 Score Comparison')
    axs[1, 1].set_xticks(x)
    axs[1, 1].set_xticklabels(vals)
    axs[1, 1].legend()

    avg_f1_1 = np.mean(f1s1)
    avg_f1_2 = np.mean(f1s2)
    var_f1_1 = np.var(f1s1)
    var_f1_2 = np.var(f1s2)
    axs[1, 1].annotate(f'Avg 1: {avg_f1_1:.2f}, Var 1: {var_f1_1:.2f}\nAvg 2: {avg_f1_2:.2f}, Var 2: {var_f1_2:.2f}',
                       xy=(0.5, 0.95), xycoords='axes fraction', ha='center', va='center')

    plt.xticks(rotation=45, ha='right')
    plt.tight_layout()

    plt.show()
```

## پرسش 6:

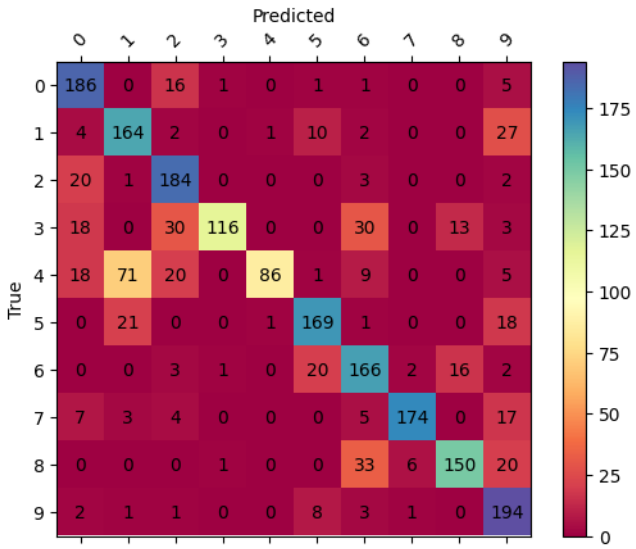
0. مدل خود را بر اساس رقم گفته شده در فایل صوتی آماده کنید. (در بخش قبل این مرحله انجام شده است)
1. Confusion Matrix رسم کنید.
2. معیار Accuracy را محاسبه کنید.
3. معیار Precision را محاسبه کنید.
4. در آخر مقادیر به دست آمده را تحلیل کنید.

پاسخ:

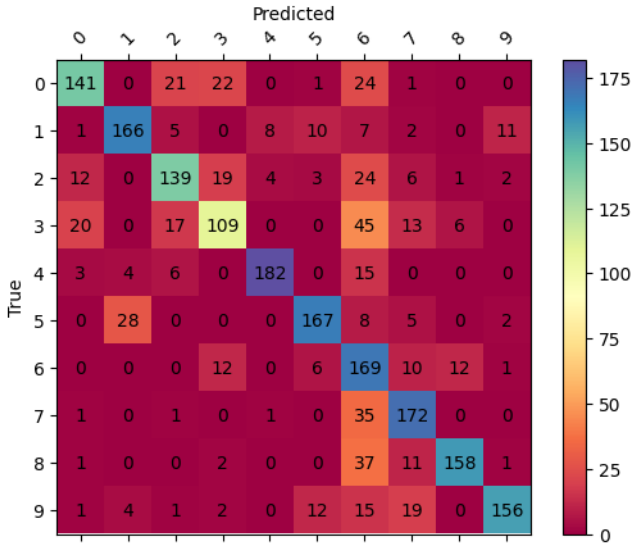
1. Confusion Matrix:

```
DIGITS = list(set(test_true_digits))
DIGITS = sorted(DIGITS)
print(DIGITS)
cm_d_1 = create_plot_confusion_matrix(test_true_digits,test_predictions_digits_1, DIGITS, "digits", "part 1")
cm_d_2 = create_plot_confusion_matrix(test_true_digits,test_predictions_digits_2, DIGITS, "digits", "part 2")
```

```
['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
for part 1:
```




```
for part 2:
```



2. متریک های بررسی دقت:

```
print_recalls(cm_d_1, cm_d_2, DIGITS, "digits")
```

```
recall for digits-part 1:
  0 : 88.57
  1 : 78.10
  2 : 87.62
  3 : 55.24
  4 : 40.95
  5 : 80.48
  6 : 79.05
  7 : 82.86
  8 : 71.43
  9 : 92.38
macro avg recall = 75.67%
------------------------------
recall for digits-part 2:
  0 : 67.14
  1 : 79.05
  2 : 66.19
  3 : 51.90
  4 : 86.67
  5 : 79.52
  6 : 80.48
  7 : 81.90
  8 : 75.24
  9 : 74.29
macro avg recall = 74.24%
------------------------------
```

```
print_accuracies(cm_d_1, cm_d_2, DIGITS, "digits")
```

```
accuracies for digits-part 1:
  0 : 95.57
  1 : 93.19
  2 : 95.14
  3 : 95.38
  4 : 94.00
  5 : 96.14
  6 : 93.76
  7 : 97.86
  8 : 95.76
  9 : 94.52
macro avg accuracy = 95.13%
------------------------------
accuracies for digits-part 2:
  0 : 94.86
  1 : 96.19
  2 : 94.19
  3 : 92.48
  4 : 98.05
  5 : 96.43
  6 : 88.05
  7 : 95.00
  8 : 96.62
  9 : 96.62
macro avg accuracy = 94.85%
------------------------------
```

```
print_f1_scores(cm_d_1, cm_d_2, DIGITS, "digits")
```

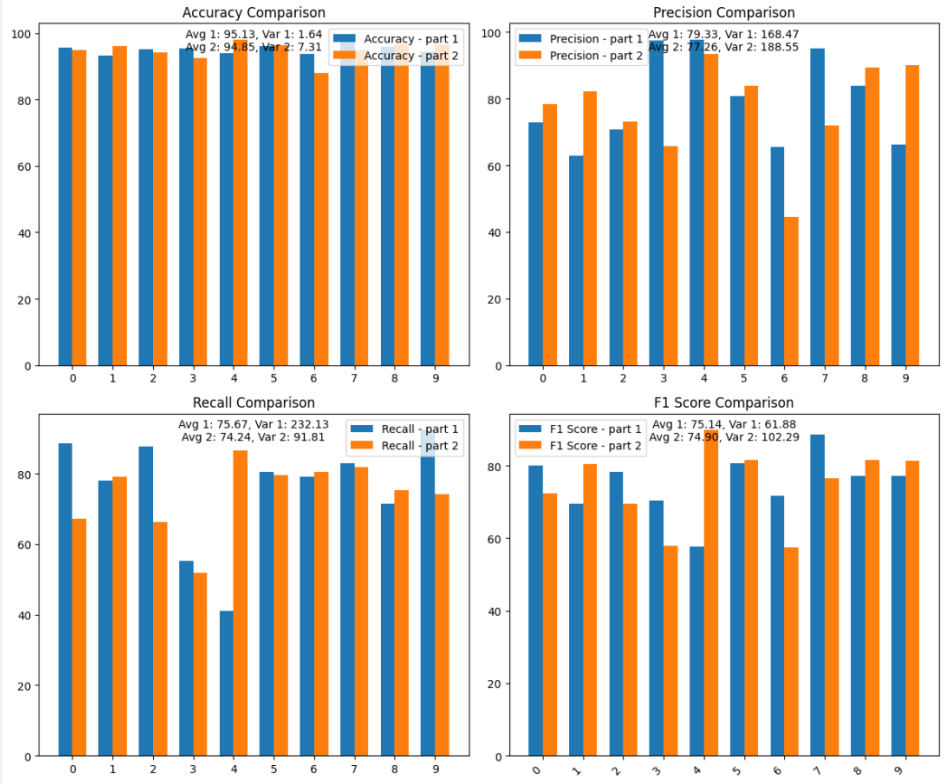
```
F1 score for digits-part 1:
  0 : 80.00
  1 : 69.64
  2 : 78.30
  3 : 70.52
  4 : 57.72
  5 : 80.67
  6 : 71.71
  7 : 88.55
  8 : 77.12
  9 : 77.14
macro avg F1 score = 75.14%
------------------------------
F1 score for digits-part 2:
  0 : 72.31
  1 : 80.58
  2 : 69.50
  3 : 57.98
  4 : 89.88
  5 : 81.66
  6 : 57.39
  7 : 76.61
  8 : 81.65
  9 : 81.46
macro avg F1 score = 74.90%
------------------------------
```

```
print_precisions(cm_d_1, cm_d_2, DIGITS, "digits")
```

```
precision for digits-part 1:
  0 : 72.94
  1 : 62.84
  2 : 70.77
  3 : 97.48
  4 : 97.73
  5 : 80.86
  6 : 65.61
  7 : 95.08
  8 : 83.80
  9 : 66.21
macro avg precision = 79.33%
------------------------------
precision for digits-part 2:
  0 : 78.33
  1 : 82.18
  2 : 73.16
  3 : 65.66
  4 : 93.33
  5 : 83.92
  6 : 44.59
  7 : 71.97
  8 : 89.27
  9 : 90.17
macro avg precision = 77.26%
------------------------------
```

- **تحلیل نتایج:**

```
all_metric_2_approch(cm_d_1, cm_d_2, DIGITS, "digits")
```



1. در اکثریت کلاس ها و به طور کلی مدل پیاده سازی شده با کتابخانه دقت بیشتری دارد.
2. برخی ارقام به طور چشم گیر کمتر از باقی ارقام تشخیص داده شده اند، مثلا 6 که میتواند به علت های مختلفی باشد، مثلا تلفظای مختلف افراد. (و متوجه میشویم این رقم در هر دو مدل درصد precision کمی داشته است.)
3. Precisionمیشود گفت بسیار سخت گیرانه تر از accuracy است.
4. بیشتر بودن accuracy نسبت به precision
5. مشخص است برای همه ارقام همیشه نمیتوان گفت یک مدل بهتر عمل کرده است، اما مشاهده میشود در اکثیرت به طور چشم گیری مدل اول بهتر بوده.

## پرسش 7:

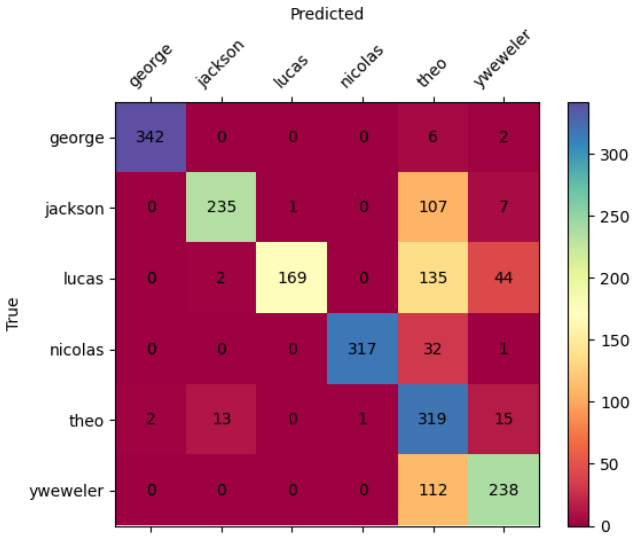
0. مدل خود را بر اساس رقم گفته شده در فایل صوتی آماده کنید. (در بخش قبل این مرحله انجام شده است)
1. Confusion Matrix رسم کنید.
2. معیار Accuracy را محاسبه کنید.
3. معیار Precision را محاسبه کنید.
4. در آخر مقادیر به دست آمده را تحلیل کنید.

پاسخ:

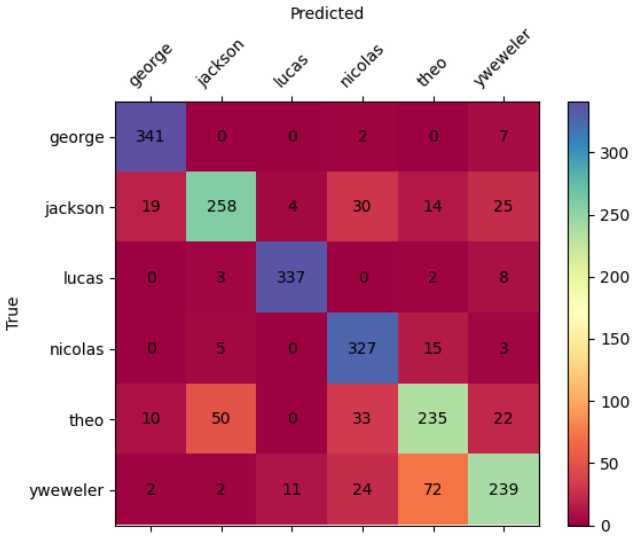
1. Confusion Matrix :

```
SPEAKERS = list(set(test_true_speakers))
SPEAKERS = sorted(SPEAKERS)
print(SPEAKERS)
cm_s_1 = create_plot_confusion_matrix(test_true_speakers,test_predictions_speakers_1, SPEAKERS, "speakers", "part 1")
cm_s_2 = create_plot_confusion_matrix(test_true_speakers,test_predictions_speakers_2, SPEAKERS, "speakers", "part 2")
```

```
['george', 'jackson', 'lucas', 'nicolas', 'theo', 'yweweler']
for part 1:
```




```
for part 2:
```



2. متریک های بررسی دقت:

```
print_accuracies(cm_s_1, cm_s_2, SPEAKERS, "speakers")
```

```
accuracies for speakers-part 1:
  george : 99.52
  jackson : 93.81
  lucas : 91.33
  nicolas : 98.38
  theo : 79.86
  yweweler : 91.38
macro avg accuracy = 92.38%
------------------------------
accuracies for speakers-part 2:
  george : 98.10
  jackson : 92.76
  lucas : 98.67
  nicolas : 94.67
  theo : 89.62
  yweweler : 91.62
macro avg accuracy = 94.24%
------------------------------
```

```
print_recalls(cm_s_1, cm_s_2, SPEAKERS, "speakers")
```

```
recall for speakers-part 1:
  george : 97.71
  jackson : 67.14
  lucas : 48.29
  nicolas : 90.57
  theo : 91.14
  yweweler : 68.00
macro avg recall = 77.14%
------------------------------
recall for speakers-part 2:
  george : 97.43
  jackson : 73.71
  lucas : 96.29
  nicolas : 93.43
  theo : 67.14
  yweweler : 68.29
macro avg recall = 82.71%
------------------------------
```

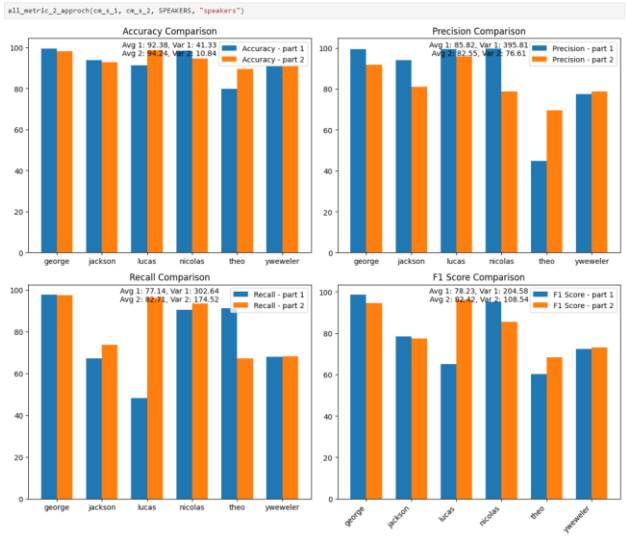
```
print_precisions(cm_s_1, cm_s_2, SPEAKERS, "speakers")
```

```
precision for speakers-part 1:
  george : 99.42
  jackson : 94.00
  lucas : 99.41
  nicolas : 99.69
  theo : 44.87
  yweweler : 77.52
macro avg precision = 85.82%
------------------------------
precision for speakers-part 2:
  george : 91.67
  jackson : 81.13
  lucas : 95.74
  nicolas : 78.61
  theo : 69.53
  yweweler : 78.62
macro avg precision = 82.55%
------------------------------
```

```
print_f1_scores(cm_s_1, cm_s_2, SPEAKERS, "speakers")
```

```
F1 score for speakers-part 1:
  george : 98.56
  jackson : 78.33
  lucas : 65.00
  nicolas : 94.91
  theo : 60.13
  yweweler : 72.45
macro avg F1 score = 78.23%
------------------------------
F1 score for speakers-part 2:
  george : 94.46
  jackson : 77.25
  lucas : 96.01
  nicolas : 85.38
  theo : 68.31
  yweweler : 73.09
macro avg F1 score = 82.42%
------------------------------
```

- **تحلیل نتایج:**




1. در اکثریت کلاس ها و به طور کلی مدل پیاده سازی شده با کتابخانه دقت بیشتری دارد.
2. عملکرد دو مدل تا حد مناسبی شبیه است، مثلا هر دو برای theo و yweweler ضعیف عمل کرده اند.
3. در مدل دوم که خودمان نوشته ایم برای دو شخصیت jacson وNicolas نیز به خوبی نتواسته مدل سازی کند پیدا کند.
4. Precisionمیشود گفت بسیار سخت گیرانه تر از accuracy است.
5. بیشتر بودن accuracy نسبت به precision
6. مشخص است برای همه افراد همیشه نمیتوان گفت یک مدل بهتر عمل کرده است، اما مشاهده میشود در اکثیرت به طور چشم گیری مدل اول بهتر بوده. (توجه شود نتایج تنها نسبت به این خروجی بیان نشده است و چندین بار ازمایش را انحام داده ایم.)

### پرسش 8:

تفاوت نتایج بخش های 6 و 7 را بررسی کنید و علل آن را مشخص کنید.

پاسخ:

نمیتوان تنها با یک بار اجرا نتیجه گرفت، پس آزمایش را علاوه بر این یک باری که در گزارش اورده ایم، چندین بار تکرار میکنیم: (توجه شود این نتیجه گیری نمیتواند کاملا مناسب باشد چون تعداد داده های Train به ازای هر گوینده/رقم برابر نیست.)

نتیجه ای که به چندین بار آزمایش با ناتخاب درصد های مختلفی از داده ها برای Trainو Testمیتوان گرفت، نشان میدهد به طور معمول، اکثرا برای اعداد متریک ها نتایج بهتری نشان میدهند که نشان دهنده عملکرد بهتر مدل ها برای اعداد است و این موضوع میتواند به چندین علت باشد:

1. تفاوت در لحن، بمی صدای افراد بسیار کم تر است.
2. برای تشخیص افراد به طور معمول نیاز به استیت ها و مدل های پیچده تر داریم.
3. برای ارقام تفاوت های بیشتری دارد و افراد متفاوت گفته اند، اما برای گویندگان حالت های مختلف شاید کمتر دیده شده باشد
4. برای تشخیص صدا شاید حذف نویز باید کمتر و ضیعف تر عمل میشود و نیاز به رکورد های طولانی اتر دارد.

### نتایج کلی تر:

نتایج مجزایی که با اجرای چندین بار ازمایش نیز میتوان گرفت:

- افزایش نسبت داده های Train به Test دقت را افزایش میدهد و نتایج بهتری میتوان گرفت
- افزایش تعداد State های پنهان، باعث افزایش چشم گیر دقت میتواند بشود، اما به شدت سرعت نتیجه گیری را کاهش میدهد.
- استفاده کلاس و کتابخانه تاثیر چشم گیری در دقت دارد.

## راهکار برای توسعه و بهبود پروژه:

- استفاده از متریک های بیشتر
- افزایش تعداد داده ها برای Train
- استفاده از feature های بیشتر و منتوع تر به همراه MFCC ها
- پیش پردازش های سینگین تر به همراه دقت های بالاتر
- افزایش تعداد استیت ها
- افزایش سرعت و بهینه کردن کد با ذخیره نکردن داده های mfcc و preprocessing در فایل

# منابع استفاده شده:


:HMM

- A novel approach to HMM-based speech recognition systems using particle swarm optimization - ScienceDirect
- Hidden Markov Model in Machine learning - GeeksforGeeks
- https://www.bing.com/ck/a?!&&p=44cbaa31437db2ebJmltdHM9MTcxMjE4ODgwMCZpZ3VpZD0yMThmMjgyNi0wNTg1LTYxMzQtMjQ3MS0zYmQ3MDQ4YzYwMWImaW5zaWQ9NTIxMA&ptn=3&ver=2&hsh=3&fclid=218f2826-0585-6134-2471-3bd7048c601b&psq=First+orfer+hmm&u=a1aHR0cHM6Ly93d3cuZGl2YS1wb3J0YWwub3JnL3NtYXNoL2dldC9kaXZhMjo4MzM2OTcvRlVMTFRFWFQwMS5wZGY&ntb=1
- /https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2766791
- https://en.wikipedia.org/wiki/Hidden_Markov_model
- https://www.researchgate.net/publication/228348090_Early_classifications_of_bearing_faults_using_hidden_Markov_models_Gaussian_mixture_models_Mel-frequency_cepstral_coefficients_and_fractals#pf8
- https://chat.openai.com/

:MFCCs

- Intuitive understanding of MFCCs. The mel frequency cepstral coefficients… | by Emmanuel Deruty | Medium
- MFCC Technique for Speech Recognition - Analytics Vidhya
- https://chat.openai.com/

:YouTube

- (ML 14.7) Forward algorithm (part 1) (youtube.com)
- Hidden Markov Models 09: the forward-backward algorithm (youtube.com)
- Forward Algorithm Clearly Explained | Hidden Markov Model | Part - 6 (youtube.com)

برای بخش تحلیل:

- Accuracy, precision, and recall in multi-class classification (evidentlyai.com)
- Confusion Matrix, Accuracy, Precision, Recall, F1 Score | by Harikrishnan N B | Analytics Vidhya | Medium