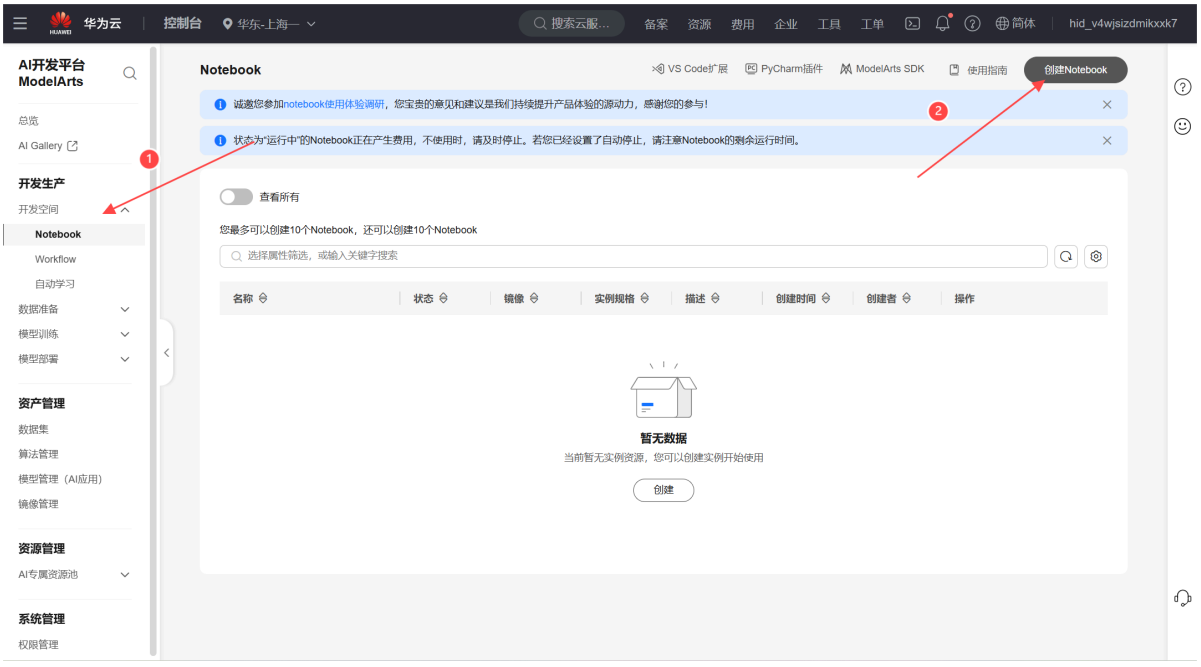
Assignment3: HMM-GMM

- Student name : **Juekai Lin**
- Student ID : **2253744**
- Tutor : **Ying Shen**

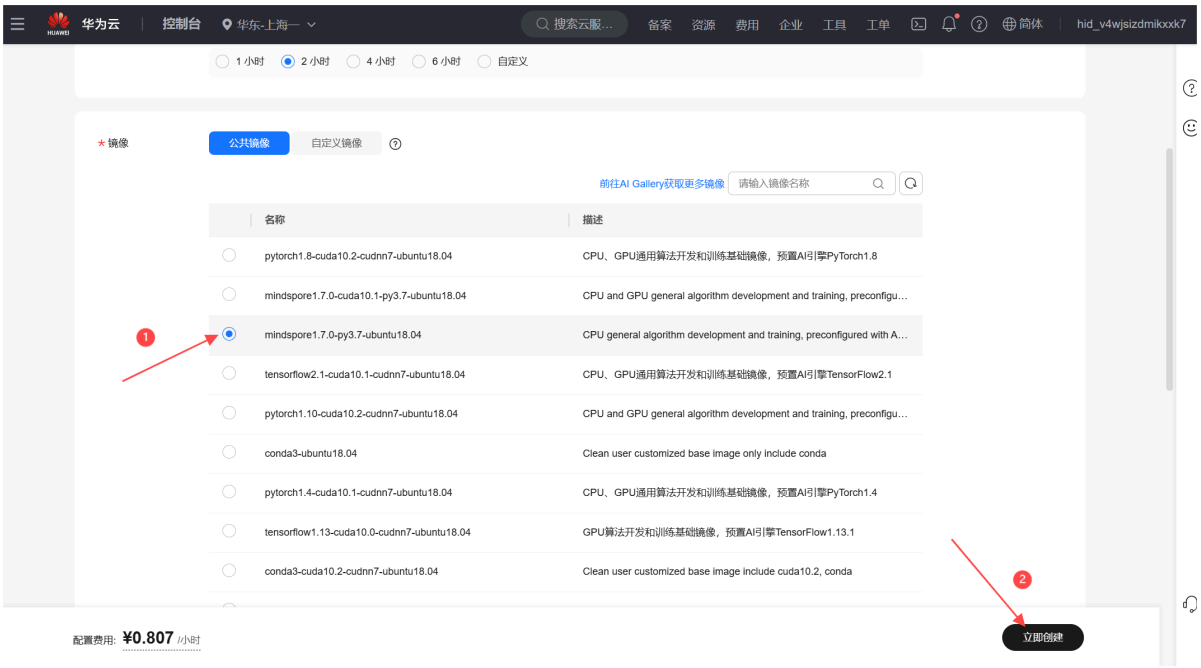
1. Experiment Report

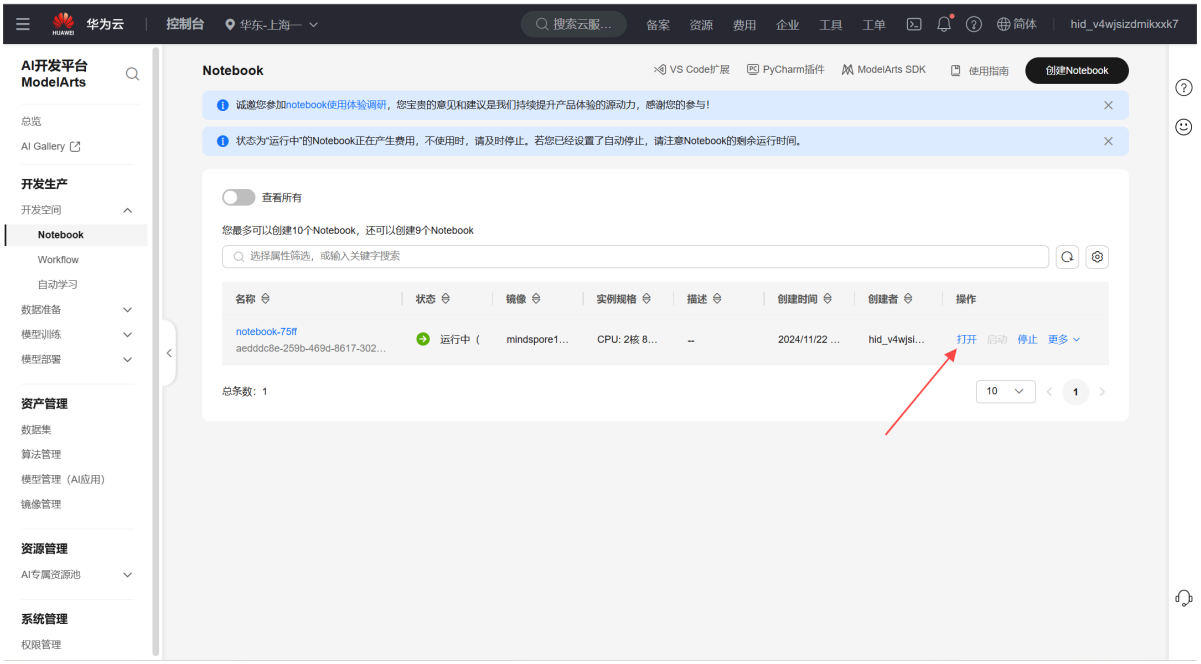
1.1. Setup and Start Notebook

Go to the Huawei Cloud ModelArts console, URL: https://console.huaweicloud.com/modelarts

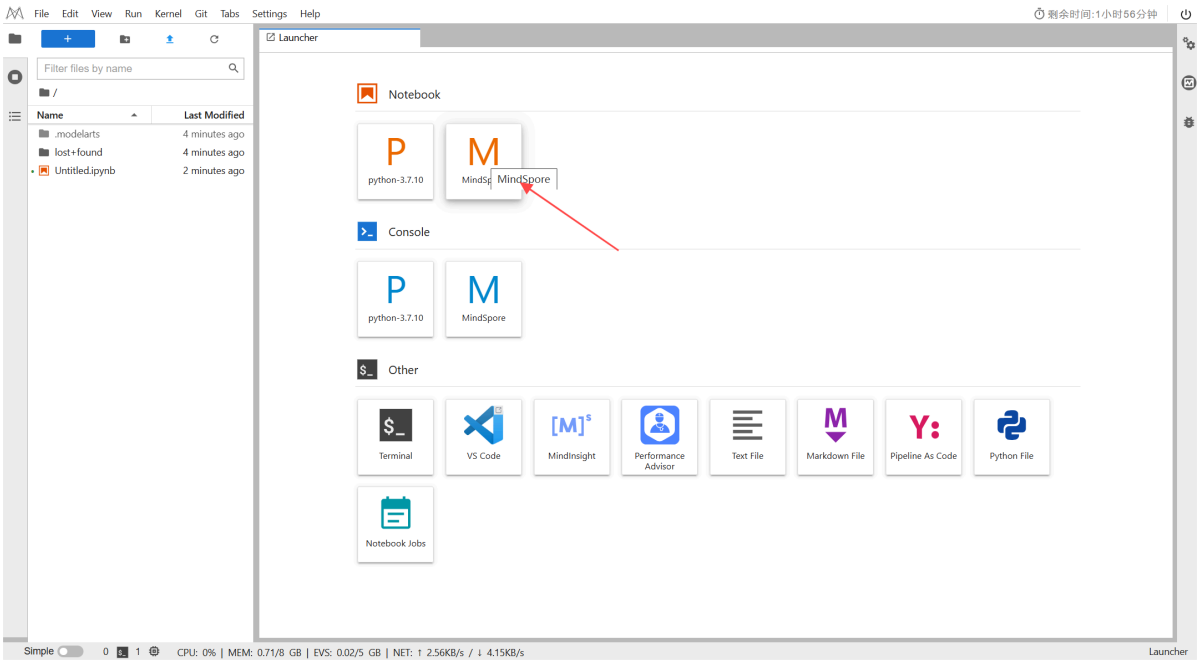


After entering the creation interface, select "mindspore1.7.0-py3.7-ubuntu18.04" in the public mirror By default, click Create Now to complete the create Notebook training job process, as shown in the figure below:



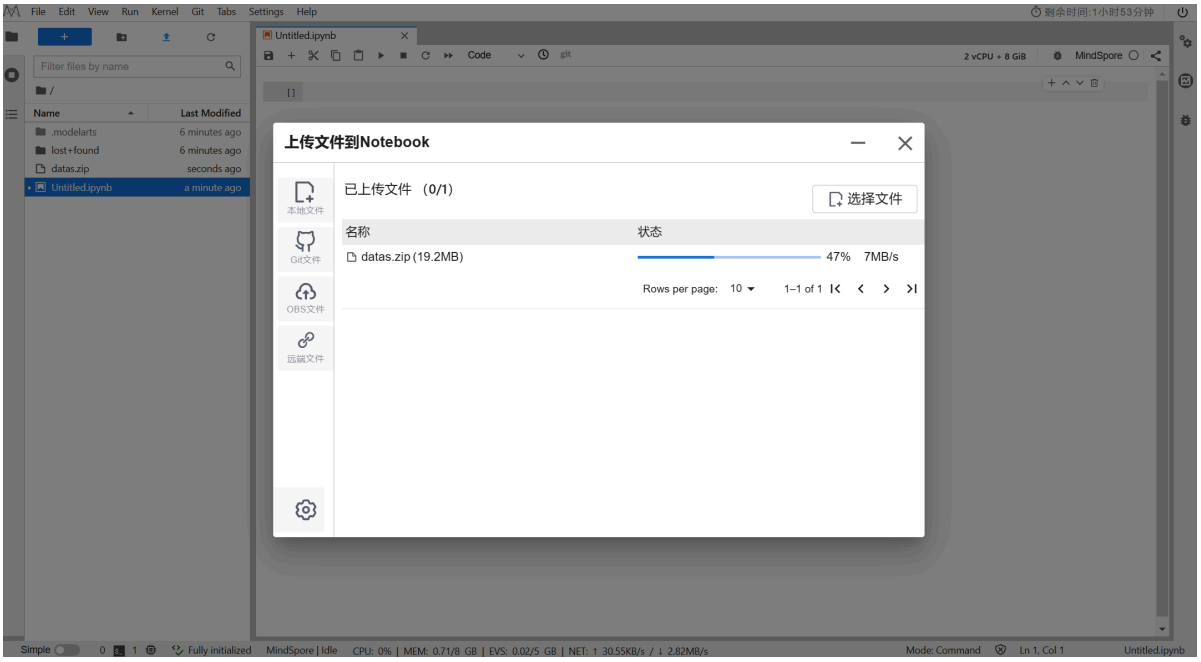


Start the Notebook into the development environment and Click on the MindSpore, as shown in the figure below:

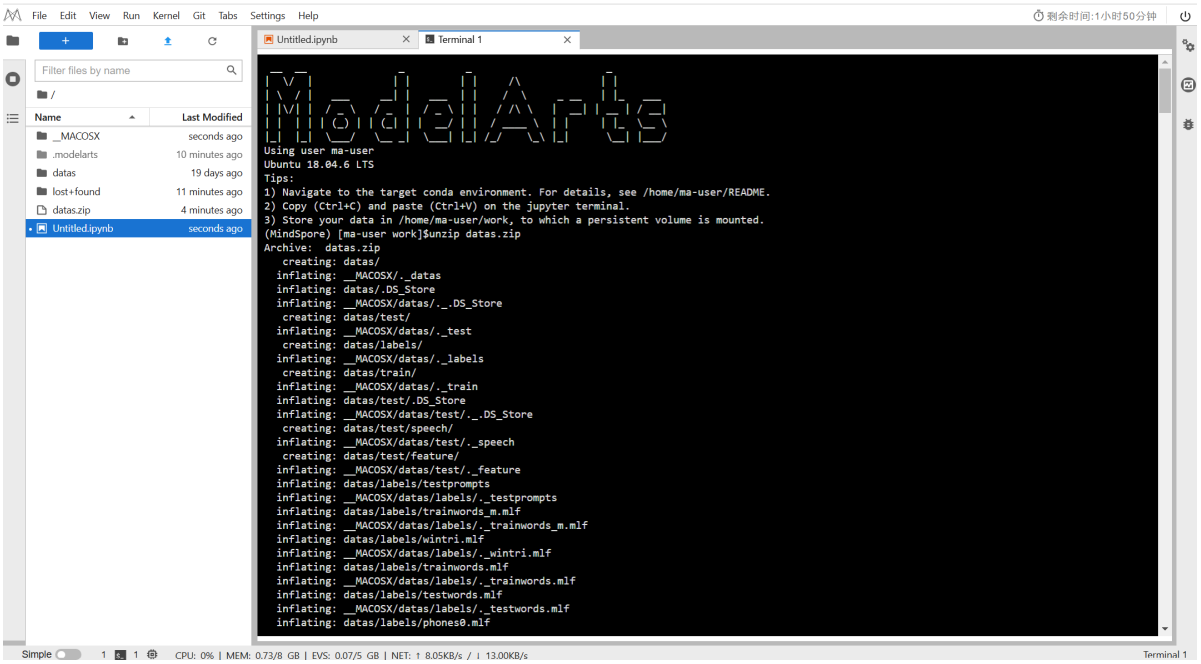


1.2. Data Preparation

Upload the data to the server, we download the existing data sets on the canvas, and upload the zip data to the server, as shown in the figure below:



Open the terminal input: unzip datas.zip, as shown in the figure below:



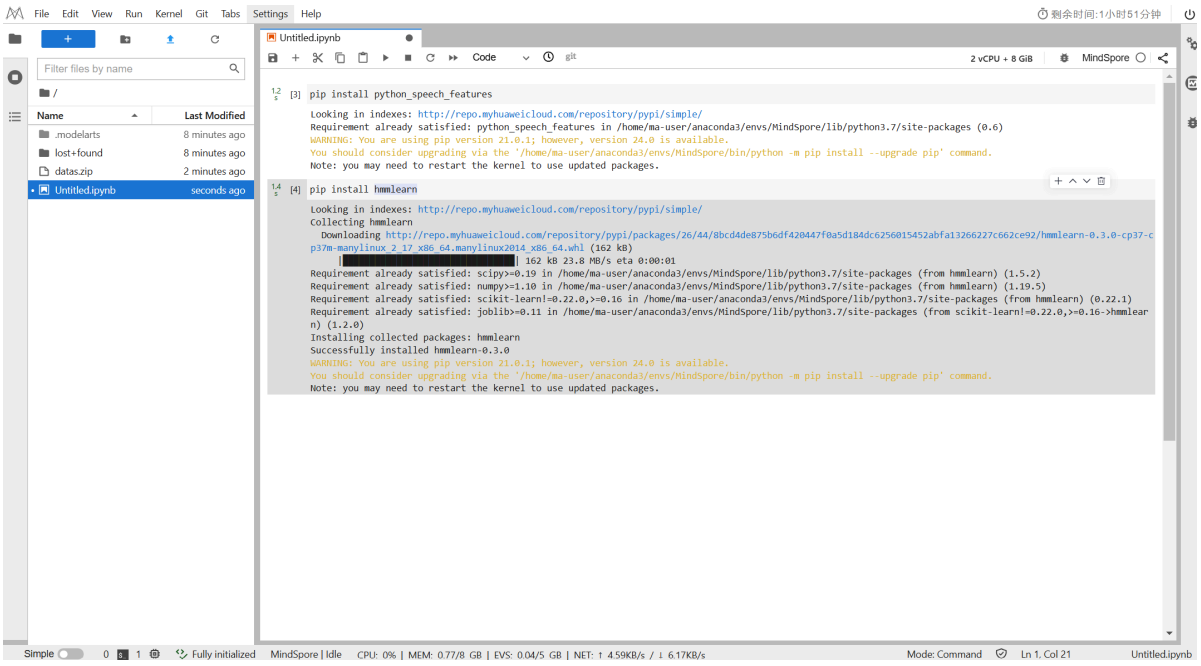
1.3. Training and Reasoning

1.3.1. Installation Library

Note Execute the install python Library command first.

```
pip install python_speech_features
pip install hmmlearn
```

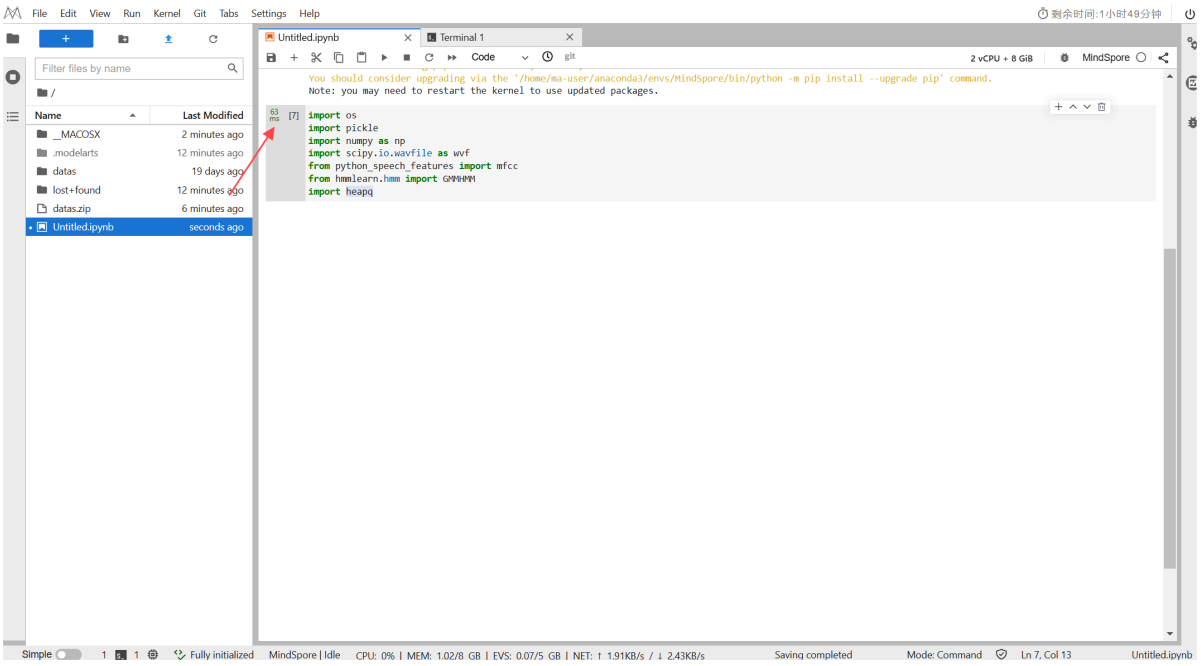
The python_speech_features is a library for extracting speech features, such as MFCC and filter bank energy, often used for speech recognition and speaker recognition; hmmlearn is a library for hidden Markov model (HMM) that can be used for sequence data modeling and analysis, as shown in the figure below:



1.3.2. Import the Desired Library

Import the desired library, as shown in the figure below:

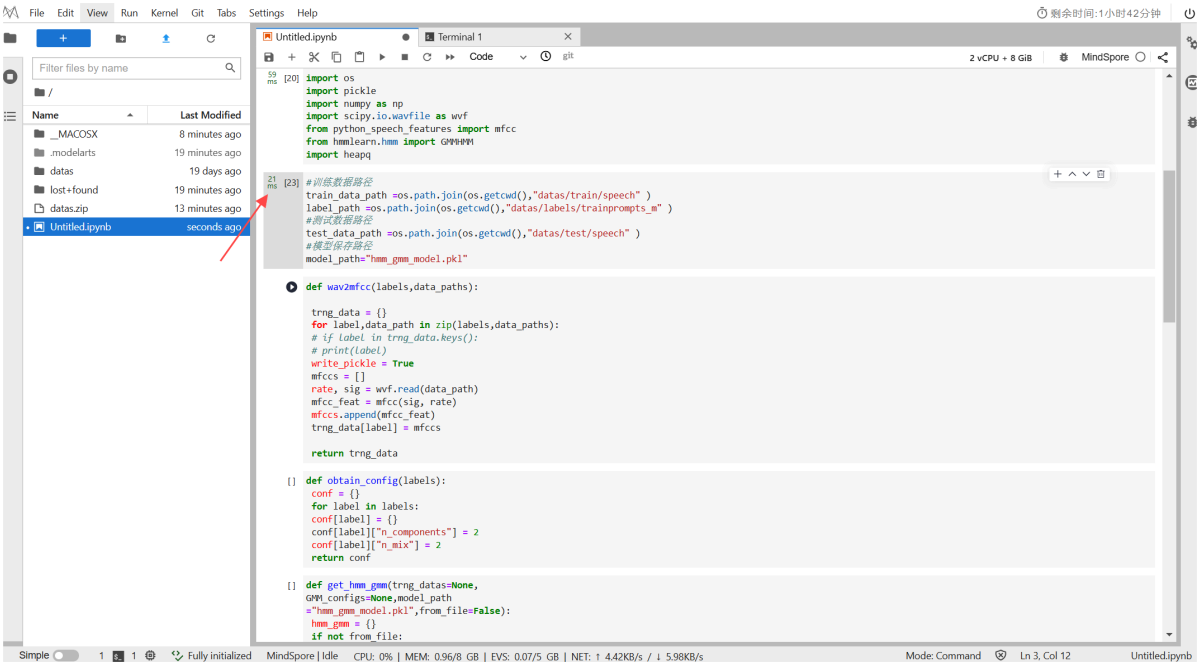
```
import os
import pickle
import numpy as np
import scipy.io.wavfile as wvf
from python_speech_features import mfcc
from hmmlearn.hmm import GMMHMM
import heapq
```



1.3.3. Configuration Path

Configure training, test data and model saved paths, as shown in the figure below:

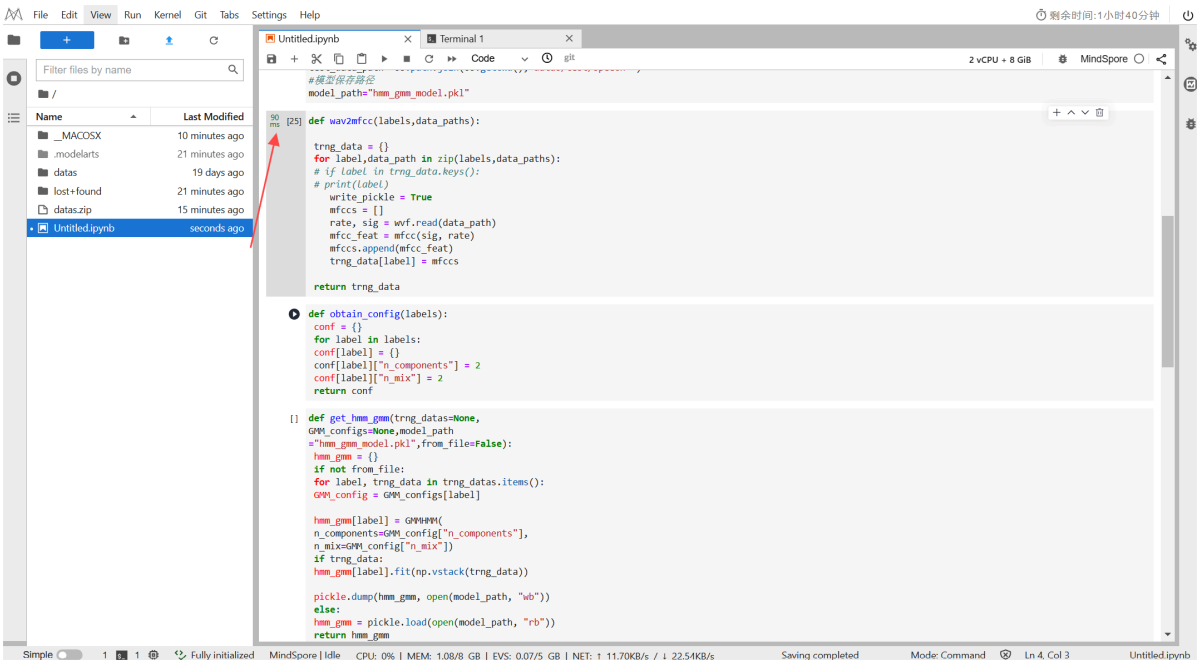
```
#训练数据路径
train_data_path =os.path.join(os.getcwd(),"datas/train/speech" )
label_path =os.path.join(os.getcwd(),"datas/labels/trainprompts_m" )
#测试数据路径
test_data_path =os.path.join(os.getcwd(),"datas/test/speech" )
#模型保存路径
model_path="hmm_gmm_model.pkl"
```



1.3.4. Define the Feature Extraction Function

A feature extraction function wav2mfcc is defined to extract MFCC features from a given speech file path and returns features in dictionary form, where each label corresponds to the MFCC feature of its audio file. The function reads audio files, extracts their MFCC features and is stored in trng _ data dictionary, which is finally used for further processing of speech data or model training, as shown in the figure below:

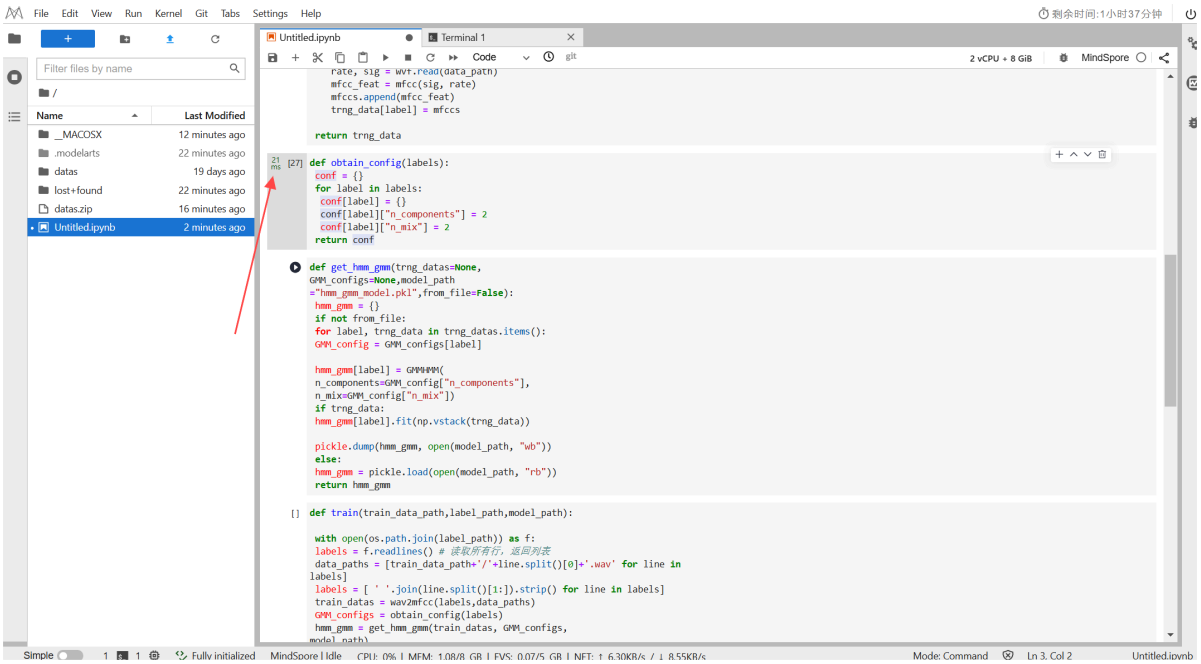
```
def wav2mfcc(labels,data_paths):

 trng_data = {}
 for label,data_path in zip(labels,data_paths):
 # if label in trng_data.keys():
 # print(label)
  write_pickle = True
  mfccs = []
  rate, sig = wvf.read(data_path)
  mfcc_feat = mfcc(sig, rate)
  mfccs.append(mfcc_feat)
  trng_data[label] = mfccs

 return trng_data
```



1.3.5. Define the Configuration Information for a Gaussian Mixture Model

Define the configuration information for a Gaussian mixture model, used to create a configuration information dictionary for a Gaussian mixture model (GMM) for each label, as shown in the figure below:

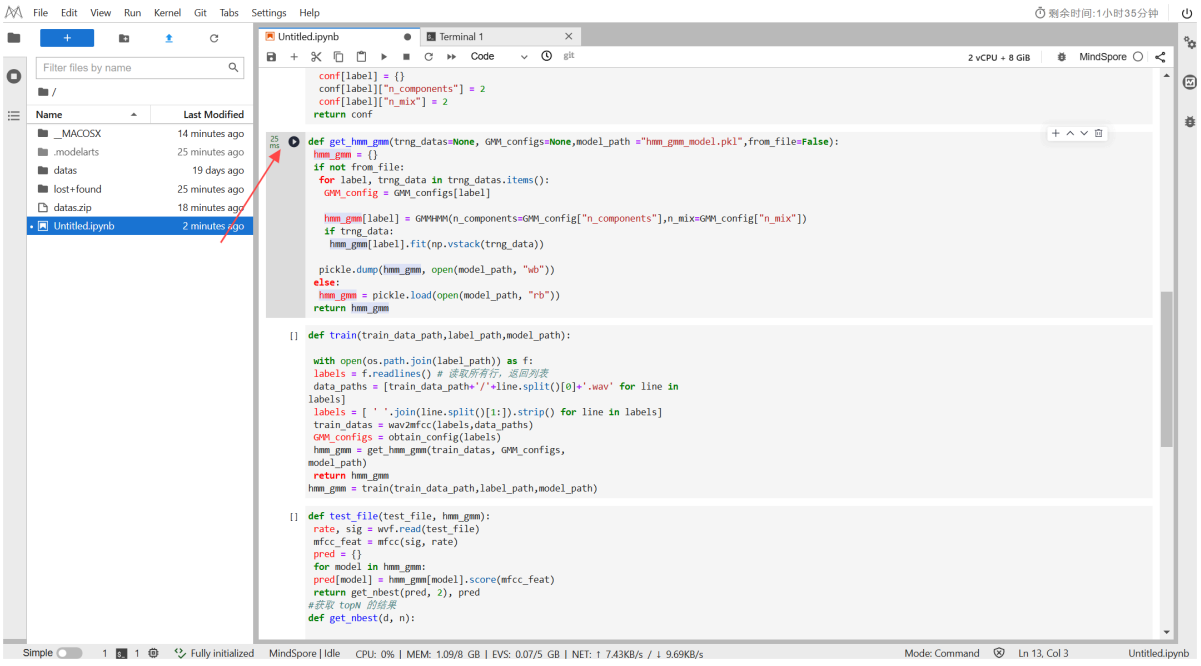
```
def obtain_config(labels):
 conf = {}
 for label in labels:
  conf[label] = {}
  conf[label]["n_components"] = 2
  conf[label]["n_mix"] = 2
 return conf
```



1.3.6. Create the GMM-HMM Model

Create GMM-HMM models used to generate GMM-HMM models based on training data and configuration, and support for loading or saving models from files. The function initializes the GMM-HMM model based on the configuration of each label, saved as a file after fitting using training data, or loaded from an existing model file, as shown in the figure below:

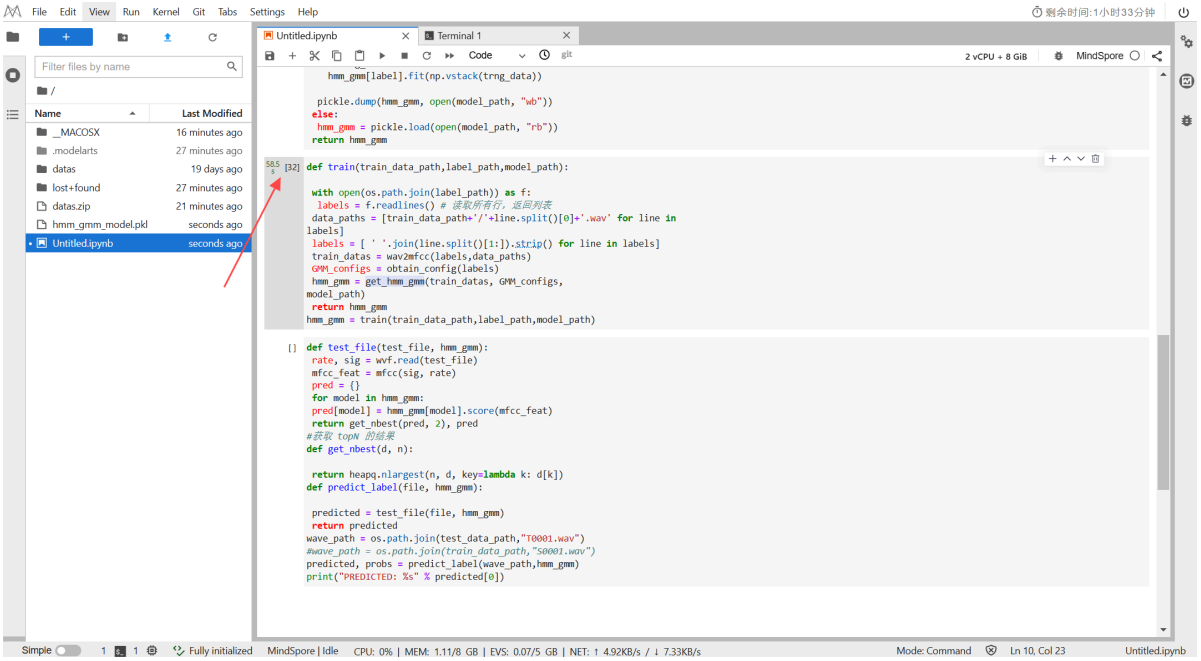
```
def get_hmm_gmm(trng_datas=None, GMM_configs=None,model_path
="hmm_gmm_model.pkl",from_file=False):
 hmm_gmm = {}
 if not from_file:
  for label, trng_data in trng_datas.items():
   GMM_config = GMM_configs[label]

   hmm_gmm[label] =
GMMHMM(n_components=GMM_config["n_components"],n_mix=GMM_config["n_mix"])
   if trng_data:
    hmm_gmm[label].fit(np.vstack(trng_data))

   pickle.dump(hmm_gmm, open(model_path, "wb"))
 else:
  hmm_gmm = pickle.load(open(model_path, "rb"))
 return hmm_gmm
```



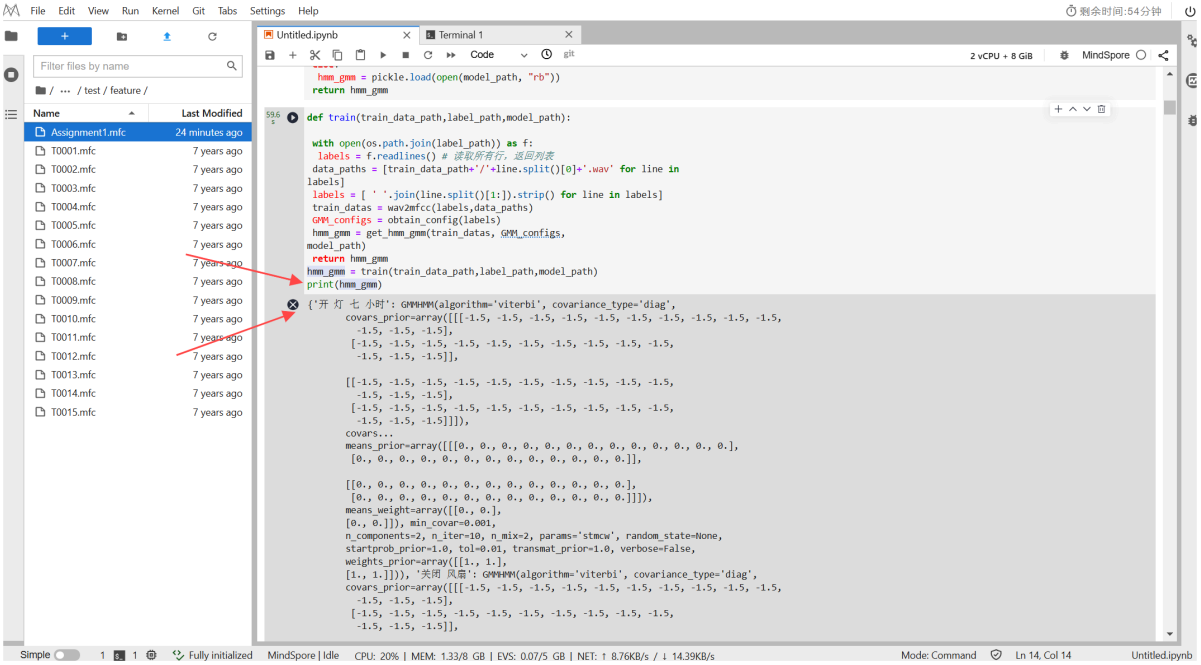
1.3.7. Read the Training Data and Train the Model

Read the training data, extract the MFCC features, and train the GMM-HMM model. The function generates the training data by reading the label file and the speech data path, calls the feature extraction function wav2mfcc and the configuration function obtain_config, and finally creates and trains the model using the get _ hmm _ gmm, while saving to the specified path, as shown in the figure below:

```
def train(train_data_path,label_path,model_path):

 with open(os.path.join(label_path)) as f:
 labels = f.readlines() # 读取所有行，返回列表
 data_paths = [train_data_path+'/'+line.split()[0]+'.wav' for line in labels]
 labels = [ ' '.join(line.split()[1:]).strip() for line in labels]
 train_datas = wav2mfcc(labels,data_paths)
 GMM_configs = obtain_config(labels)
 hmm_gmm = get_hmm_gmm(train_datas, GMM_configs, model_path)
 return hmm_gmm
hmm_gmm = train(train_data_path,label_path,model_path)
```



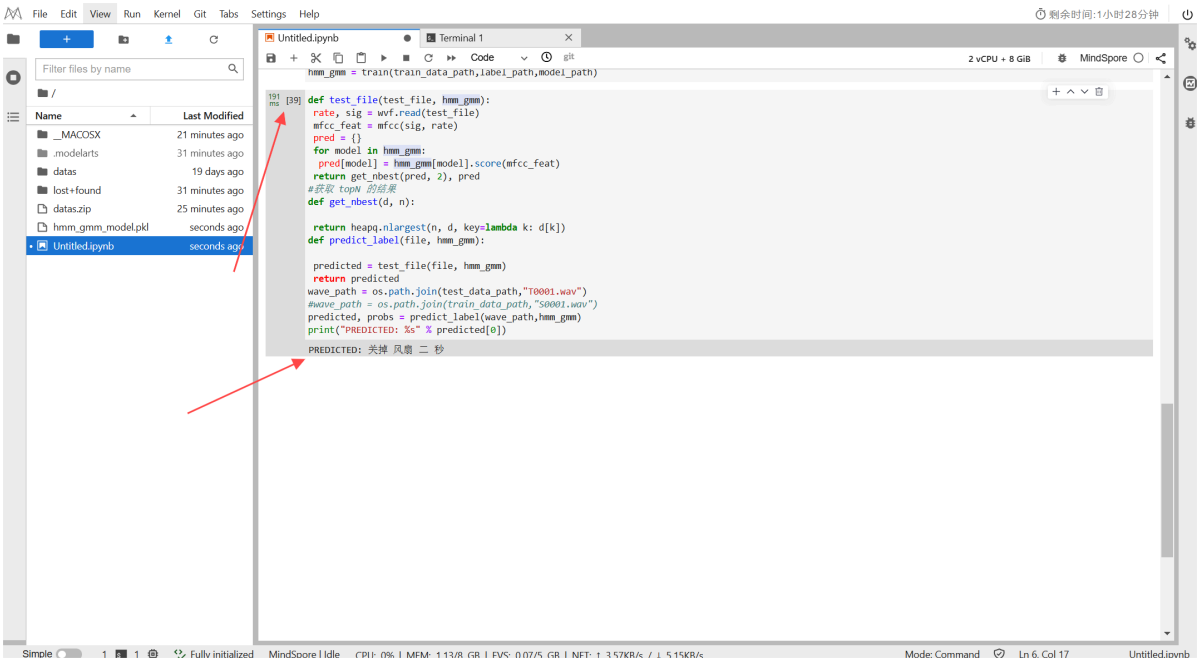
Of course, we can also add ***print (hmm _ gmm)*** at the end of the code to see the output details:



1.3.8. Call the Test

The feature extracts the input test audio file and uses the GMM-HMM model to calculate the score of each model to obtain the most matching label; the get_nbest function obtains the top N labels with the top _ score; the predict_label function encapsulates the test and prediction process and returns the prediction results. Finally, the audio file of the specified path is used to make the prediction and output the predicted label, as shown in the figure below:

```
def test_file(test_file, hmm_gmm):
 rate, sig = wvf.read(test_file)
 mfcc_feat = mfcc(sig, rate)
 pred = {}
 for model in hmm_gmm:
 pred[model] = hmm_gmm[model].score(mfcc_feat)
 return get_nbest(pred, 2), pred
#获取 topN 的结果
def get_nbest(d, n):

 return heapq.nlargest(n, d, key=lambda k: d[k])
def predict_label(file, hmm_gmm):

 predicted = test_file(file, hmm_gmm)
 return predicted
wave_path = os.path.join(test_data_path,"T0001.wav")
#wave_path = os.path.join(train_data_path,"S0001.wav")
predicted, probs = predict_label(wave_path,hmm_gmm)
print("PREDICTED: %s" % predicted[0])
```

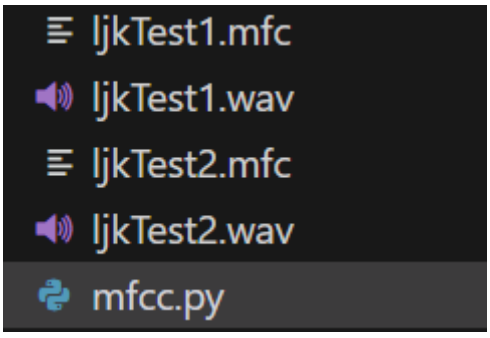


Output "关掉风扇二秒", which is the result of the model prediction, so it is likely to be the prediction label generated by the model based on the input test data.

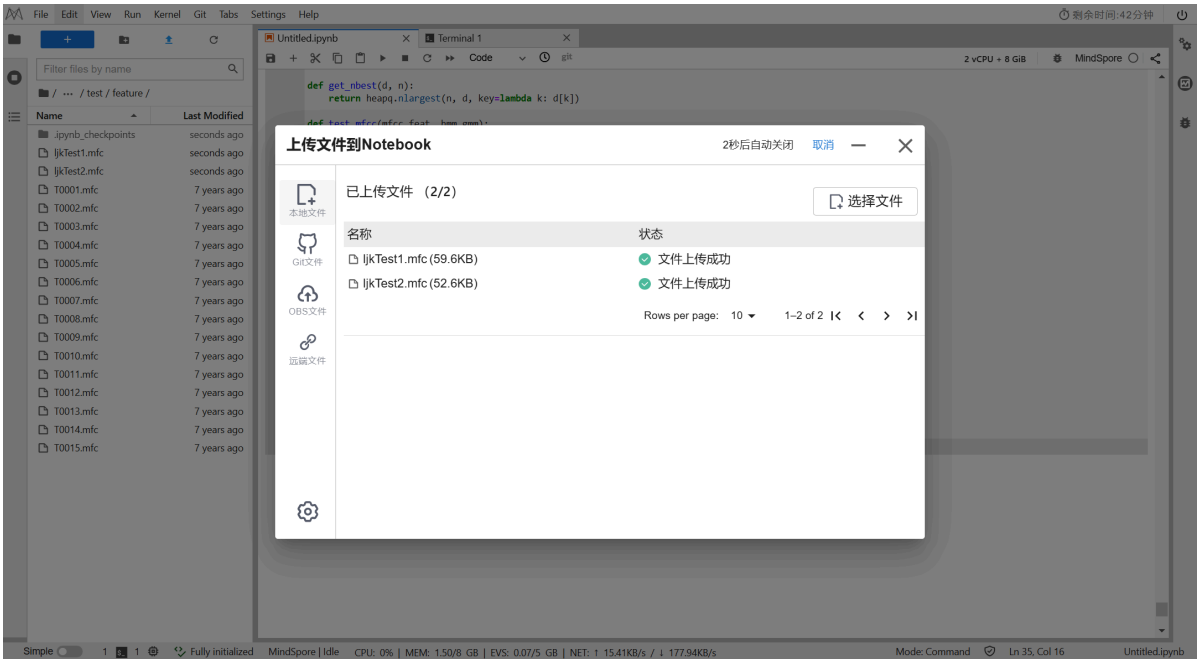
1.4. My Input Test-Related to Assignment 1

Next I need to use the code I wrote in job 1 to generate MFCC to generate the extraction value MFCC for two audio files and apply it to this code to show the prediction result. I recorded two audio clips myself, the first was "关掉阀门六小时" and the second was "打开风扇八分钟".

First I use the code of my first assignment to generate two audio of MFCC:



Then the two files are uploaded to the platform, as shown in the figure below:

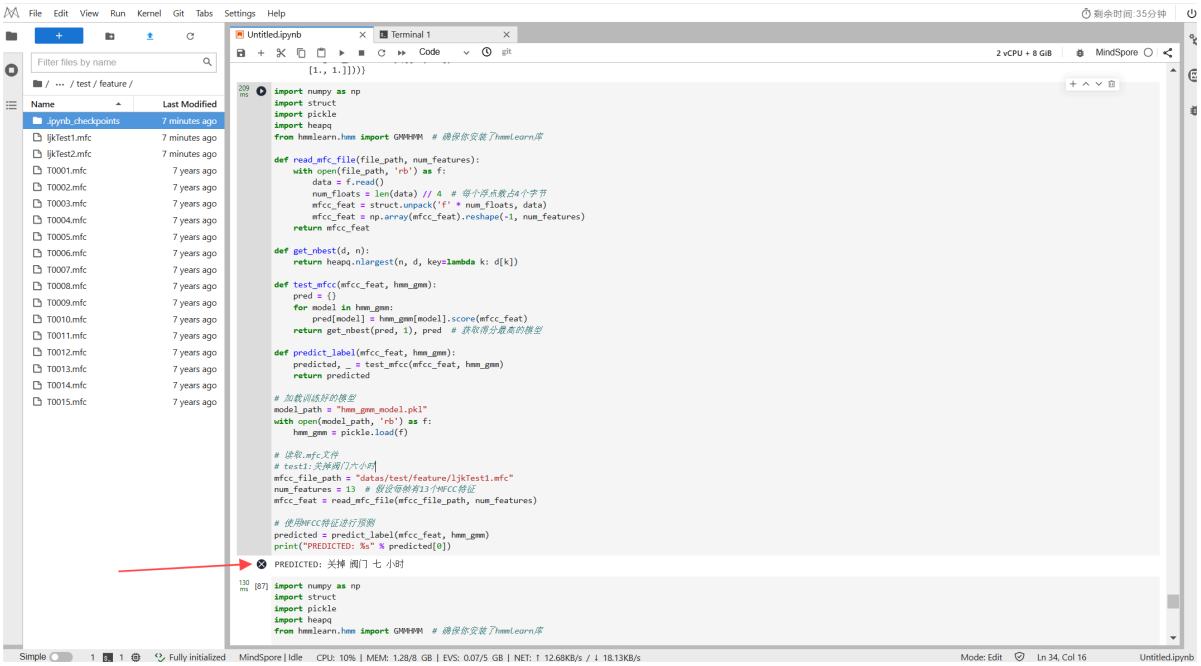


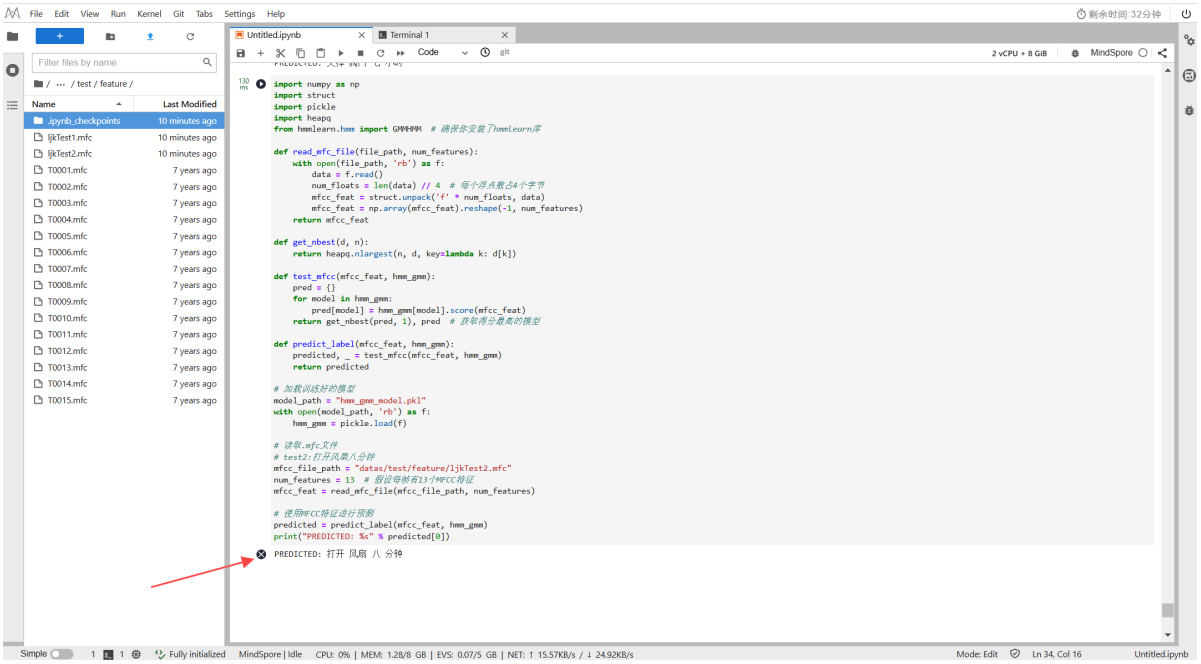
Predictive output of the model was performed using the following code:

```
import numpy as np
import struct
import pickle
import heapq
from hmmlearn.hmm import GMMHMM

def read_mfc_file(file_path, num_features):
    with open(file_path, 'rb') as f:
        data = f.read()
        num_floats = len(data) // 4  # 每个浮点数占4个字节
        mfcc_feat = struct.unpack('f' * num_floats, data)
        mfcc_feat = np.array(mfcc_feat).reshape(-1, num_features)
    return mfcc_feat

def get_nbest(d, n):
    return heapq.nlargest(n, d, key=lambda k: d[k])
```

```python
def test_mfcc(mfcc_feat, hmm_gmm):
    pred = {}
    for model in hmm_gmm:
        pred[model] = hmm_gmm[model].score(mfcc_feat)
    return get_nbest(pred, 1), pred  # 获取得分最高的模型

def predict_label(mfcc_feat, hmm_gmm):
    predicted, _ = test_mfcc(mfcc_feat, hmm_gmm)
    return predicted

# 加载训练好的模型
model_path = "hmm_gmm_model.pkl"
with open(model_path, 'rb') as f:
    hmm_gmm = pickle.load(f)

# 读取.mfc文件
# test1:关掉阀门六小时
mfcc_file_path = "datas/test/feature/ljkTest1.mfc"
num_features = 13
mfcc_feat = read_mfc_file(mfcc_file_path, num_features)

# 读取.mfc文件
# test2:打开风扇八分钟
# mfcc_file_path = "datas/test/feature/ljkTest2.mfc"
# num_features = 13
# mfcc_feat = read_mfc_file(mfcc_file_path, num_features)

predicted = predict_label(mfcc_feat, hmm_gmm)
print("PREDICTED: %s" % predicted[0])
```

The output of the last two predictions is as follows:





Experimental result:

I found that the prediction of the model still had some accuracy, and the MFCC output of my first work also successfully made the model identify the text. For the first test case, it identifies "关掉阀门六小时" as "关掉阀门七小时"; for the second test case, it can accurately identify "打开风扇八分钟".

Analysis:

- I think there are two reasons for the slight difference in the first test case identification. First of all, according to the assistant, the identification label statement is discrete not continuous, there may be "关掉阀门五小时" and "关掉阀门七小时", the label is not "关掉阀门六小时", which may be the reason. Second, the details of my MFCC extraction function, such as frame size, window function, Mel filter number and size Settings, and some normalized processing to be further improved.
- For both test cases, the results, I think because these steps cover many important links in speech feature extraction, involving multi-level analysis of time domain and frequency domain. At the same time, by extracting the dynamic features, not only retain the static frequency features, but also capture the dynamic changes in the time series, so my final MFCC has achieved good results.

The above is the verification and identification results of MFCC extracted from my first assignment.

2. Estmate the parameters of mean μ in a multivariate Gaussian model

2.1. Problem Description

Using Maximum Likelihood Estimation method, estmate the parameters of mean $\boldsymbol{\mu}$ in a multivariate Gaussian model given a set of sampled data $\mathbf{X} = \{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}\}$.

The pdf of the multivariate Gaussian model is $p(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})}$

2.2. Solution

We notice that:

- For each $\mathbf{x_i}$, $\mathbf{x_i} = (x_1, x_2, \ldots, x_D)^T \in R^D$, is a D-dimensional vector;
- $\boldsymbol{\mu} = (\mu_1, \mu_2, \ldots, \mu_D)^T \in R^D$, is the mean vector;
- $\boldsymbol{\Sigma} \in R^{D \times D}$ is the covariance matrix, which is a diagonal matrix.

Suppose we have a set of sampled data $\mathbf{X} = \{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}\}$, each sample $\mathbf{x_i}$ follows the same multivariate Gaussian distribution. Then the likelihood function is the product of the joint probabilities for all sample points:

$$L(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \prod_{i=1}^{n} p(\mathbf{x_i} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$$

By substituting the probability density function of the multivariate Gaussian distribution, it yields:

$$L(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \prod_{i=1}^{n} \frac{1}{(2\pi)^{D/2} |\boldsymbol{\Sigma}|^{1/2}} e^{-\frac{1}{2}(\mathbf{x_i} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x_i} - \boldsymbol{\mu})}$$

To simplify the calculation, the log likelihood function:

$$\ln L(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = -\frac{n}{2} \ln |\boldsymbol{\Sigma}| - \frac{nD}{2} \ln(2\pi) - \frac{1}{2} \sum_{i=1}^{n} (\mathbf{x_i} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x_i} - \boldsymbol{\mu})$$

We find that the first two terms were not related to the $\boldsymbol{\mu}$.Next, we derive this function $\ln L(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ from the pair $\boldsymbol{\mu}$:

$$\frac{\partial \ln L(\boldsymbol{\mu}, \boldsymbol{\Sigma})}{\partial \boldsymbol{\mu}} = -\frac{1}{2} \frac{\partial}{\partial \boldsymbol{\mu}} \sum_{i=1}^{n} (\mathbf{x_i} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x_i} - \boldsymbol{\mu})$$

Calculate:

$$(\mathbf{x_i} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x_i} - \boldsymbol{\mu}) = (\mathbf{x_i}^T - \boldsymbol{\mu}^T) \Sigma^{-1} (\mathbf{x_i} - \boldsymbol{\mu}) = \mathbf{x_i}^T \boldsymbol{\Sigma}^{-1} \mathbf{x_i} - 2\boldsymbol{\mu}^T \boldsymbol{\Sigma}^{-1} \mathbf{x_i} + \boldsymbol{\mu}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}$$

So, we can get:

$$\frac{\partial \ln L(\boldsymbol{\mu}, \boldsymbol{\Sigma})}{\partial \boldsymbol{\mu}} = -\frac{1}{2} \frac{\partial}{\partial \boldsymbol{\mu}} \sum_{i=1}^{n} (\mathbf{x_i} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x_i} - \boldsymbol{\mu}) = -\frac{1}{2} \frac{\partial}{\partial \boldsymbol{\mu}} \sum_{i=1}^{n} (-2\boldsymbol{\mu}^T \boldsymbol{\Sigma}^{-1} \mathbf{x_i} + \boldsymbol{\mu}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu})$$

We know that:

$$\frac{\partial}{\partial \boldsymbol{\mu}} \left(-2\boldsymbol{\mu}^T \boldsymbol{\Sigma}^{-1} \mathbf{x_i}\right) = \left(-2\boldsymbol{\Sigma}^{-1} \mathbf{x_i}\right) = -2\boldsymbol{\Sigma}^{-1} \mathbf{x_i}$$

$$\frac{\partial}{\partial \boldsymbol{\mu}} \left(\boldsymbol{\mu}^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}\right) = 2\boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}$$

So, we can get:

$$\frac{\partial \ln L(\boldsymbol{\mu})}{\partial \boldsymbol{\mu}} = \sum_{i=1}^{n} \boldsymbol{\Sigma}^{-1} \mathbf{x_i} - \sum_{i=1}^{n} \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}$$

We derive the mean parameter $\boldsymbol{\mu}$ and set the result to zero to find the maximum point. Finding the partial derivatives for $\boldsymbol{\mu}$:

$$\frac{\partial \ln L(\boldsymbol{\mu})}{\partial \boldsymbol{\mu}} = \sum_{i=1}^{n} \boldsymbol{\Sigma}^{-1} \mathbf{x_i} - \sum_{i=1}^{n} \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu} = \boldsymbol{\Sigma}^{-1} (\sum_{i=1}^{n} \mathbf{x_i} - \sum_{i=1}^{n} \boldsymbol{\mu}) = 0$$

Owing to $\boldsymbol{\Sigma}^{-1}$ is a nonsingular matrix and can be eliminated:

$$\sum_{i=1}^{n} \mathbf{x_i} - \sum_{i=1}^{n} \boldsymbol{\mu} = \sum_{i=1}^{n} \mathbf{x_i} - n\boldsymbol{\mu} = 0$$

After finishing the above form:

$$\boldsymbol{\mu} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x_i}$$

Therefore, the maximum likelihood estimate of the parameter μ is the mean of the sample:

$$\hat{\boldsymbol{\mu}} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x_i}$$