SPEECH ANALYSIS

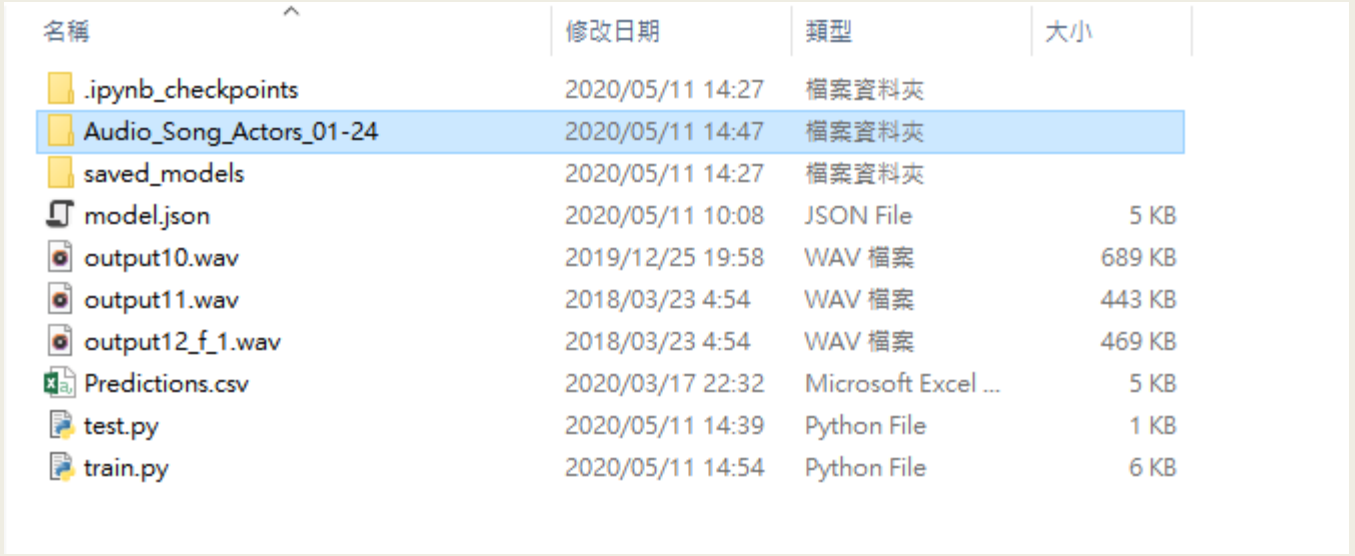
版本與套件

- **Python 3.6.8**
- pip install librosa==0.7.2 #用於進行資料前處理
- pip install Tensorflow>=2 #用於模型建構
- pip install pandas #用於資料格式整理，作為Sklearn或Tensorflow的輸入
- pip install matplotlib #用於畫圖

檔案介紹

https://drive.google.com/file/d/1mwoKSUO3xnNtlOYnMbR6guW2fgtjdZ8o/view?usp=sharing

- Audio_Song_Actors – 是資料集
- Saved – 儲存模型權重的地方
- Model.json -訓練好的模型
- Test.py – 測試程式
- Train.py – 訓練程式



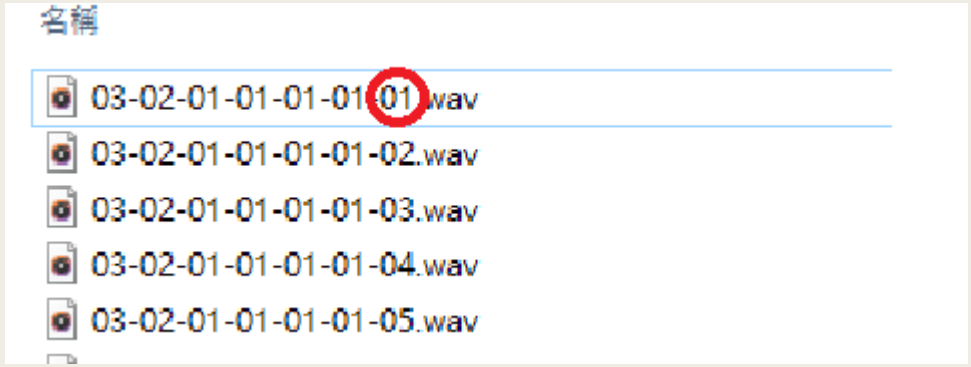
RAVDESS資料集

- 共860筆資料
- 由 11男 9女組成
- 分別說了43句話

音檔性別命名

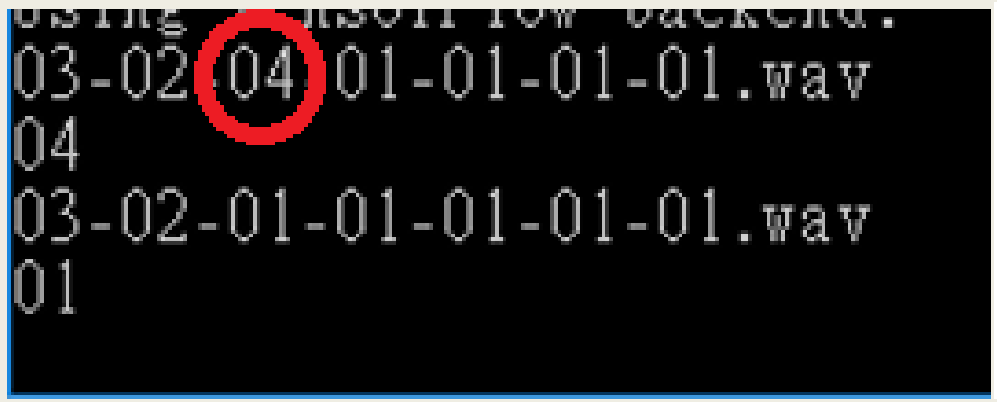
由倒數兩個數字區分男女
單數為男,偶數為女
Ex ：03-02-01-01-01-01-01後兩碼為01 故為男
Ex 2 : 03-02-02-02-02-02-14後兩碼為14 故為女
不同數字則為不同男聲或女聲

```python
import os
mylist= os.listdir('Audio_Song_Actors_01-24/')
# 讀入音檔
print(mylist[400][18:-4])
# 前處理設立個音訊檔對應的label
# 建立出其音訊對應的結果
feeling_list=[]
for item in mylist:
    if item[18:-4]=='01':
        feeling_list.append('male')
    elif item[18:-4]=='02':
        feeling_list.append('female')
    elif item[18:-4]=='03':
        feeling_list.append('male')
    elif item[18:-4]=='04':
        feeling_list.append('female')
```



音檔情緒命名

由第三段作為標準
01 – neutral 各有4句
02 – calm　各有8句
03 – happy　各有8句
04 – sad　各有8句
05 – angry　各有8句
06 – fearful　各有7句



Ex : 03-02-04-01-01-01-01
為 傷心之聲音情緒
Ex : 03-02-01-01-01-01-01
為 正常之聲音情緒

載入資料

```
import os
mylist= os.listdir('Audio_Song_Actors_01-24/')
# 讀入音檔
print(mylist[400][18:-4])
# 前處理設立個音訊檔對應的label
# 建立出其音訊對應的結果
feeling_list=[]
```

對資料進行標記

建立一個陣列,將標記好之資料label放入此陣列形成一個[male,female,male,female,.....female]之陣列

```
feeling_list=[]
for item in mylist:
    if item[18:-4]=='01':
        feeling_list.append('male')
    elif item[18:-4]=='02':
        feeling_list.append('female')
    elif item[18:-4]=='03':
        feeling_list.append('male')
    elif item[18:-4]=='04':
        feeling_list.append('female')
    elif item[18:-4]=='05':
```

資料前處理

- 前處理上，聲音是由一串不固定頻率之波譜組成的而其中每個區段的又可能是由不同頻率的小波譜疊加而成ㄉ而從時域上是無法發現各類頻率的，所以聲音資料通常都需要藉由復立葉轉換將其轉變為頻域進行分析
- 通常是透過librosa套件中的函數進行特徵擷取 https://librosa.github.io/librosa/
- 常用的librosa函數mfcc()與melspectrogram()
- mfcc :使用於類人聽覺　　melspectrogram :使用於聲音識別

加入特徵

- 建立df為一個columns名為feature的陣列
- Index – 索引　y-檔名
- X – 音檔之時域圖(陣列)
- Sample_rate – 我們取的sr也就是多少毫秒取一次
- np.mean()算出指定軸之平均值
- Mfcc()將剛剛取得之時域陣列轉換為頻域
- df.loc()=>將轉換後的feature放入我們上面設計的Columens中

```
labels = pd.DataFrame(feeling_list)

# Getting the features of audio files using librosa¶(使用librosa去處理聲音特徵)
df = pd.DataFrame(columns=['feature'])
bookmark=0

# 使用mfcc特徵擷取好處 可以在多個領域中使聲音訊號有更好的表示 它比用於正常的對數
for index,y in enumerate(mylist):
    X, sample_rate = librosa.load('Audio_Song_Actors_01-24/'+y, res_type='kaiser
    print(X)
    print(sample_rate)
    sample_rate = np.array(sample_rate)
    mfccs = np.mean(librosa.feature.mfcc(y=X,
                                         sr=sample_rate),
                    axis=0)
    feature = mfccs
    #[float(i) for i in feature]
    #feature1=feature[:135]
    df.loc[bookmark] = [feature]
    bookmark=bookmark+1
```

```
Using TensorFlow backend.
01
[ 0.0000000e+00  0.0000000e+00  0.0000000e+00 ... -1.3243372e-04
 -9.5149648e-05 -1.4267711e-04]
44100
[ 2.8406583e-07 -4.4985632e-07  5.9294553e-07 ... -1.5544590e-02
 -2.1059537e-02 -2.1686770e-02]
44100
[0.00042499 0.00040208 0.00018122 ... 0.02675534 0.03054328 0.02739273]
44100
[ 0.          0.          0.         ... -0.00470676 -0.00529516
 -0.00617851]
44100
[-5.4674099e-05 -6.4911655e-05 -5.9010814e-05 ...  7.0044323e-04
  8.8420341e-04  7.9041981e-04]
44100
```

針對librosa介紹
讀取音檔

- 常用變數
- sr – 以多少毫秒去切割特徵
- Duration – 只讀這些量的音訊
- Offset - 從此時段開始讀取
- res_type –採樣類型

```
librosa.load('Audio_Song_Actors_01-24/'+y, res_type='kaiser_fast',duration=2.5,sr=22050*2,offset=0.5)
.array(sample_rate)
```

擷取特徵量
MFCC()

```
mfccs = np.mean(librosa.feature.mfcc(y=X,
                                     sr=sample_rate,
                                     n_mfcc=13),
                axis=0)
```

y - 音頻時間序列
Sr – 多少時間採樣一次
N_mfcc - 更改mfcc回傳數(跟特徵量有關

轉換資料格式 tolist()

- 將量變成list型式

```
print(df)
df3 = pd.DataFrame(df['feature'].values.tolist())
print(df3)
# # ndarray.tolist() 對ndarry做遞歸降維，1維則不變
```

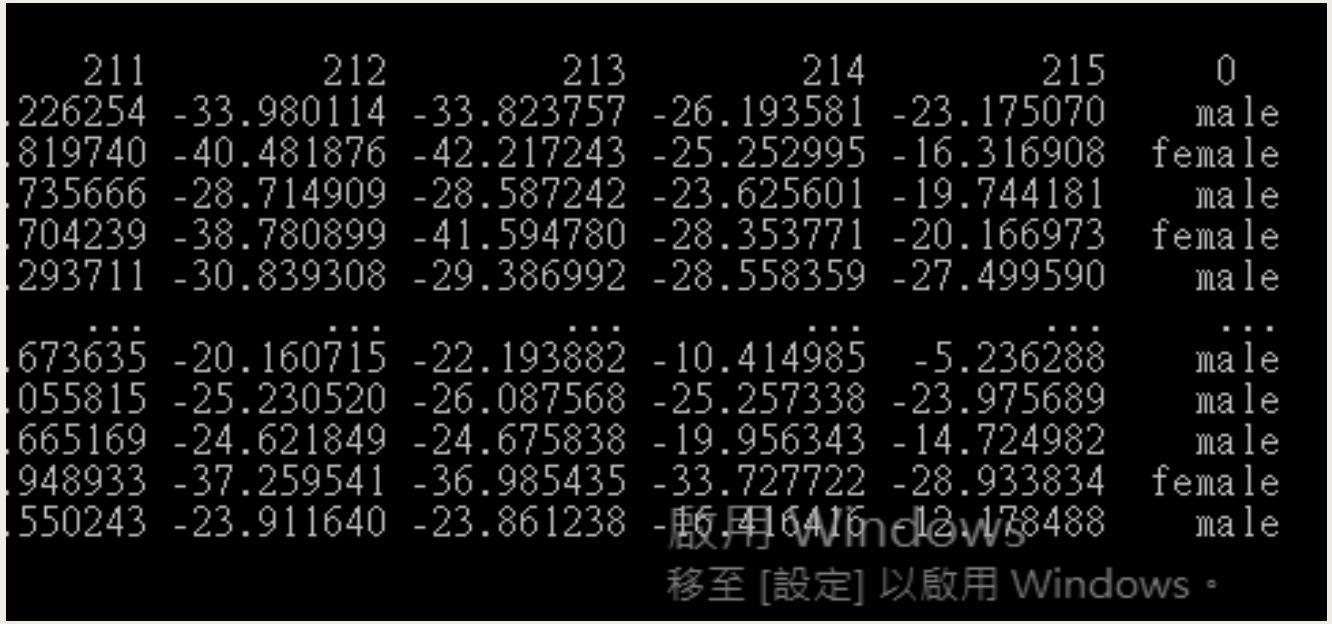
```
                                                 feature
0    [-61.437893, -61.437893, -61.437893, -61.43789...
1    [-56.998222, -57.33978, -57.16586, -56.601364,...
2    [-52.929504, -55.759624, -55.591812, -55.64324...
3    [-59.526695, -59.526695, -59.526695, -59.52669...
4    [-58.03214, -59.64904, -61.456345, -56.858982,...
..                                                 ...
855  [-31.16216, -32.271633, -35.222115, -34.010807...
856  [-51.899754, -52.792828, -54.92356, -53.1228, ...
857  [-53.04238, -53.04238, -51.16644, -49.679585, ...
858  [-49.082905, -48.570633, -48.614326, -49.49452...
859  [-51.863243, -51.863243, -51.863243, -51.86324...

[860 rows x 1 columns]
            0           1           2           3           4
0   -61.437893 -61.437893 -61.437893 -61.437893 -61.4378
1   -56.998222 -57.339779 -57.165859 -56.601364 -56.2314
2   -52.929504 -55.759624 -55.591812 -55.643246 -55.9291
3   -59.526695 -59.526695 -59.526695 -59.526695 -59.5266
4   -58.032139 -59.649040 -61.456345 -56.858982 -55.7887
..         ...        ...        ...        ...      ...
855 -31.162161 -32.271633 -35.222115 -34.010807 -35.5389
856 -51.899754 -52.792828 -54.923561 -53.122799 -53.0520
857 -53.042381 -53.042381 -51.166439 -49.679585 -50.0174
858 -49.082905 -48.570633 -48.614326 -49.494526 -49.7921
859 -51.863243 -51.863243 -51.863243 -51.863243 -51.8632

[860 rows x 216 columns]
```

newdf = pd.concat([df3,labels], axis=1)

- 將前面的labels陣列加入到剛剛變換的df3陣列



211	212	213	214	215	0
.226254	-33.980114	-33.823757	-26.193581	-23.175070	male
.819740	-40.481876	-42.217243	-25.252995	-16.316908	female
.735666	-28.714909	-28.587242	-23.625601	-19.744181	male
.704239	-38.780899	-41.594780	-28.353771	-20.166973	female
.293711	-30.839308	-29.386992	-28.558359	-27.499590	male
...	...	...	...	...	...
.673635	-20.160715	-22.193882	-10.414985	-5.236288	male
.055815	-25.230520	-26.087568	-25.257338	-23.975689	male
.665169	-24.621849	-24.675838	-19.956343	-14.724982	male
.948933	-37.259541	-36.985435	-33.727722	-28.933834	female
.550243	-23.911640	-23.861238	-16.516416	-13.178488	male

Shuffle()
打亂陣列

```
            0           1
0   -61.437893 -61.437893 -61
1   -56.998222 -57.339779 -57
2   -52.929504 -55.759624 -55
3   -59.526695 -59.526695 -59
4   -58.032139 -59.649040 -61
..         ...        ...
855 -31.162161 -32.271633 -35
856 -51.899754 -52.792828 -54
857 -53.042381 -53.042381 -51
858 -49.082905 -48.570633 -48
859 -51.863243 -51.863243 -51

[860 rows x 217 columns]
            0           1
637 -48.939556 -48.939556 -48
492 -57.322628 -56.360298 -55
25  -54.059284 -52.502888 -51
20  -59.768211 -59.768211 -59
809 -43.038723 -44.091232 -42
..         ...        ...
717 -43.498188 -43.674004 -43
105 -61.479057 -62.185814 -62
610 -53.458023 -53.458023 -53
51  -61.844883 -62.111942 -58
615 -39.207333 -41.463760 -44
```

將資料集分為訓練與測試

```
# Dividing the data into test and train
newdf1 = np.random.rand(len(rnewdf)) < 0.8
train = rnewdf[newdf1]
test = rnewdf[~newdf1]
print(train)
print(test)
```

分別找出label與向量值

- Trainfeatures - 特徵
- Trainlabel – 標籤

```
print(train)
trainfeatures = train.iloc[:, :-1]
trainlabel = train.iloc[:, -1:]
testfeatures = test.iloc[:, :-1]
testlabel = test.iloc[:, -1:]

print(trainfeatures)
print(trainlabel)
```

建立二維陣列

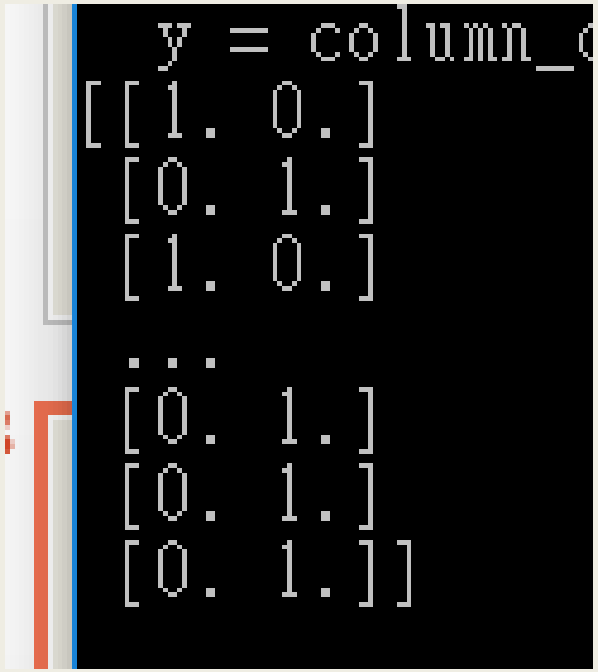
```
from keras.utils import np_utils
from sklearn.preprocessing import LabelEncoder

X_train = np.array(trainfeatures)
y_train = np.array(trainlabel)
X_test = np.array(testfeatures)
y_test = np.array(testlabel)

print(X_train)
print(y_train)
```

```
Using TensorFlow backend.
01
[[-54.41120911 -54.51576996 -53.2771759  ... -31.1500473  -23.994133
  -18.8252182 ]
 [-54.13692856 -53.85390091 -53.04828644 ... -34.56499863 -22.20160866
  -16.0321331 ]
 [-53.31479645 -53.31479645 -53.31479645 ... -45.695858   -29.19203377
  -21.27311134]
 ...
 [-51.18098831 -48.52749252 -48.34543228 ... -26.84209251 -24.02609444
  -21.45054054]
 [-55.34756088 -54.89084625 -54.76285934 ... -31.96567535 -23.52489281
  -18.50447464]
 [-53.0283165  -53.0283165  -53.0283165  ... -19.67622566 -17.46390915
  -12.82906246]]
[['male']
 ['female']
 ['female']
 ['male']
 ['female']
 ['male']
 ['male']
 ['female']
 ['female']
 ['female']
 ['male']
 ['female']
 ['female']
```

將標籤改為[0. 1.]為男 [1. 0.]為女
one hot encoder




```
lb = LabelEncoder()
print(y_train)
y_train = np_utils.to_categorical(lb.fit_transform(y_train))
y_test = np_utils.to_categorical(lb.fit_transform(y_test))
print(y_train)
```

模型構成

- CNN模型
- CNN --https://brohrer.mcknote.com/zh-Hant/how_machine_learning_works/how_convolutional_neural_networks_work.html
- 卷積層 4層
- 激活函數 選用RELU 解決梯度消失 增加收斂速度
- Dropout(0.1) 正規化

訓練結果

- 執行train.py進行模型訓練 會產生一個model.json與.h5的模型權重檔

```
model_name = 'Emotion_Voice_Detection_Model.h5'
save_dir = os.path.join(os.getcwd(), 'saved_models')
# Save model and weights

if not os.path.isdir(save_dir):
    os.makedirs(save_dir)
model_path = os.path.join(save_dir, model_name)
model.save(model_path)
print('Saved trained model at %s ' % model_path)
import json
model_json = model.to_json()
with open("model.json", "w") as json_file:
    json_file.write(model_json)
```

測試方法

- 執行test.py將訓練後的模型讀入並丟入一個音檔進行預測

```python
from tensorflow import keras
from tensorflow.keras.models import model_from_json
import librosa
import numpy as np
import pandas as pd

json_file = open('model.json', 'r')
# 讀入模型
loaded_model_json = json_file.read()
json_file.close()
loaded_model = model_from_json(loaded_model_json)
# load weights into new model
loaded_model.load_weights("saved_models/Voice_Detection_Model.h5")
# 讀入先前訓練模型的權重
print("Loaded model from disk")


# 輸入音訊預測
def classify(pretrain, file):
    X, sample_rate = librosa.load(file, res_type='kaiser_fast',duration=2.5,sr=22050*2,offset=0.5)
    sample_rate = np.array(sample_rate)
    mfccs = np.mean(librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=13),axis=0)
    featurelive = mfccs
    livedf = featurelive
    livedf= pd.DataFrame(data=livedf)
    livedf = livedf.stack().to_frame().T
    twodim= np.expand_dims(livedf, axis=2)
    livepreds = pretrain.predict(twodim, batch_size=32, verbose=1)
    return livepreds

print(classify(loaded_model, '0.wav'))
print("---")
print(classify(loaded_model, '1.wav'))
```

神經網路觀念

- COURSERA --https://www.coursera.org/learn/machine-learning?