```
#!/usr/bin/env python
# coding: utf-8

# # More To Come. Stay Tuned. !!
# If there are any suggestions/changes you would like to
see in the Kernel please let me know :). Appreciate
every ounce of help!
#
# **This notebook will always be a work in progress**.
Please leave any comments about further improvements to
the notebook! Any feedback or constructive criticism is
greatly appreciated!. **If you like it or it helps you ,
you can upvote and/or leave a comment :).**|
#

# In[1]:


import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O
(e.g. pd.read_csv)
import matplotlib.pyplot as plt
import seaborn as sns
get_ipython().magic('matplotlib inline')

import IPython.display as ipd  # To play sound in the
notebook
from tqdm import tqdm_notebook
import wave
from scipy.io import wavfile
SAMPLE_RATE = 44100

import seaborn as sns # for making plots with seaborn
color = sns.color_palette()
import plotly.offline as py
py.init_notebook_mode(connected=True)
import plotly.graph_objs as go
import plotly.offline as offline
offline.init_notebook_mode()
import plotly.tools as tls
# Math
import numpy as np
from scipy.fftpack import fft
from scipy import signal
from scipy.io import wavfile
import librosa
```

```
# In[2]:


import os
print(os.listdir("../input"))


# In[3]:


INPUT_LIB = '../input/'
audio_train_files = os.listdir('../input/audio_train')
audio_test_files = os.listdir('../input/audio_test')
train = pd.read_csv('../input/train.csv')
submission = pd.read_csv("../input/
sample_submission.csv", index_col='fname')
train_audio_path = '../input/audio_train/'
filename = '/001ca53d.wav' # Hi-hat
sample_rate, samples =
wavfile.read(str(train_audio_path) + filename)
#sample_rate = 16000


# In[4]:


print(samples)


# In[5]:


print("Size of training data",train.shape)


# In[6]:


train.head()


# In[7]:
```

```
submission.head()


# In[8]:


def clean_filename(fname, string):
    file_name = fname.split('/')[1]
    if file_name[:2] == '__':
        file_name = string + file_name
    return file_name

def load_wav_file(name, path):
    _, b = wavfile.read(path + name)
    assert _ == SAMPLE_RATE
    return b


# In[9]:


train_data = pd.DataFrame({'file_name' : train['fname'],
                          'target' : train['label']})
train_data['time_series'] =
train_data['file_name'].apply(load_wav_file,

path=INPUT_LIB + 'audio_train/')
train_data['nframes'] =
train_data['time_series'].apply(len)


# In[10]:


train_data.head()


# In[11]:


print("Size of training data after some preprocessing :
",train_data.shape)


# In[12]:
```

```
# missing data in training data set
total = train_data.isnull().sum().sort_values(ascending
= False)
percent = (train_data.isnull().sum()/
train_data.isnull().count()).sort_values(ascending =
False)
missing_train_data = pd.concat([total, percent], axis=1,
keys=['Total', 'Percent'])
missing_train_data.head()


# There is no missing data in training dataset

# # Manually verified Audio

# In[17]:


temp = train['manually_verified'].value_counts()
labels = temp.index
sizes = (temp / temp.sum())*100
trace = go.Pie(labels=labels, values=sizes,
hoverinfo='label+percent')
layout = go.Layout(title='Manually varification of
labels(0 - No, 1 - Yes)')
data = [trace]
fig = go.Figure(data=data, layout=layout)
py.iplot(fig)


# * Approximately 40 % labels are manually varified.

# In[18]:


plt.figure(figsize=(12,8))
sns.distplot(train_data.nframes.values, bins=50,
kde=False)
plt.xlabel('nframes', fontsize=12)
plt.title("Histogram of #frames")
plt.show()


# In[19]:
```

```
plt.figure(figsize=(17,8))
boxplot = sns.boxplot(x="target", y="nframes",
data=train_data)
boxplot.set(xlabel='', ylabel='')
plt.title('Distribution of audio frames, per label',
fontsize=17)
plt.xticks(rotation=80, fontsize=17)
plt.yticks(fontsize=17)
plt.xlabel('Label name')
plt.ylabel('nframes')
plt.show()


# In[20]:


print("Total number of labels in training data :
",len(train_data['target'].value_counts()))
print("Labels are : ", train_data['target'].unique())
plt.figure(figsize=(15,8))
audio_type = train_data['target'].value_counts().head(30)
sns.barplot(audio_type.values, audio_type.index)
for i, v in enumerate(audio_type.values):
    plt.text(0.8,i,v,color='k',fontsize=12)
plt.xticks(rotation='vertical')
plt.xlabel('Frequency')
plt.ylabel('Label Name')
plt.title("Top 30 labels with their frequencies in
training data")
plt.show()


# ### Total number of labels are 41

# In[ ]:


temp = train_data.sort_values(by='target')
temp.head()


# ## Now look at  some labels waveform :
#   1. Acoustic_guitar
#   2. Applause
#   3. Bark
```

```
# ## 1. Acoustic_guitar

# In[ ]:


print("Acoustic_guitar : ")
fig, ax = plt.subplots(10, 4, figsize = (12, 16))
for i in range(40):
    ax[i//4, i%4].plot(temp['time_series'][i])
    ax[i//4, i%4].set_title(temp['file_name'][i][:-4])
    ax[i//4, i%4].get_xaxis().set_ticks([])
fig.savefig("AudioWaveform", dpi=900)


# ## 2. Applause

# In[ ]:


print("Applause : ")
fig, ax = plt.subplots(10, 4, figsize = (12, 16))
for i in range(40):
    ax[i//4, i%4].plot(temp['time_series'][i+300])
    ax[i//4, i%4].set_title(temp['file_name'][i+300]
[:-4])
    ax[i//4, i%4].get_xaxis().set_ticks([])


# ## 3. Bark

# In[ ]:


print("Bark : ")
fig, ax = plt.subplots(10, 4, figsize = (12, 16))
for i in range(40):
    ax[i//4, i%4].plot(temp['time_series'][i+600])
    ax[i//4, i%4].set_title(temp['file_name'][i+600]
[:-4])
    ax[i//4, i%4].get_xaxis().set_ticks([])


# In[ ]:
```

```
from wordcloud import WordCloud
wordcloud = WordCloud(max_font_size=50, width=600,
height=300).generate(' '.join(train_data.target))
plt.figure(figsize=(15,8))
plt.imshow(wordcloud)
plt.title("Wordcloud for Labels", fontsize=35)
plt.axis("off")
plt.show()
#fig.savefig("LabelsWordCloud", dpi=900)


# # Spectrogram

# In[ ]:


def log_specgram(audio, sample_rate, window_size=20,
                 step_size=10, eps=1e-10):
    nperseg = int(round(window_size * sample_rate / 1e3))
    noverlap = int(round(step_size * sample_rate / 1e3))
    freqs, times, spec = signal.spectrogram(audio,
                                    fs=sample_rate,
                                    window='hann',
                                    nperseg=nperseg,
                                    noverlap=noverlap,
                                    detrend=False)
    return freqs, times,
np.log(spec.T.astype(np.float32) + eps)


# In[ ]:


freqs, times, spectrogram = log_specgram(samples,
sample_rate)

fig = plt.figure(figsize=(18, 8))
ax2 = fig.add_subplot(211)
ax2.imshow(spectrogram.T, aspect='auto', origin='lower',
           extent=[times.min(), times.max(),
freqs.min(), freqs.max()])
ax2.set_yticks(freqs[::40])
ax2.set_xticks(times[::40])
ax2.set_title('Spectrogram of Hi-hat ' + filename)
ax2.set_ylabel('Freqs in Hz')
ax2.set_xlabel('Seconds')
```

```python
# # Specgtrogram of "Hi-Hat" in 3d

# If we use spectrogram as an input features for NN, we
have to remember to normalize features.

# In[ ]:


mean = np.mean(spectrogram, axis=0)
std = np.std(spectrogram, axis=0)
spectrogram = (spectrogram - mean) / std


# In[ ]:


data = [go.Surface(z=spectrogram.T)]
layout = go.Layout(
    title='Specgtrogram of "Hi-Hat" in 3d',
    scene = dict(
    yaxis = dict(title='Frequencies', range=freqs),
    xaxis = dict(title='Time', range=times),
    zaxis = dict(title='Log amplitude'),
    ),
)
fig = go.Figure(data=data, layout=layout)
py.iplot(fig)


# # More To Come. Stayed Tuned !!
```