MUSIC ANALYSIS SYSTEM

Product User Guide

Contents

1. Product Overview

1.1 Introduction

Music is such a language that sometimes you don't need words to express how you feel. It is a great reason why many people use it as a medium of therapy to find a way through the emotional challenges in their life. No matter whichever part of the world you live in, whatever language you speak, music finds a way to reach the hearts of the people across all cultures and religions. This is the reason why many music streaming services like JioSaavn and Apple optimise their content in such a way that it feels more personalised and connected to their users. To accomplish this task, they use certain tools to analyse different genres of music to predict the notes instruments used, beats duration, etc.

1.2 Product Objective

It is extremely necessary for these companies to analyse the audio and predict these parameters correctly without affecting their current resources. Latest studies show that Spotify adds more than 5000 hours of content to be released everyday. This means it's crucial to identify which type of music is most appreciated by people of which region or culture, to offer personalised services.

This product aims to use deep learning based algorithms written in python to predict the instruments used in a classical song with the highest possible accuracy.

1.3 Dataset Used

Enroute to design an instrument prediction algorithm, MusicNet audio collection has been used, which has 330 classical music files. It has over 1 million annotated labels indicating the precise time of each note in every recording, the instrument that plays each note, and the note's position in the metrical structure of the composition. These parameters can be used in both music recommendation and generation. Specifically, MusicNet labels aim to address the following tasks:

- Identify the *notes* performed at specific times in a recording.
- Classify the *instruments* that perform in a recording.
- Classify the *composer* of a recording.
- Identify precise *onset* times of the notes in a recording.
- Predict the *next note* in a recording, conditioned on history.

Considering the flexibility of the dataset, it was best suited to predict different types of instruments used in a classical song.

1.4 Database Management

Database activity can be tracked using a manifest file that contains the CRC values corresponding to each audio file. Data storage comes with a flexibility of location as well. All the audio files can be stored in a local machine or a hosted server with sufficient storage capacity.

1.5 Methodologies Used

This product has been divided into various segments, namely, Data Analysis, Data Pre-processing, Model Training, Model Testing. Most of the operations have been carried out in a python while the data analysis has been executed in a Linux environment using Bash Script. For model training, a well renowned deep learning based algorithm called LSTM has been used with an Attention Mechanism Architecture. You will learn about this concept in the further sections of this document in detail.

2. Setting Up The Environment

In this document, all the installation steps are demonstrated after running on Windows Operating System. For other operating systems, however, links have been provided to obtain all the setups and installation instructions.

2.1 System Requirements

Hardware:-
Processor: 1 GHz or faster CPU.
GPU: 8 GB VRAM (DirectX 9 or higher).
RAM: 2 GB or higher

Software:-
OS: Windows 7 or more.
Python 3.8 (or higher)
Tensorflow 2.4
Librosa
FFmpeg
MediaInfo

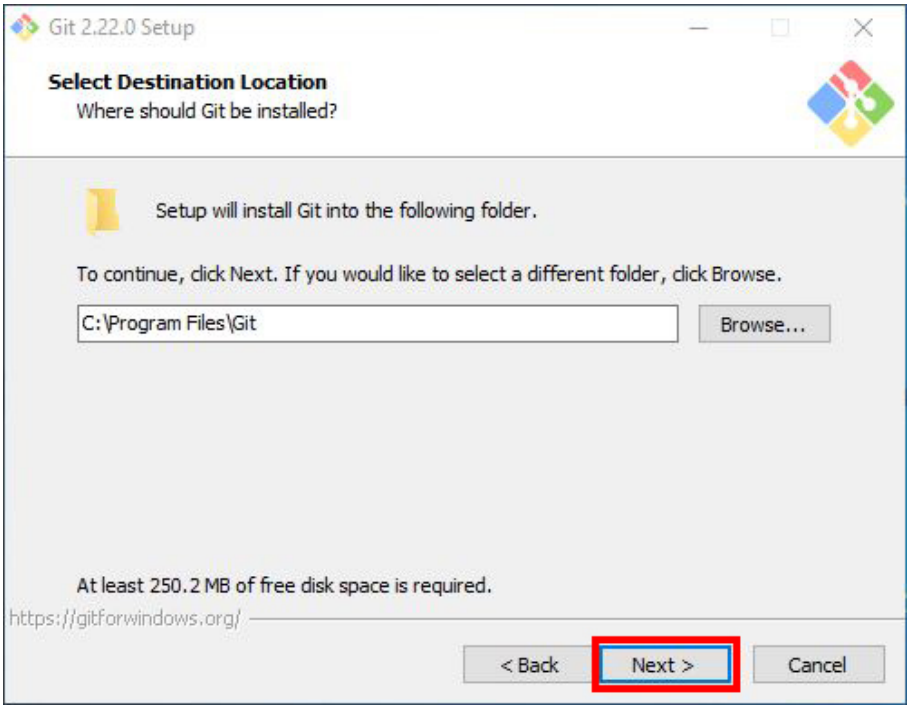
2.2 How to install Git?

You need to set up Git on your system to establish a Linux environment to run certain audio analysis commands.

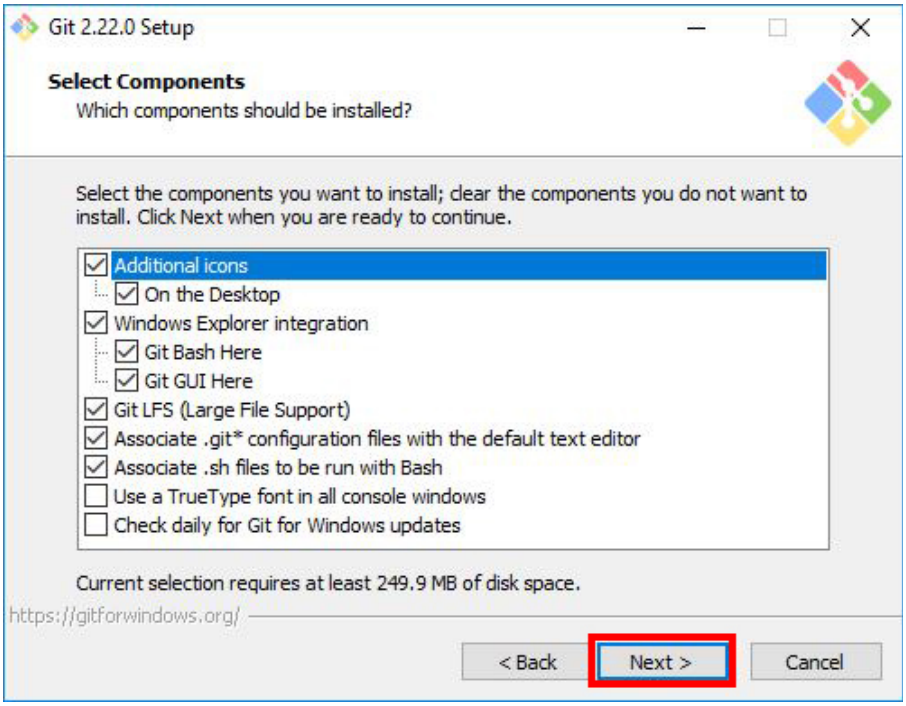
Step 1: Visit the official Git website → https://git-scm.com/ and select your OS version to start the download.

Step 2: After the setup file is downloaded, run the installer to extract Git.

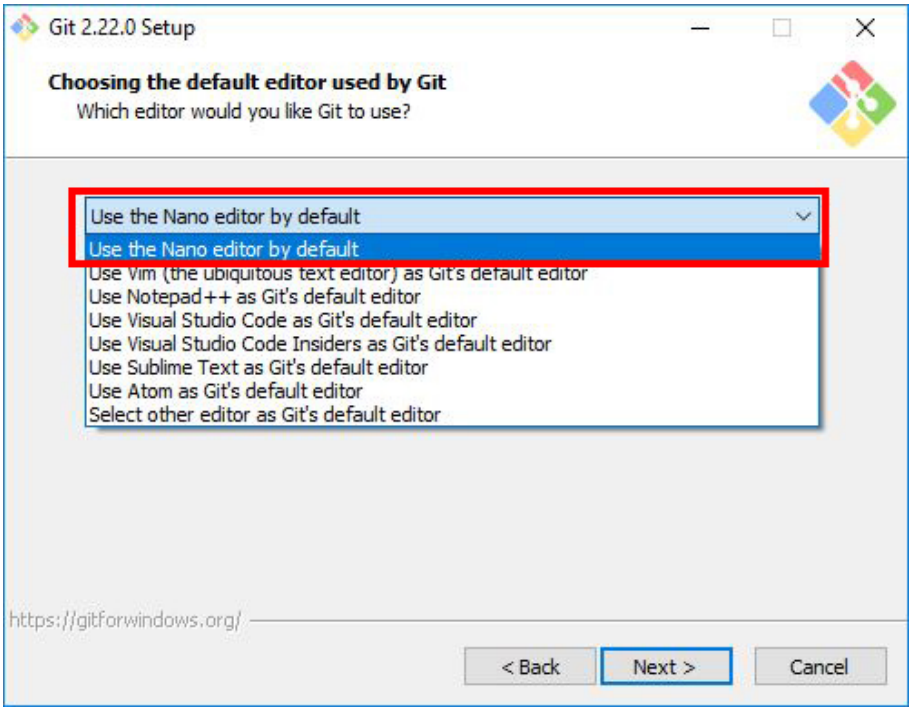
Step 3: Select the destination where you want to extract the files.



Step 4: You can proceed with the components selected by default (recommended) or you can select the components on your own.

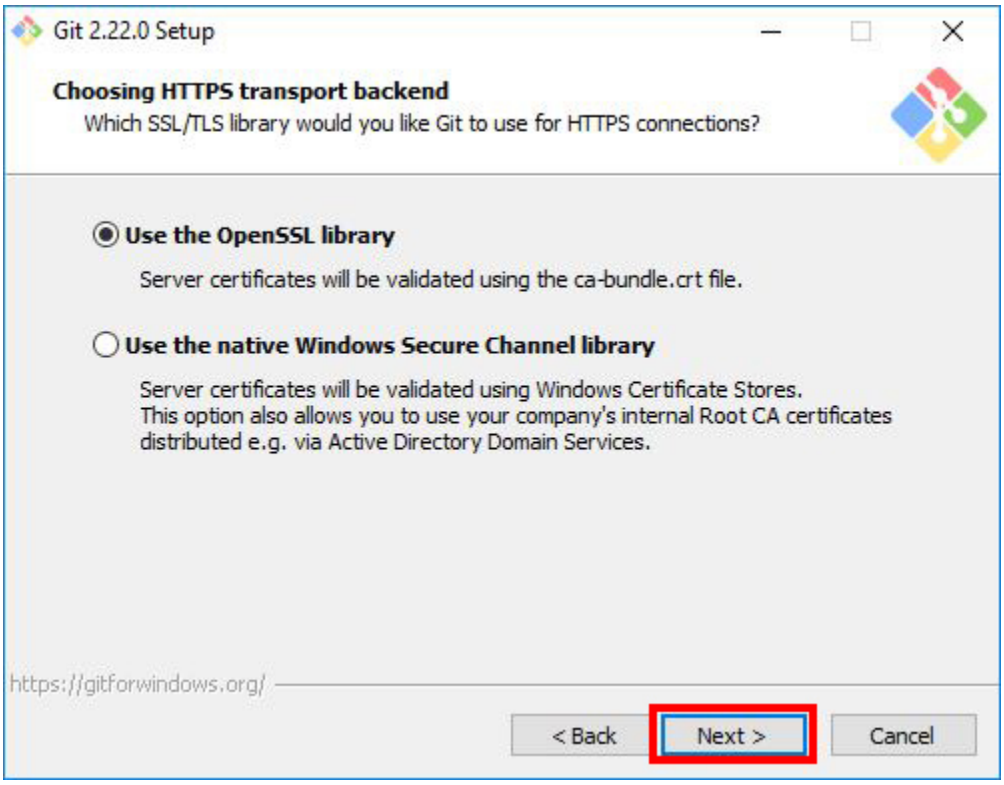


Step 5: Select the Start Menu folder and then choose the default editor used by Git.

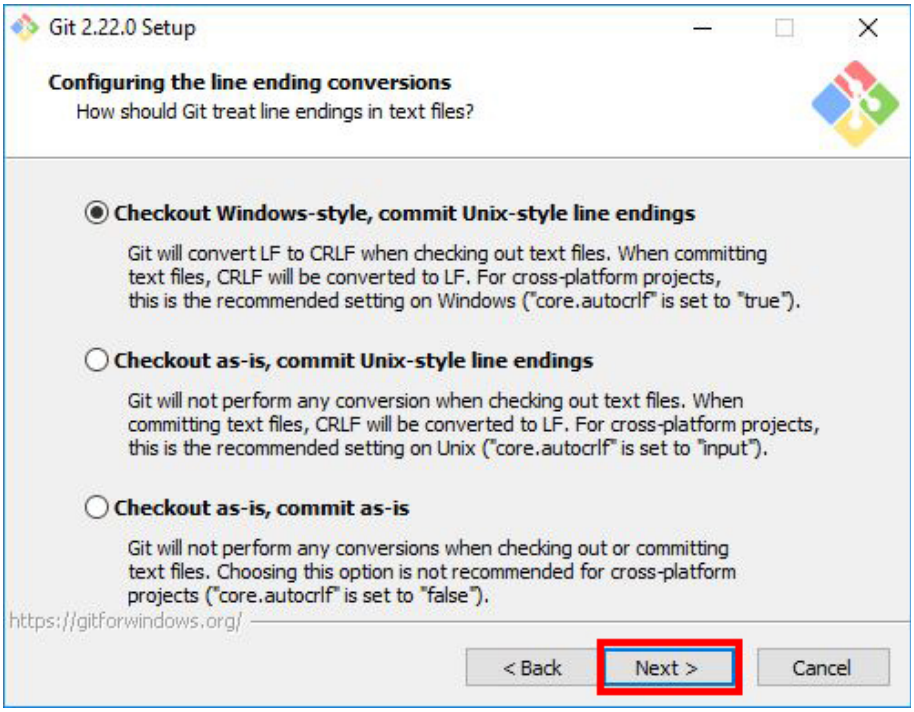


Step 6: Select “Git from the command line and also from 3rd-party software” option to run Git commands on Windows Command Prompt or Powershell.

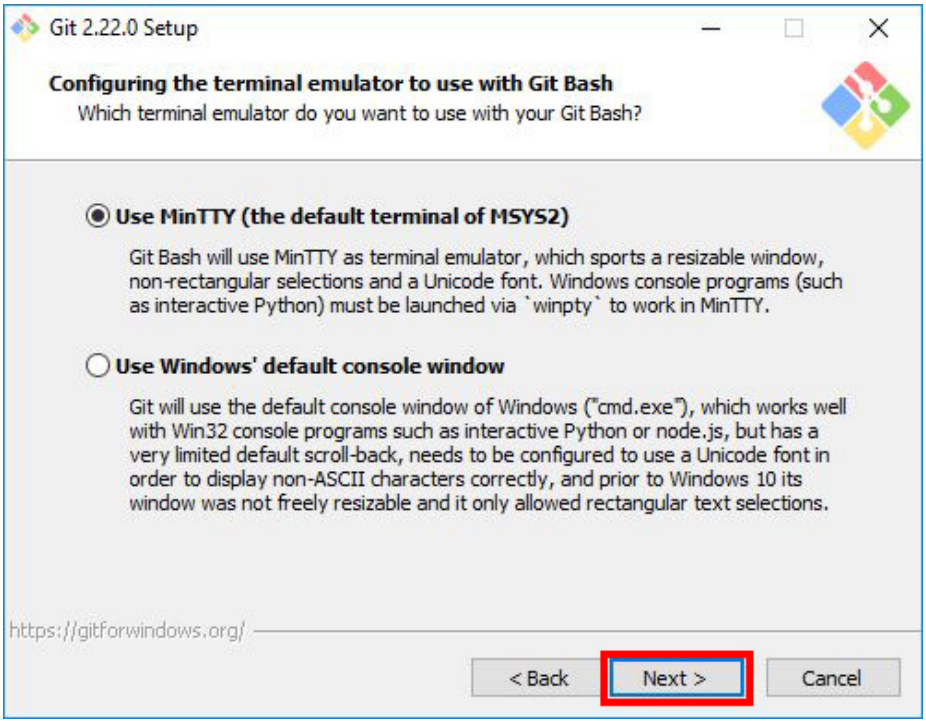
Step 7: Now select the “Use the OpenSSL Library” option.



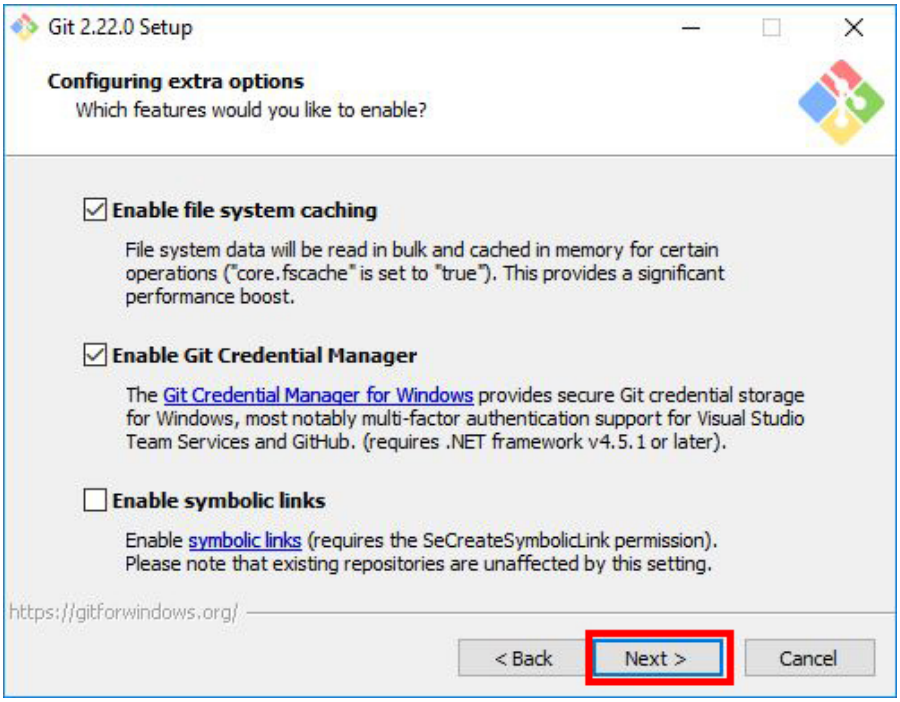
Step 8: Now select the **recommended version for your OS** to configure the Line Ending Conversions.



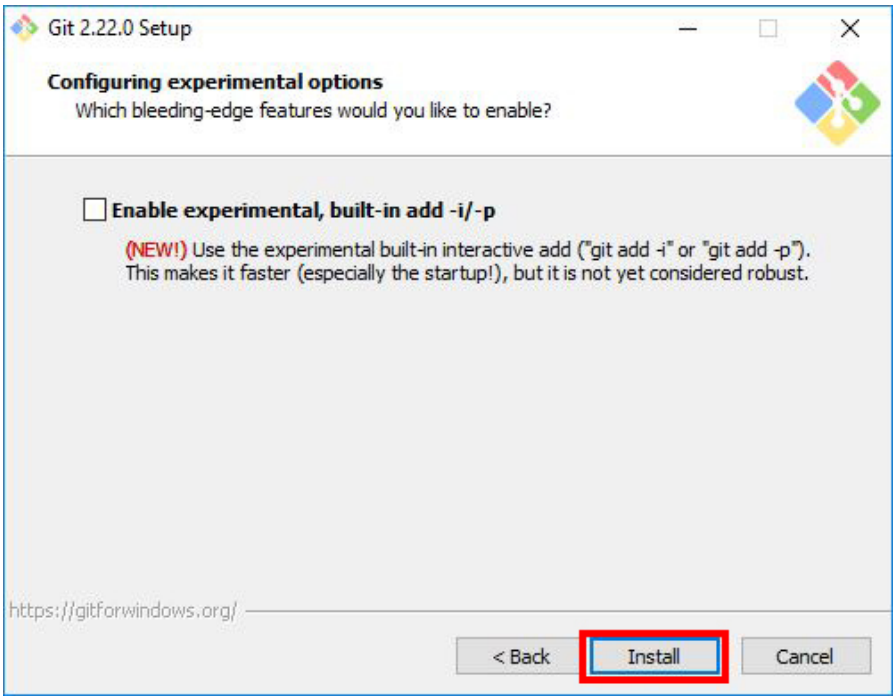
Step 9: Configure the Terminal Emulator to proceed with Git Bash.



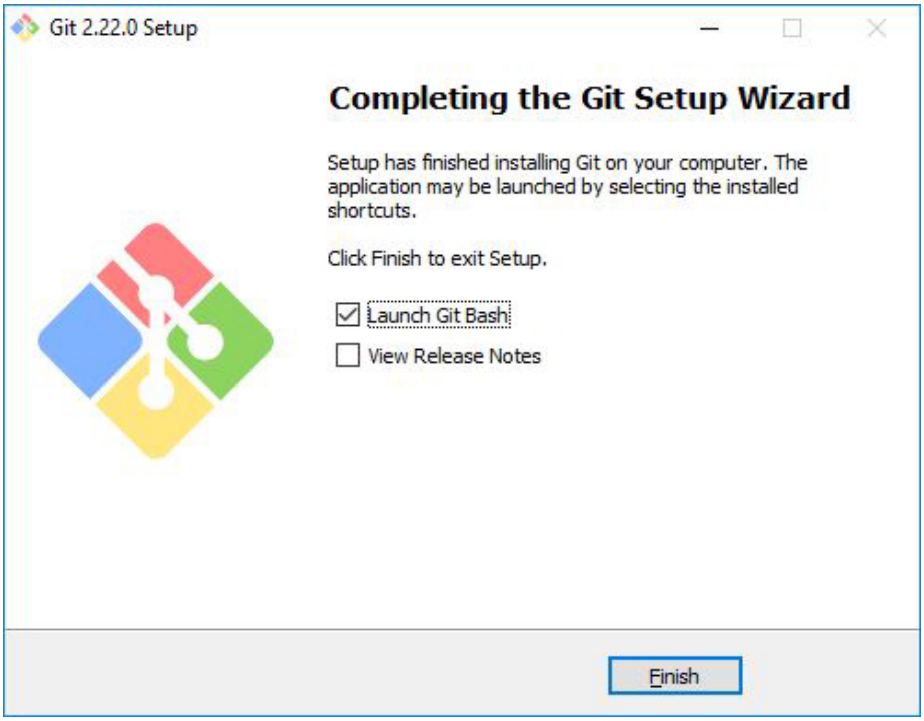
Step 10: Select the extra options that you want to go with (default options are enough)



Step 11: Enable experimental options if you want. This allows you to try out newer features that are still in development. Click on install to start installation.



Step 12: Installation may take a few minutes. Once it is complete, select launch Git Bash and click Finish to check if it is successfully installed.

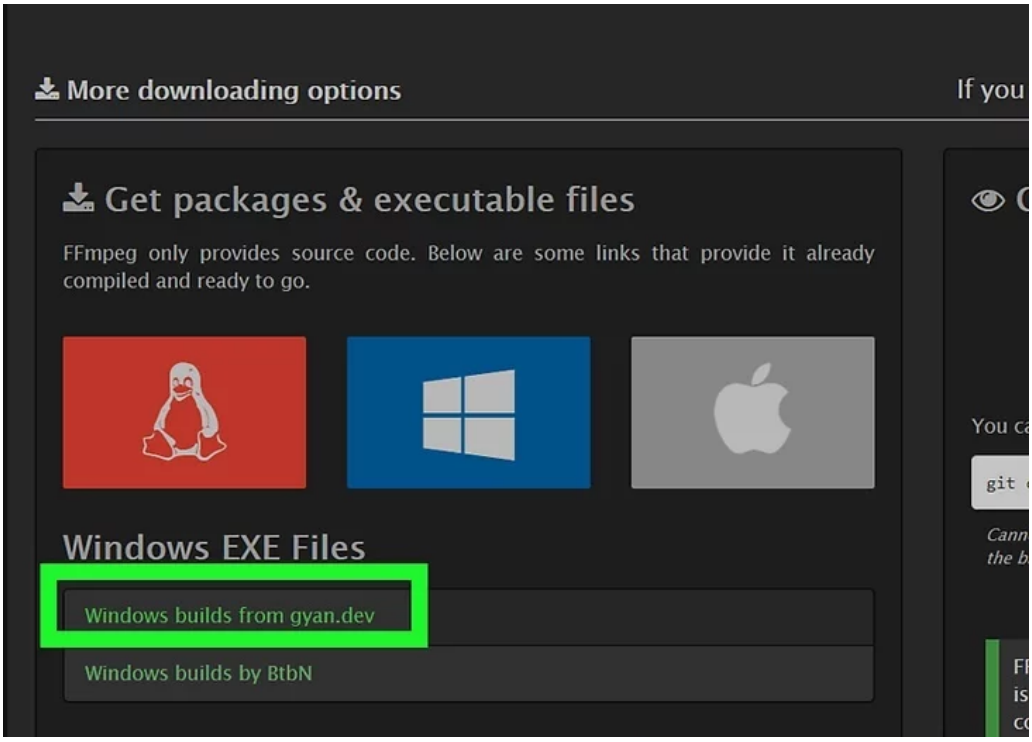


2.2 How to install FFmpeg?

FFmpeg is an open source media tool that allows you to perform various operations on an audio or video file using the command line.

Step 1: Visit the official website → https://ffmpeg.org/download.html and select your version of operating system. In this demo, we are using Windows.

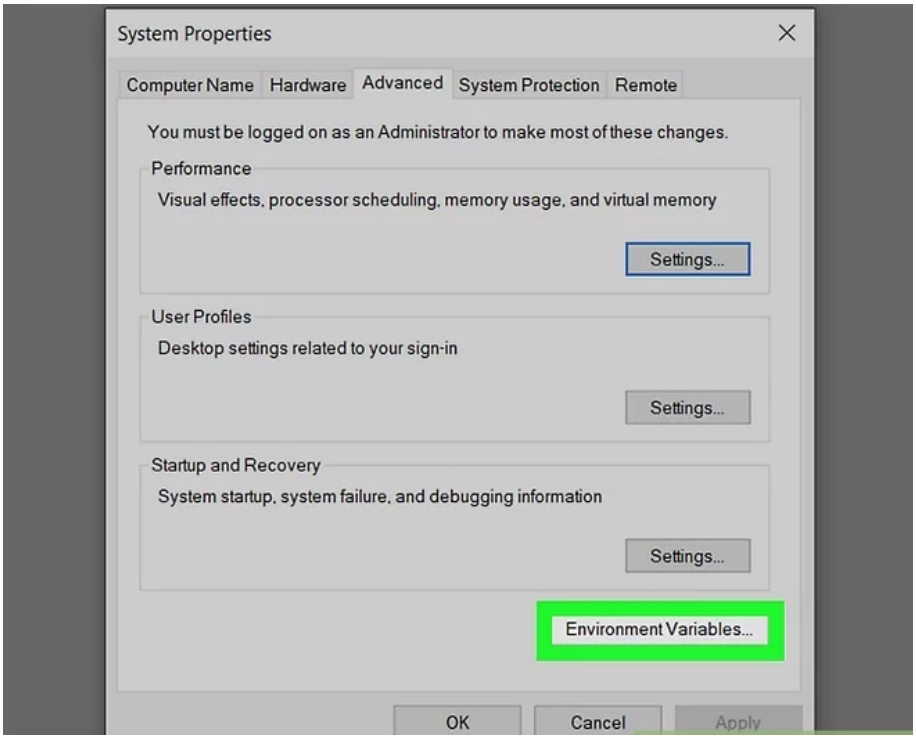
Step 2: Click Windows builds from gyan.dev. This takes you to a page that contains FFmpeg builds specifically for Windows that contains all of the hardware libraries you could possibly need.



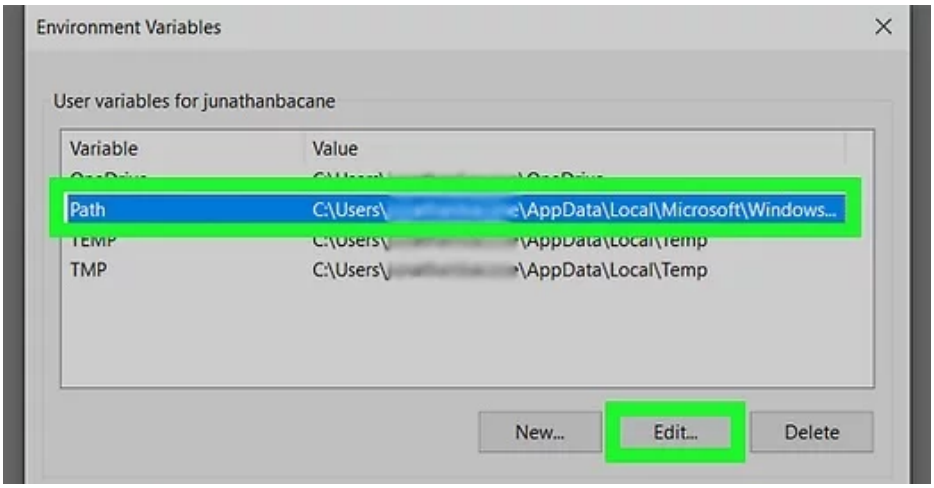
Step 3: Scroll down to the "git" section. It's about halfway down the page between the set of green boxes and the "release" sections. Now start the download by clicking the link shown in the figure below.

Step 4: Now extract the downloaded file and rename the folder to “FFmpeg”. Then move the folder into the Windows folder or Local Disc C.

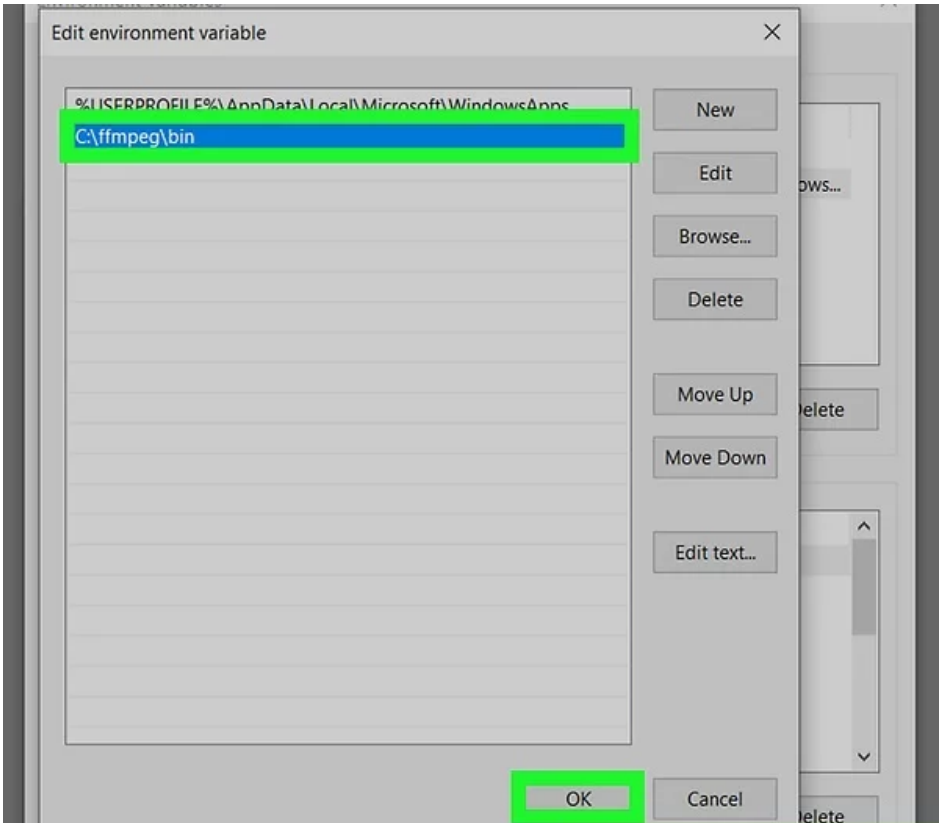
Step 6: Now open the “System Environment Variables Control Panel” and click on “Environment Variables”.



Step 7: Select the Path variable under "User variables for (your name)" and click Edit. A list of paths will appear.



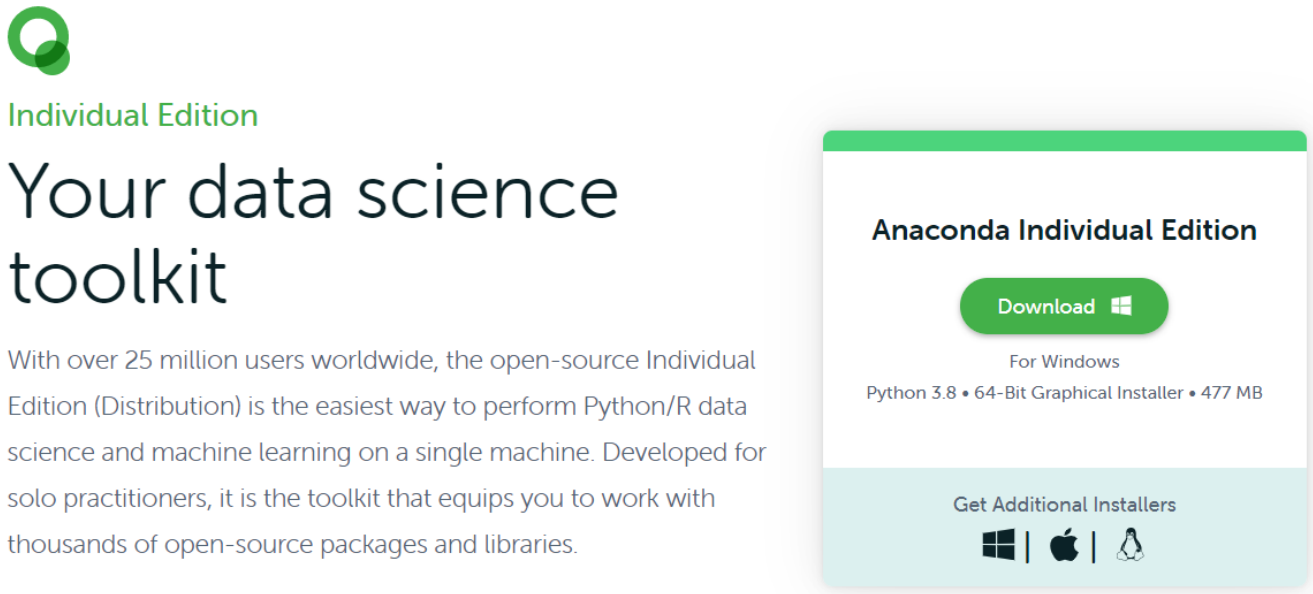
Step 8: Now add the FFmpeg binary directory to the path by clicking the “New” button. Type C:\ffmpeg\bin and click Ok to save your changes.



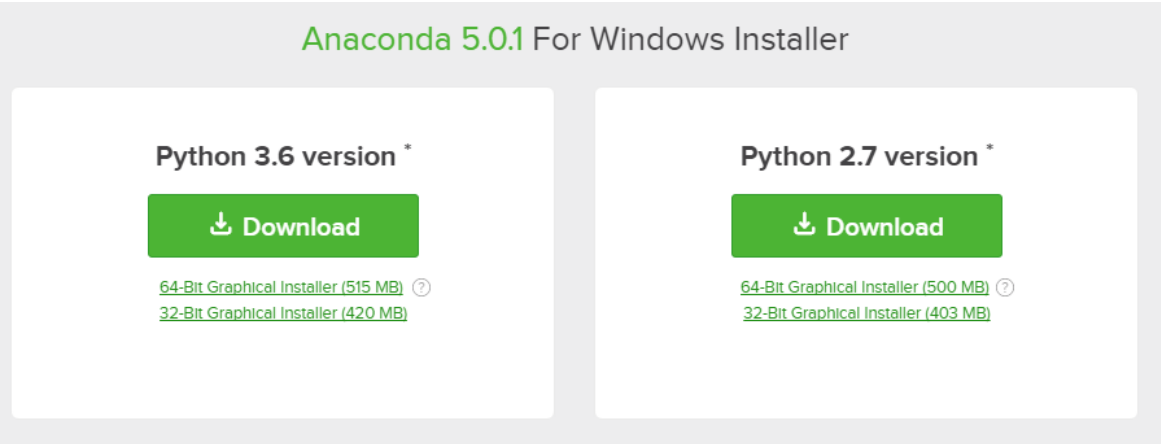
2.3 How to install Anaconda

Anaconda is an open source tool for developers. It provides various Integrated Development Environments (IDEs) like Jupyter Notebook, Spyder, etc. with more than 600 modules or packages or libraries to execute complex python programs.

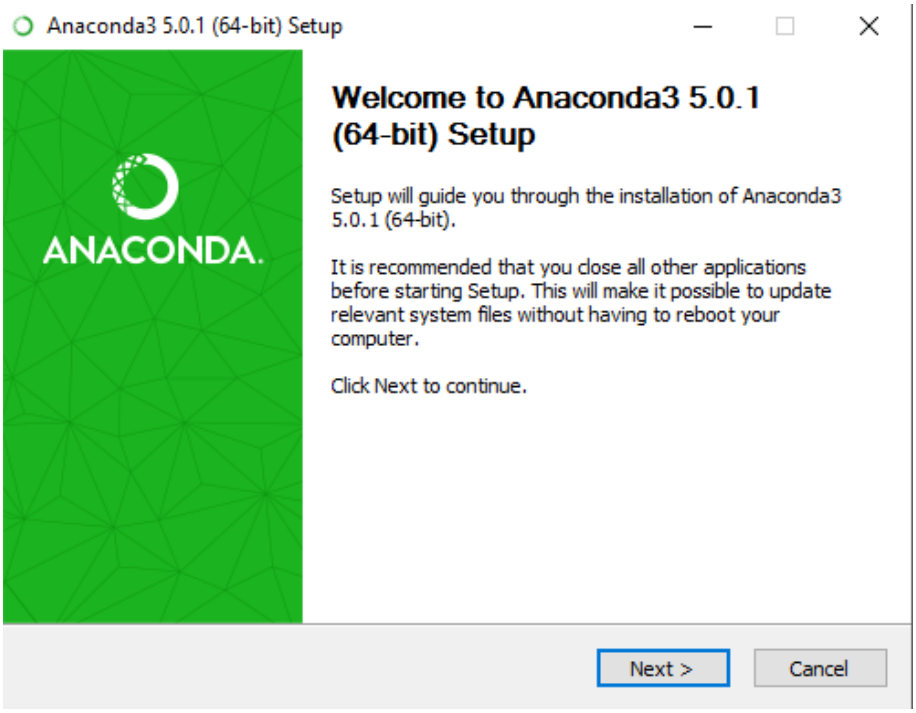
Step 1: Visit the official website → https://www.anaconda.com/download/ and select the operating system to go to the download page.



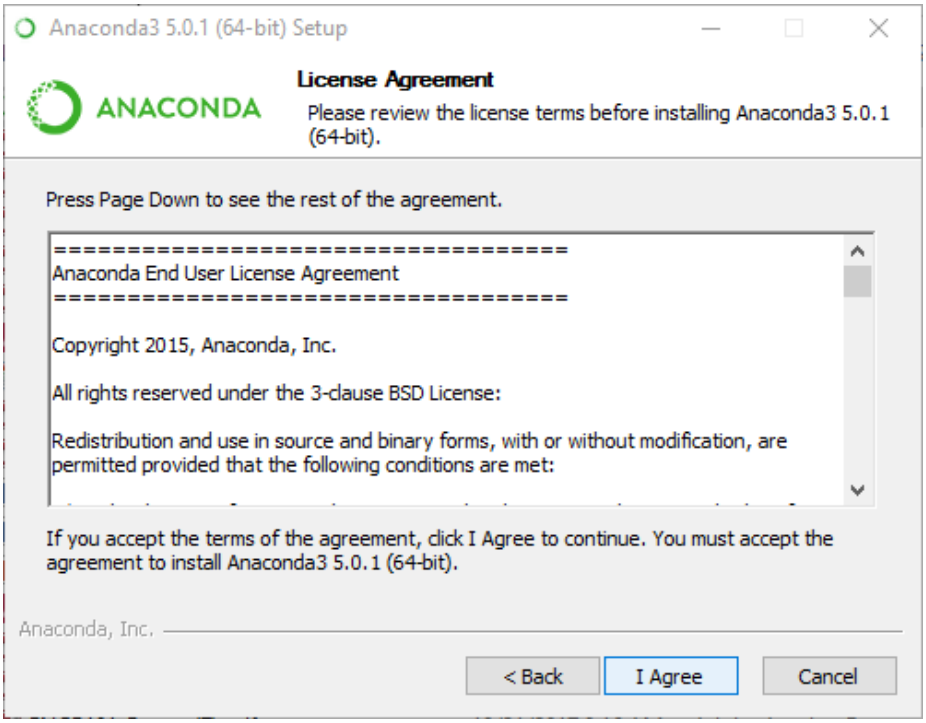
Step 2: Download the installer for the latest Python version.



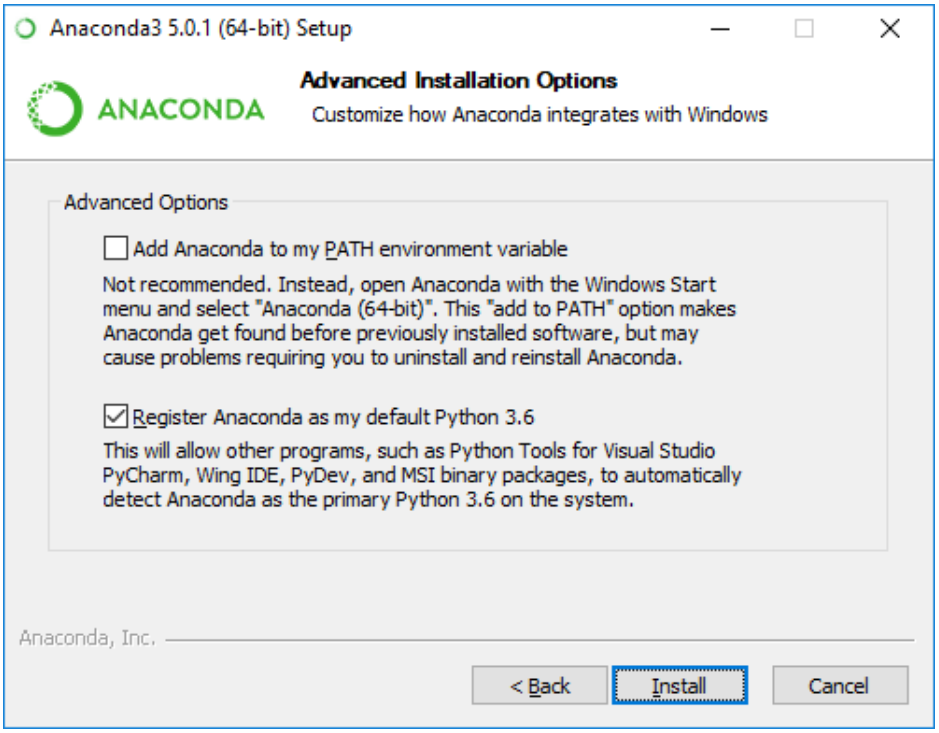
Step 3: Now open the installer and click on “Next”.



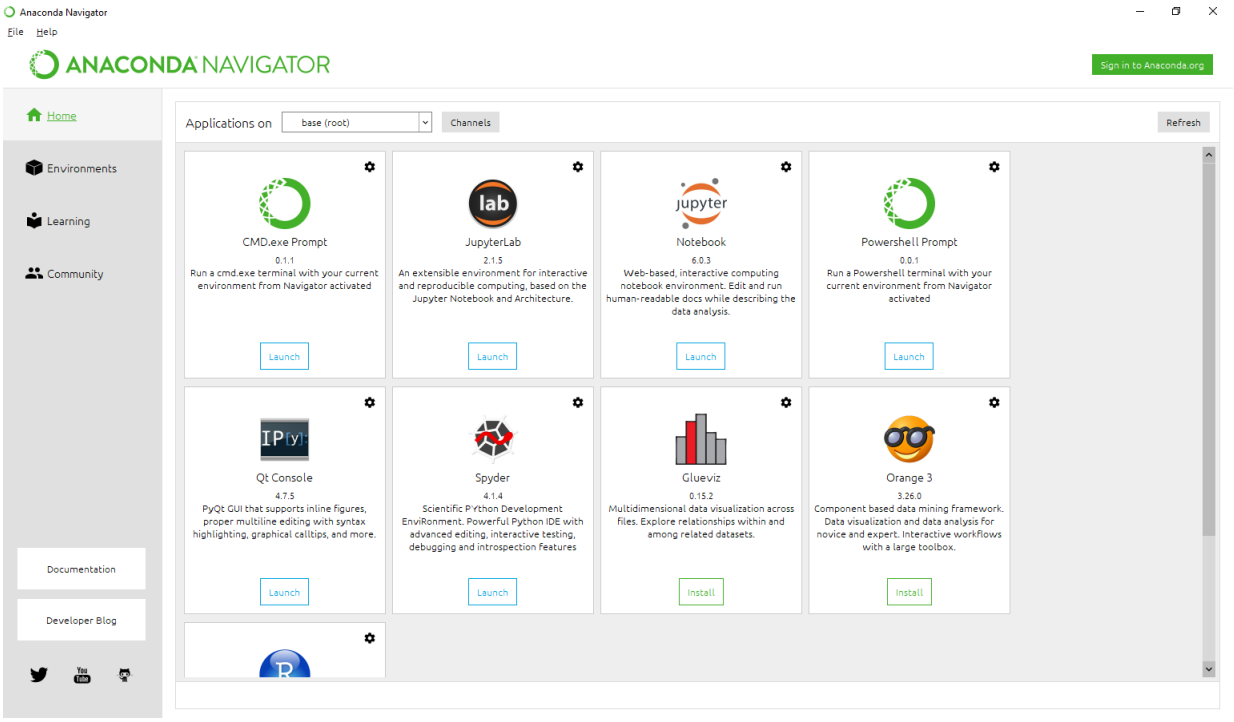
Step 4: Accept the license agreement and proceed.



Step 5: At the Advanced Installation Options screen, it is recommended that you **do not check** "Add Anaconda to my PATH environment variable".



Step 6: Now open Anaconda in the Start menu. We have used Jupyter Notebook to execute our program.

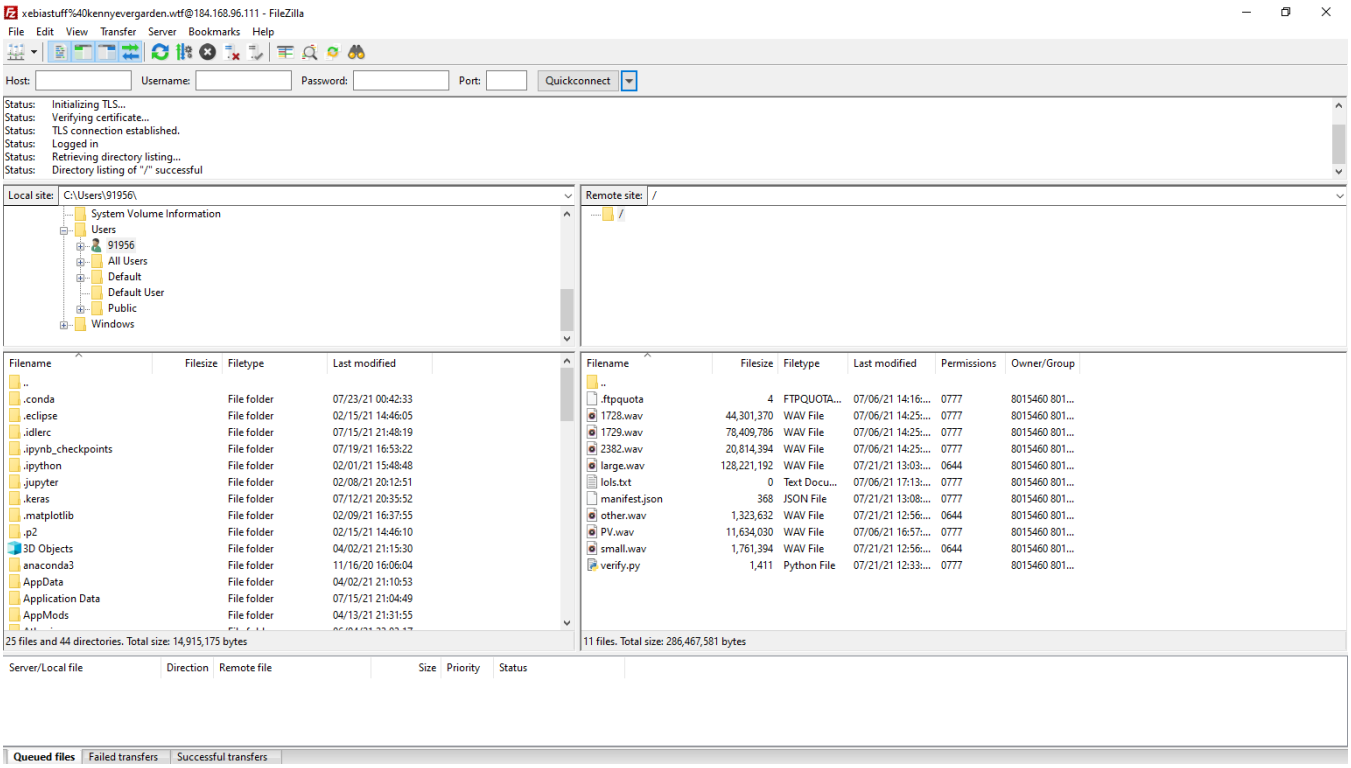


3. Database Management

This section guides you through all the necessary operations that can be carried out while handling the database.

3.1 Data Storage

The system allows you to manage the database on two possible options. You can either store the audio files on a web hosted server or on the hard drive of a local system or PC. This document will show you how you can use the data from different sources using a couple of lines of code.
If you have stored the database on a server then you can simple access the files from the server CMS or any FTP client like the demonstrated the example of Filezilla:

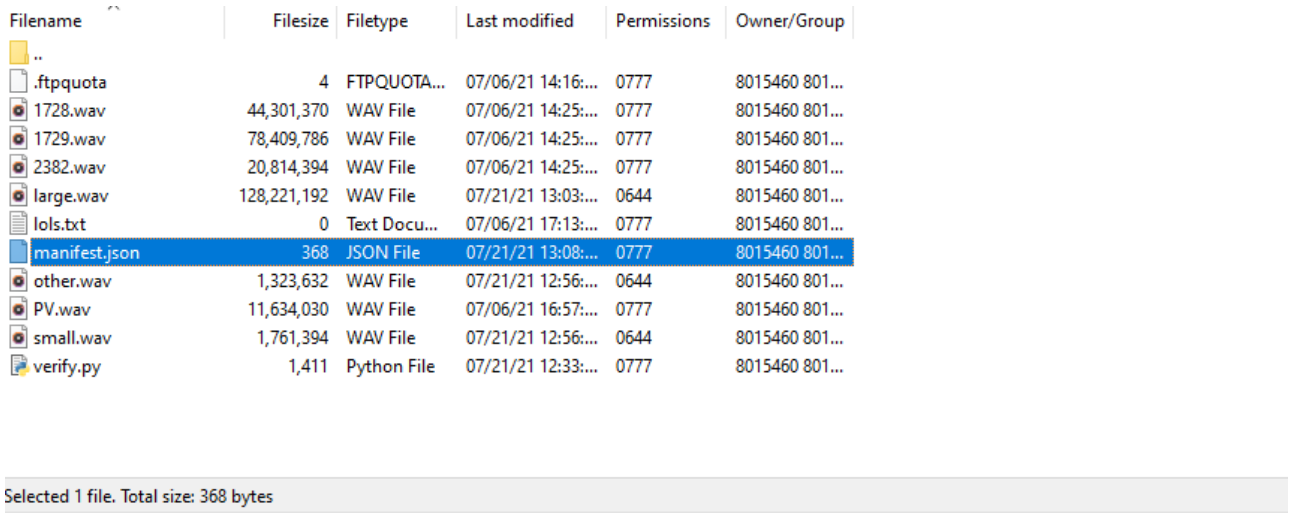


This is where all the data can be stored and managed.

3.2 Manifest Files

A Manifest File allows you to keep a track of all the audio files that you have fed to the server or saved in the local database in the past. It is the duty of the user to save each audio file's entry manually in the Manifest File.

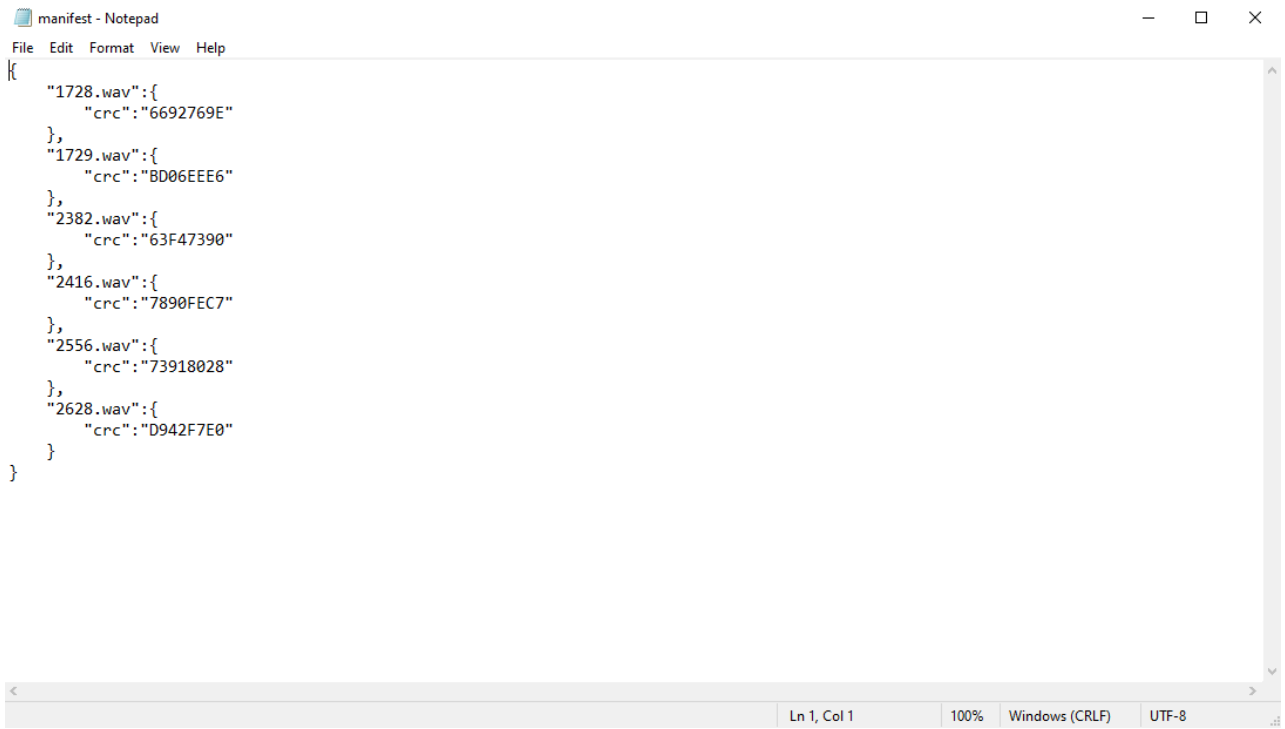
Manifest File on the FTP server:



Filename	Filesize	Filetype	Last modified	Permissions	Owner/Group
..					
.ftpquota	4	FTPQUOTA...	07/06/21 14:16:...	0777	8015460 801...
1728.wav	44,301,370	WAV File	07/06/21 14:25:...	0777	8015460 801...
1729.wav	78,409,786	WAV File	07/06/21 14:25:...	0777	8015460 801...
2382.wav	20,814,394	WAV File	07/06/21 14:25:...	0777	8015460 801...
large.wav	128,221,192	WAV File	07/21/21 13:03:...	0644	8015460 801...
lols.txt	0	Text Docu...	07/06/21 17:13:...	0777	8015460 801...
manifest.json	368	JSON File	07/21/21 13:08:...	0777	8015460 801...
other.wav	1,323,632	WAV File	07/21/21 12:56:...	0644	8015460 801...
PV.wav	11,634,030	WAV File	07/06/21 16:57:...	0777	8015460 801...
small.wav	1,761,394	WAV File	07/21/21 12:56:...	0644	8015460 801...
verify.py	1,411	Python File	07/21/21 12:33:...	0777	8015460 801...

Selected 1 file. Total size: 368 bytes

Manifest File:



Now, through this, even if you delete any files from your database, you'll still have a track of which files you deleted and what you currently have.

3.3 Basic Database Operations

If you have the database and the executable files on a local system then you can simply perform the manipulations. It is highly recommended to not disturb or make changes in any of the executable files unless absolutely necessary.
Once you have access to the server CMS, you can manage all the files from there as well. In case any changes in the executable files are required, you can simply download them and upload the recently updated file on the server.

4. Execution

4.1 Procedure

The execution of the entire system is carried out in four segments as mentioned below:
Step 1: Data Validation: This step deals with data analysis to check whether there is any discrepancy in the input audio files. These discrepancies may be of several types, like,

1. Corrupt file → This error shows up when a file is not completely downloaded or has any errors in the audio configurations.
2. Blank file → This means that the input audio is empty or there is no music in the file for a single second.
3. Wrong format → This is a type of exception when somebody feeds a file of a format other than *.wav* but saves it by *.wav*. For example if you pass a .txt file but save it as .wav, it will raise an error.

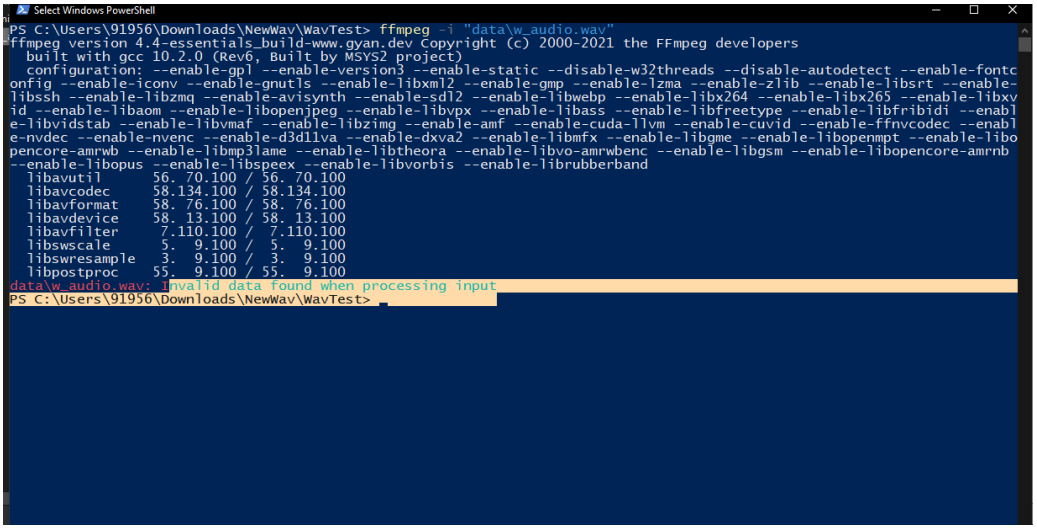
Step 2: Data Pre-processing: In this step, the audio files are checked for the threshold and then converted into Mel Spectrograms. These show the audio features in the form of a graph. Then they are converted into images so that they can be fed as input into the machine learning model to train it.

Step 3: Model Training: Now that you have the data ready to be fed to the model, you can begin with the training process. The model used in this product uses a bi-directional LSTM to train the images with Attention Mechanism architecture.

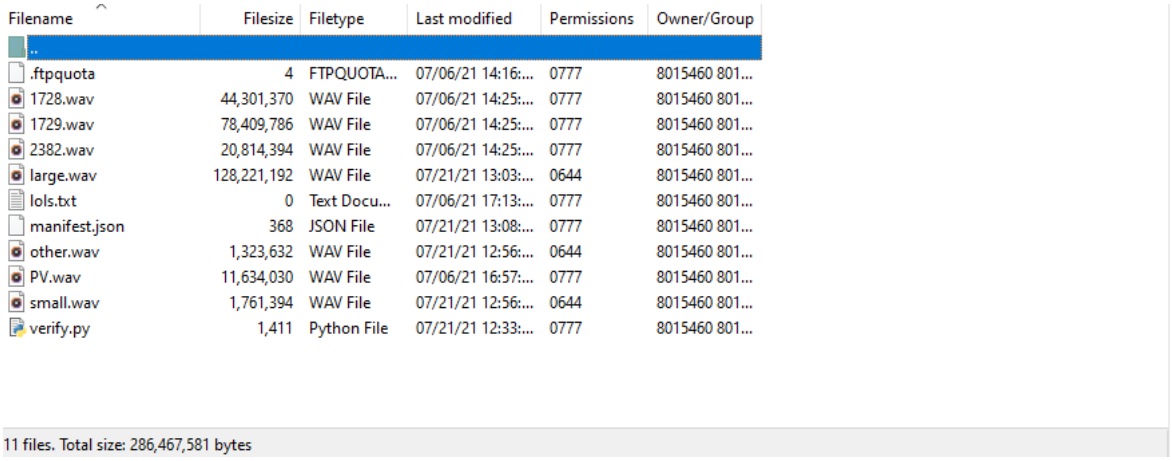
Step 4: Model Testing: You can check the efficiency of the model by passing sample inputs to check whether it predicts the correct instruments or not.

4.2 Data Validation

Once you have uploaded the files in the database, you can begin with checking for discrepancies in the files. FFmpeg command to analyse the input file:



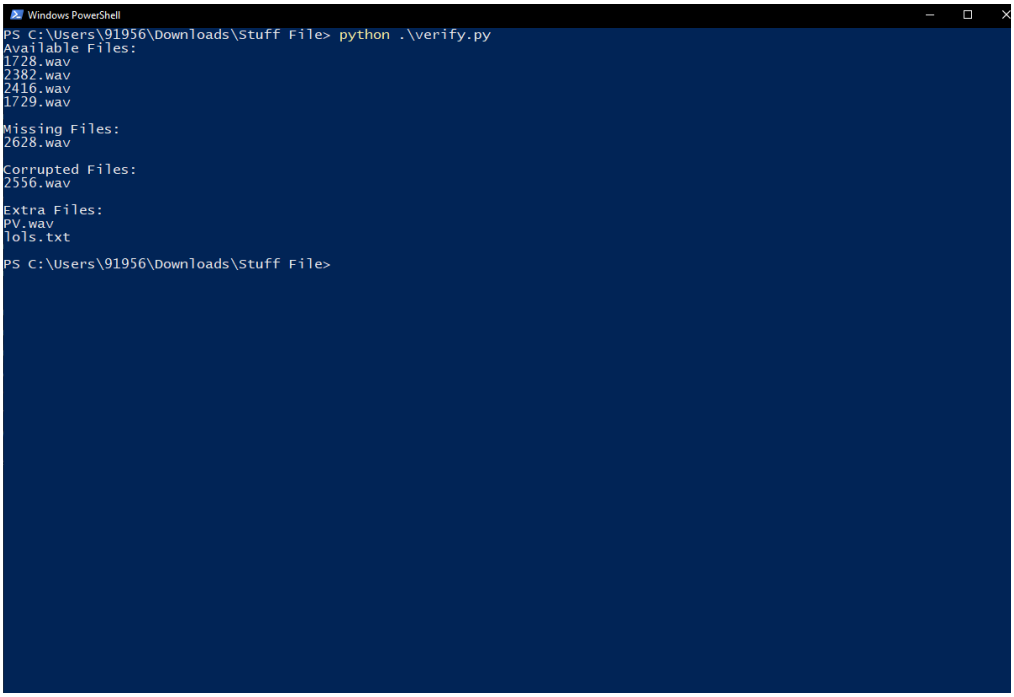
The *verify.py* checks for CRC of the files and matches it against the manifest file, allowing us to determine any missing or corrupted files.



Filename	Filesize	Filetype	Last modified	Permissions	Owner/Group
..					
.ftpquota	4	FTPQUOTA...	07/06/21 14:16:...	0777	8015460 801...
1728.wav	44,301,370	WAV File	07/06/21 14:25:...	0777	8015460 801...
1729.wav	78,409,786	WAV File	07/06/21 14:25:...	0777	8015460 801...
2382.wav	20,814,394	WAV File	07/06/21 14:25:...	0777	8015460 801...
large.wav	128,221,192	WAV File	07/21/21 13:03:...	0644	8015460 801...
lols.txt	0	Text Docu...	07/06/21 17:13:...	0777	8015460 801...
manifest.json	368	JSON File	07/21/21 13:08:...	0777	8015460 801...
other.wav	1,323,632	WAV File	07/21/21 12:56:...	0644	8015460 801...
PV.wav	11,634,030	WAV File	07/06/21 16:57:...	0777	8015460 801...
small.wav	1,761,394	WAV File	07/21/21 12:56:...	0644	8015460 801...
verify.py	1,411	Python File	07/21/21 12:33:...	0777	8015460 801...

11 files. Total size: 286,467,581 bytes

You can check for this by running the following command in the terminal:



This command reads input from the manifest file and checks which file is not there in the database but is registered in the manifest file.
All the files with any discrepancy, will be rejected and won't be sent to the pre-processing stage.

4.3 Data Pre-processing

You can convert the audio files into spectrograms by executing the below code segment in Jupyter Notebook:

```
#Convert audio to melspectrogram

def audio_to_melspectrogram(audio):

    spectrogram = librosa.feature.melspectrogram(audio,
                                                 sr=sr)

    return librosa.power_to_db(spectrogram).astype(np.float32)

def read_as_melspectrogram(path):

    mels = audio_to_melspectrogram(read_audio(path))
    return mels
```

Now you have to convert these spectrograms into images to be fed into the model for training.

```
def convert_wav_to_image(df, path):
    X = []
    for _, row in tqdm_notebook(df.iterrows()):
        x = read_as_melspectrogram('{}{}.wav'.format(path, str(row['audio']).split('.')[0]))
        X.append(x.transpose())

    return X

def normalize(img):
    eps = 0.001
    if np.std(img) != 0:
        img = (img - np.mean(img)) / np.std(img)
    else:
        img = (img - np.mean(img)) / eps
    return img

def normalize_dataset(X):
    normalized_dataset = []
    for img in X:
        normalized = normalize(img)
        normalized_dataset.append(normalized)
    return normalized_dataset
```

4.4 Model Training

This step allows you to train the model on the images that you converted in the pre-processing step.

```
# LSTM Model
input_shape = (431, 128)

model = Sequential()

model.add(Bidirectional(LSTM(64, return_sequences=True), input_shape=input_shape))
model.add(Attention(431))
model.add(Dropout(0.2))
model.add(Dense(16))
model.add(Dropout(0.2))
model.add(Dense(n_instruments, activation='softmax'))

model.compile(loss='categorical_crossentropy', optimizer="Adam", metrics=['acc'])
```

You can also check how accurately the data could be trained by the model by running the following segment:

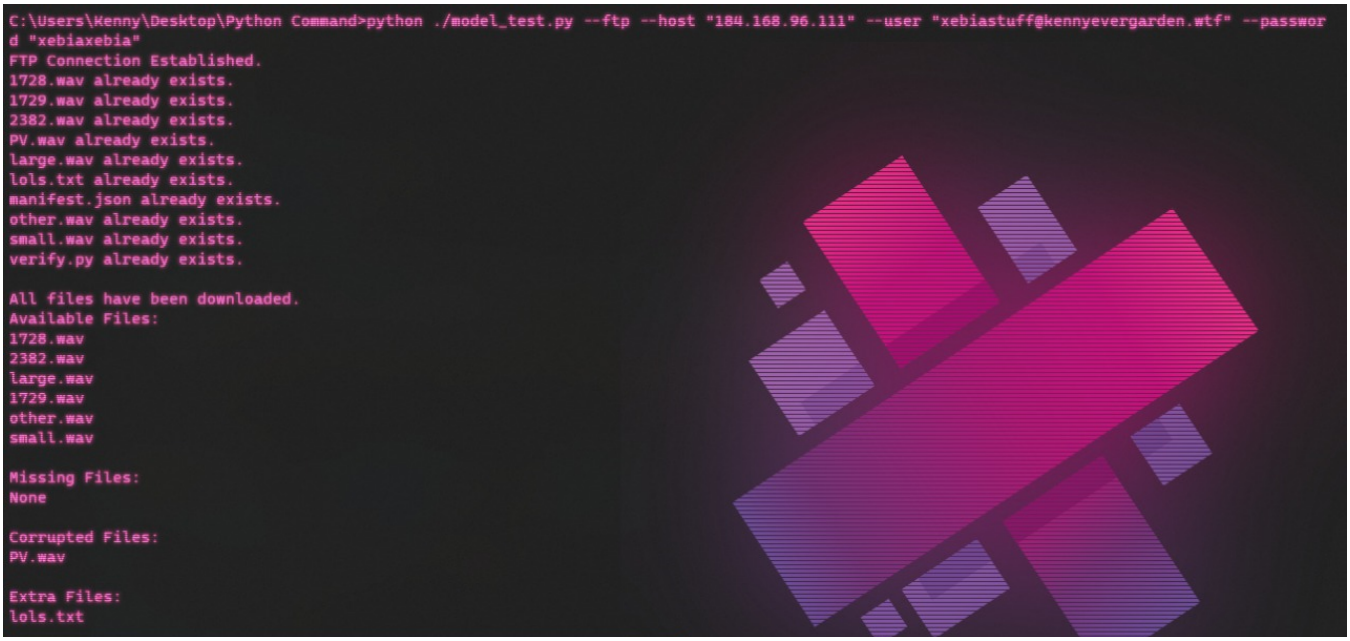
```
# Accuracy Visualization
epochs = range(250)

plt.figure(figsize=(15,5))
plt.plot(epochs, history.history['acc'])
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend(['Training Accuracy'])
plt.show()
```

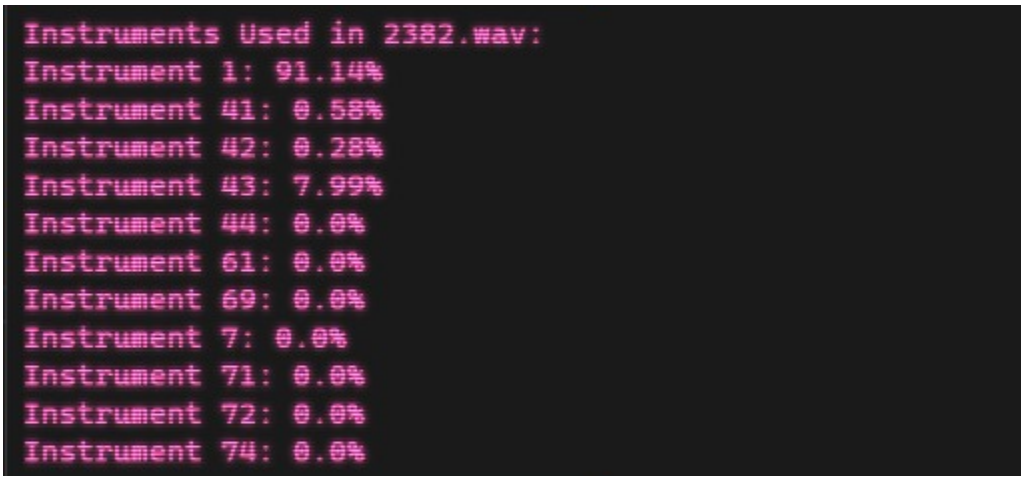
Note: The above code segments are just functions to implement the data on. You will know how to run the full code later in this document.

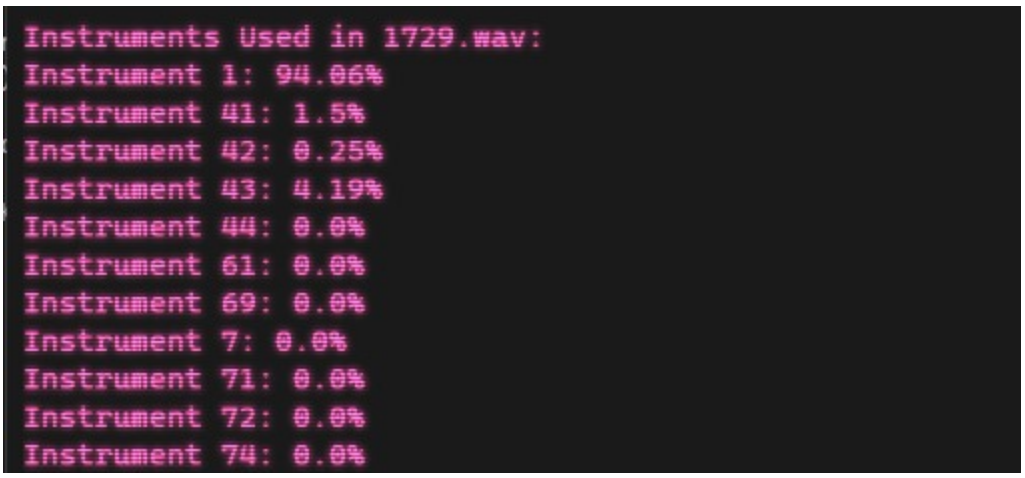
4.5 Model Testing

Now as instructed in Section 1.4, you can store data on both a local machine and a web hosted server.
Case 1: If the data is stored on an FTP server
Command to be executed in the command prompt to check the instruments used in an audio file:
python ./model_test.py --ftp --host “type your host ip address here” user “type your username here” --password “type your server password here”

The output is shown in the below snapshot:

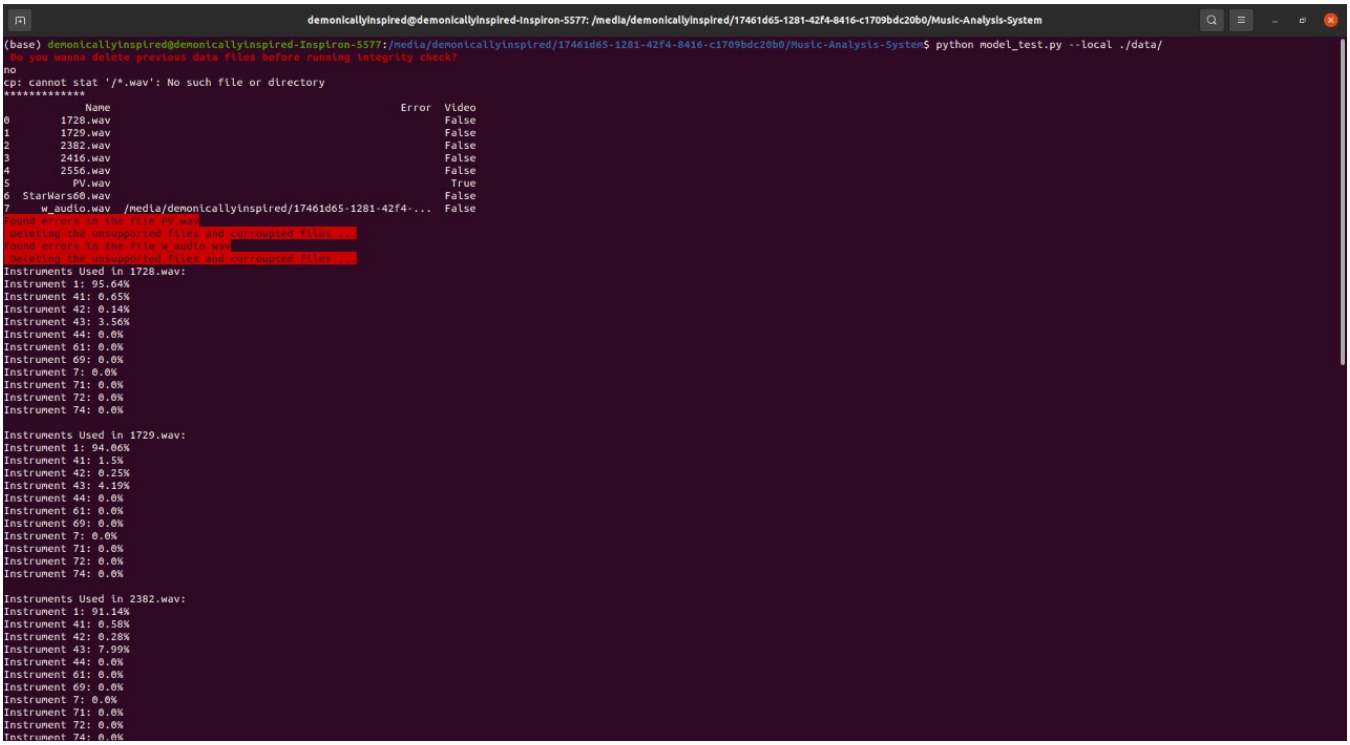




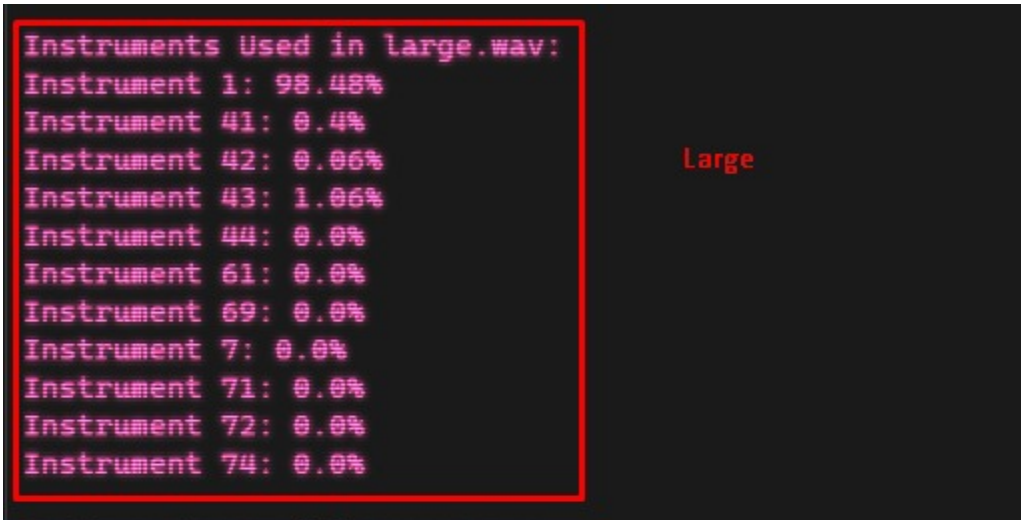


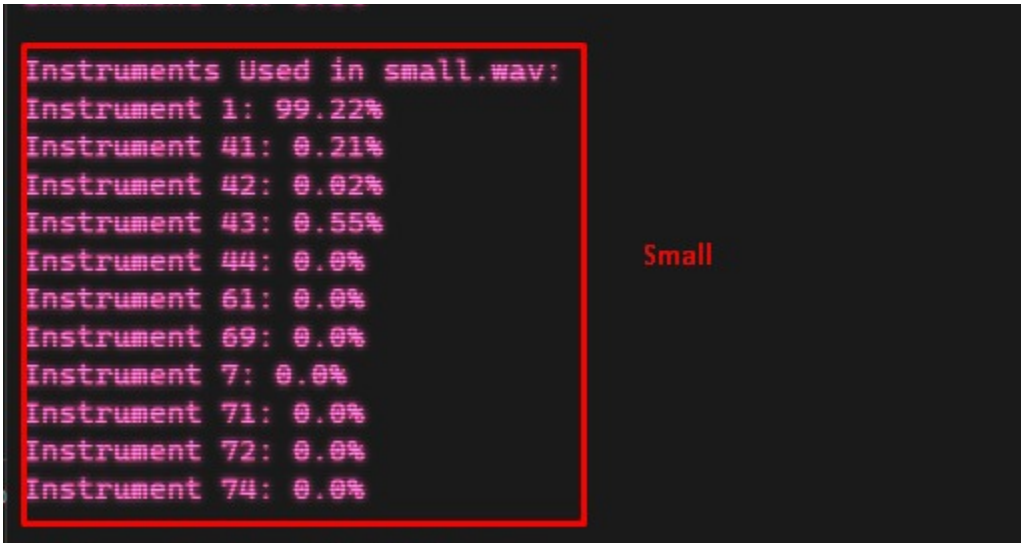
The usage of instruments in the audio sample by percentage is predicted.

Case 2: If the data is stored on a local machine
Command to be executed in the command prompt to check the instruments used in an audio file:
python model_test.py --local ./data/



The algorithm has also been tested on outliers, which include audios of unreasonably long or short duration, audios other than classical genre.
Refer to the following figures to check the outcomes:





4.6 How To Run The Code?

Step 1: Refer to the Step 6 of Section 2.4 to start the Jupyter Notebook and open the python file to run the code.

Step 2: Once you have opened the code, click on kernel in the menu bar at the top. Then run the whole code all at once and let it undisturbed until the execution is complete.