 MUSIFY- Music Composition using A.I.

## M\_ u\_si\_c Composition

Music Composition is a process of creating a new piece of music

Composition means “putting together”. Thus, music composition is something where music notes are put together in such a way that it gives pleasant sensation to our ears

Parameters such as pitch interval, notes, chords, tempo etc. are used for composing short piece of music

## A\_b\_ou\_t the Project

* The Project mainly focusses on music from **Piano** instrument
* Uses **Long Short Term Memory** (LSTM) , a type of Recurrent Neural Network (RNN)
* Platform: **Google Colab**
* Language: **Python 3.8**
* Libraries Used: **Tensorflow, Music21, Keras, NumPy, Sklearn, tqdm**
* Dataset: [Classical Music MIDI | Kaggle](https://www.kaggle.com/datasets/soumikrakshit/classical-music-midi)

## T\_e\_rm\_inologies

* **Note**: This is a sound produced by a single key
* **Chords**: The combination of 2 or more notes is called a chord
* **Octave**: The distance between two notes is stated as an octave in a piano

It is specifically the gap between the two notes that share the same letter name

* **Recurrent Neural Networks (RNN)**

A recurrent neural network is a class of artificial neural networks that make use of sequential information. They are called recurrent because they perform the same function for every single element of a sequence, with the result being dependent on previous computations

* **Long Short Term Memory (LSTM)**
  + LSTMs are a type of Recurrent Neural Network that can efficiently learn via gradient descent
  + Using a gating mechanism, they are able to recognize and encode long-term patterns
  + Useful to solve problems where the network has to remember information for a long period of time
  + Applications: Music and text generation etc.
  + Limitation: It requires lots of resources and time to get trained for real world applications

**L**\_**ib**\_**ra**\_**ries**

* **Music21**
  + Music21 is a Python toolkit used for computer-aided musicology
  + It allows us to teach the fundamentals of music theory, generate music examples and study music
  + The toolkit provides a simple interface to acquire the musical notation of MIDI files
  + Additionally, it allows us to create Note and Chord objects so that we can make our own MIDI files easily
* **Keras**
  + Keras is a high-level neural networks API that simplifies interactions with TensorFlow
  + It was developed with a focus on enabling fast experimentation
* **TensorFlow**
  + TensorFlow is a free and open-source software library for machine learning and artificial intelligence
  + It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks
* **NumPy**
  + NumPy is a Python library used for working with arrays
  + It also has functions for working in domain of linear algebra, Fourier transform, and matrices
* **tqdm**
  + *tqdm* is a library in *Python* which is used for creating Progress Meters or Progress Bars

## P\_r\_oje\_ct Structure

* **All Midi Files/** : This is the dataset folder containing various midi files of different composers
* **code.ipynb** : In this file, we will build, train and test our model
* **MOD/** : This directory contains optimizer, metrics, and weights of our trained model
* **AI\_composed\_music.mid**: This is a music file of predicted notes



**STEPS**

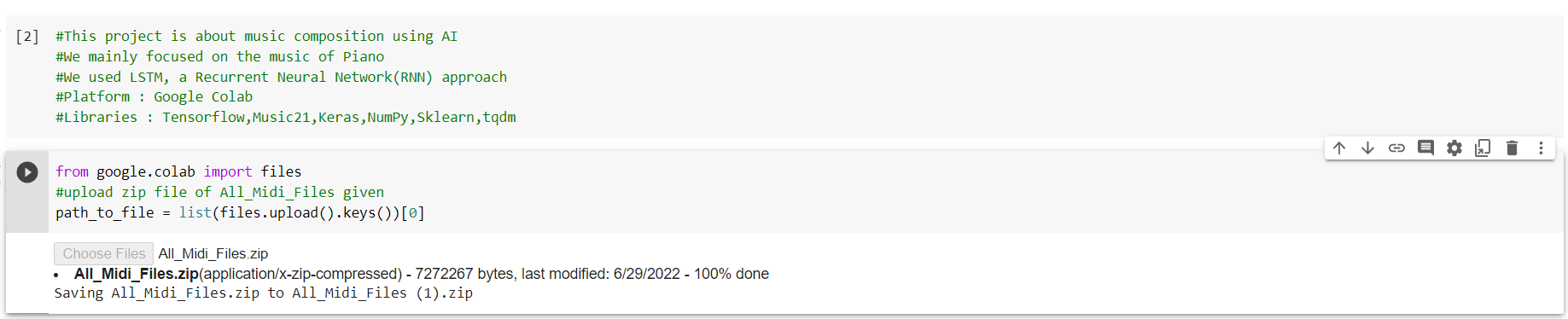
**(For developing Code from Scratch)**

**STEP 1**

**Choosing zip file f**

* To load midi files in the code, I need to load the data first to the google colab session
* I need to select the zip file of all music files

**CODE SNIPPET**

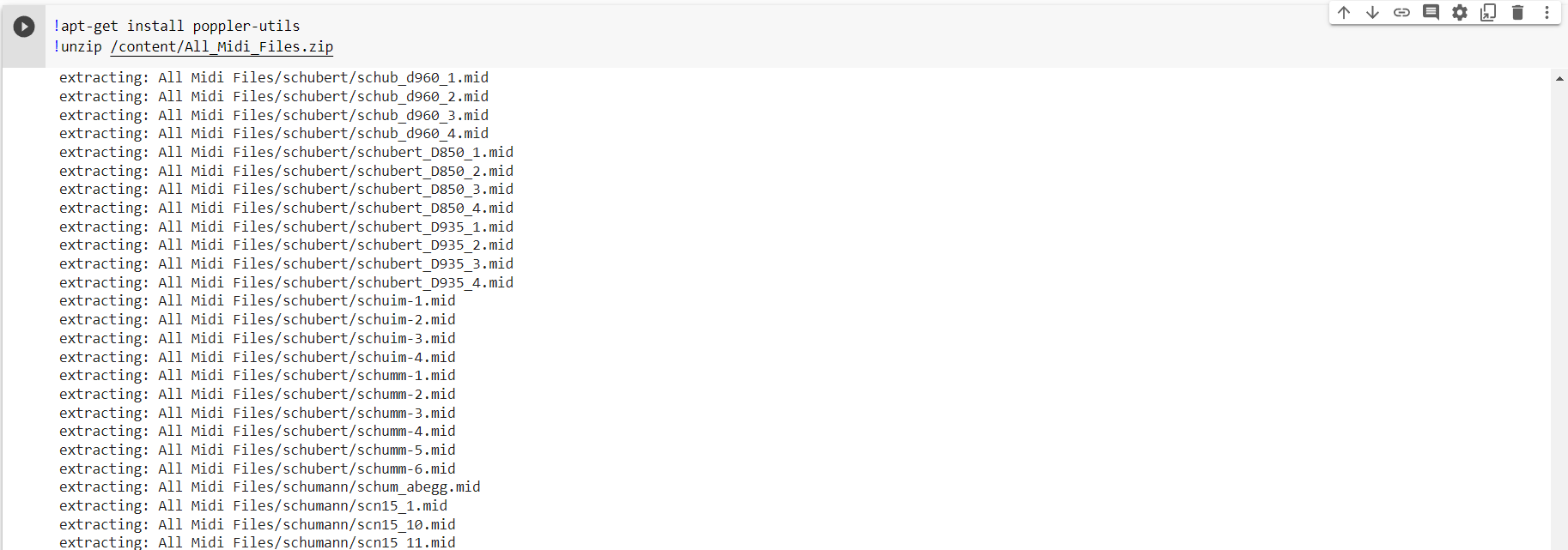


**STEP 2**

**Extracting zip file**

* After uploading the zip file, I need to extract the music files in the session itself.

**CODE SNIPPET**



**STEP 3**

**Import Libraries**

* + Import all the important Libraries

**CODE SNIPPET**



**STEP 4**

**Reading and Parsing the Midi File**

* + The midi file dataset has to be read using Music21 library
  + “**Haydn**” composed files has been used. (You can use more or less depending on your system)
  + For this project, the files that contain sequential streams of **Piano** data has only been worked on
  + All files are separated by their instruments and **Piano** is used only
  + Piano stream from the midi file contains many data like **Keys**, **Time Signature**, **Chord**, **Note** etc.
  + We require only **Notes** and **Chords** to generate music
  + Lastly, the arrays of notes and chords has to be returned

### CODE SCODNIPPETCXO

### CODE CODE SNIPPETl

### 

**STEP 5**

**Exploring the Dataset**

* + This is done to check the number of **unique notes** and their **distribution**
  + **50** is used as a threshold frequency
  + Only those notes which have frequencies more than **50** have been considered
  + **Two** dictionaries are created where one will have **notes index as a key and notes as value** and other will be the reverse of the first i.e. **key will be notes and value will be its respective index**
  + These dictionaries will be used in the next steps

### CODE SNIPPET

### 

**STEP 4**

**Input and Output Sequence for model**

* + **Input** and **output** sequences for our model are created
  + A **timestep** of **50** has been used. So, if we traverse 50 notes of our input sequence then the **51st** note will be the output for that sequence

#### Example:

* + - While using ‘SOC stands for Seasons of Code’ sentence with a timestep of 2, we will have to

provide 2 words at every input to get the output

#### (x) (y)

SOC stands for

Stands for Seasons

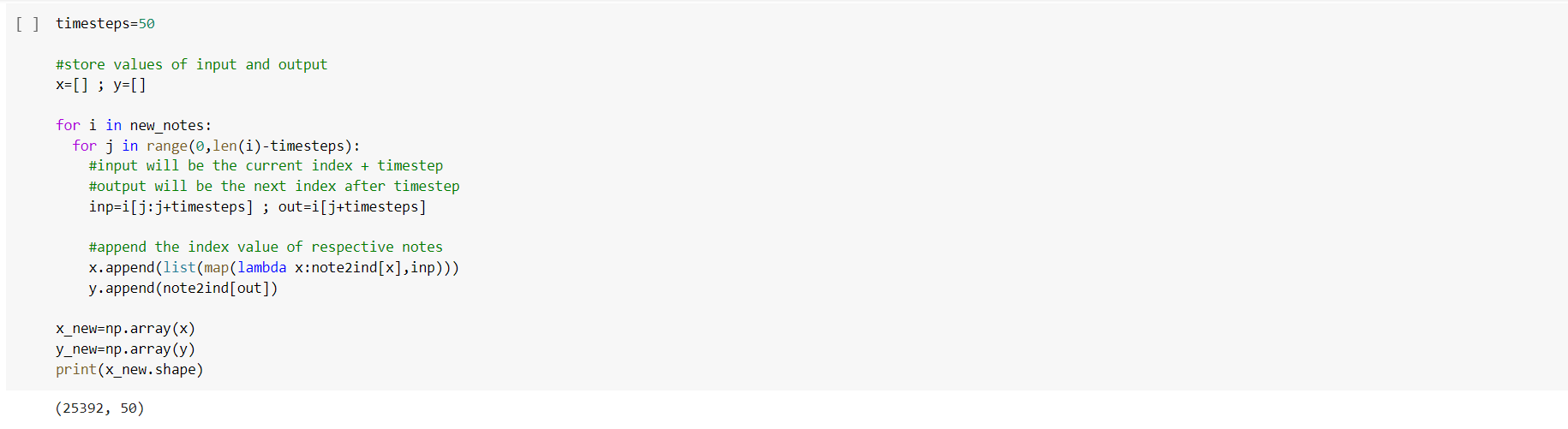
for Seasons of

Seasons of Code

* + As our model requires numeric data, all notes are converted to its respective index value using the

“**note2ind**” (note to index) dictionary which has been created earlier

**CODE SNIPPET**



* + Array for our model is re-shaped and the data is split into 80:20 ratio.

#### CODE SNIPPET

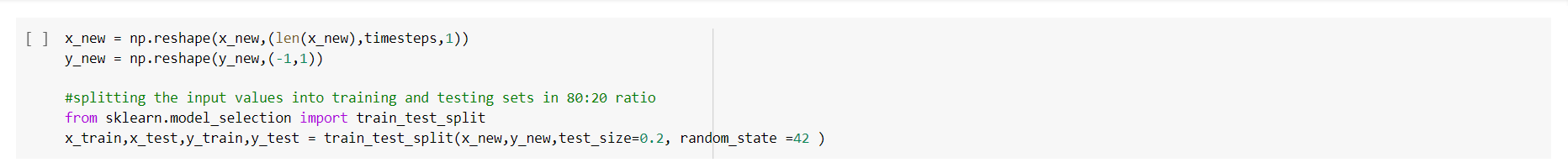
Training set

|  |  |  |
| --- | --- | --- |
|  |  | |
|  | |  |
| 80 % | | 20% |

Testing set

**STEP 5**

**Training and Testing sets**



**STEP 6**

**Building the Model**

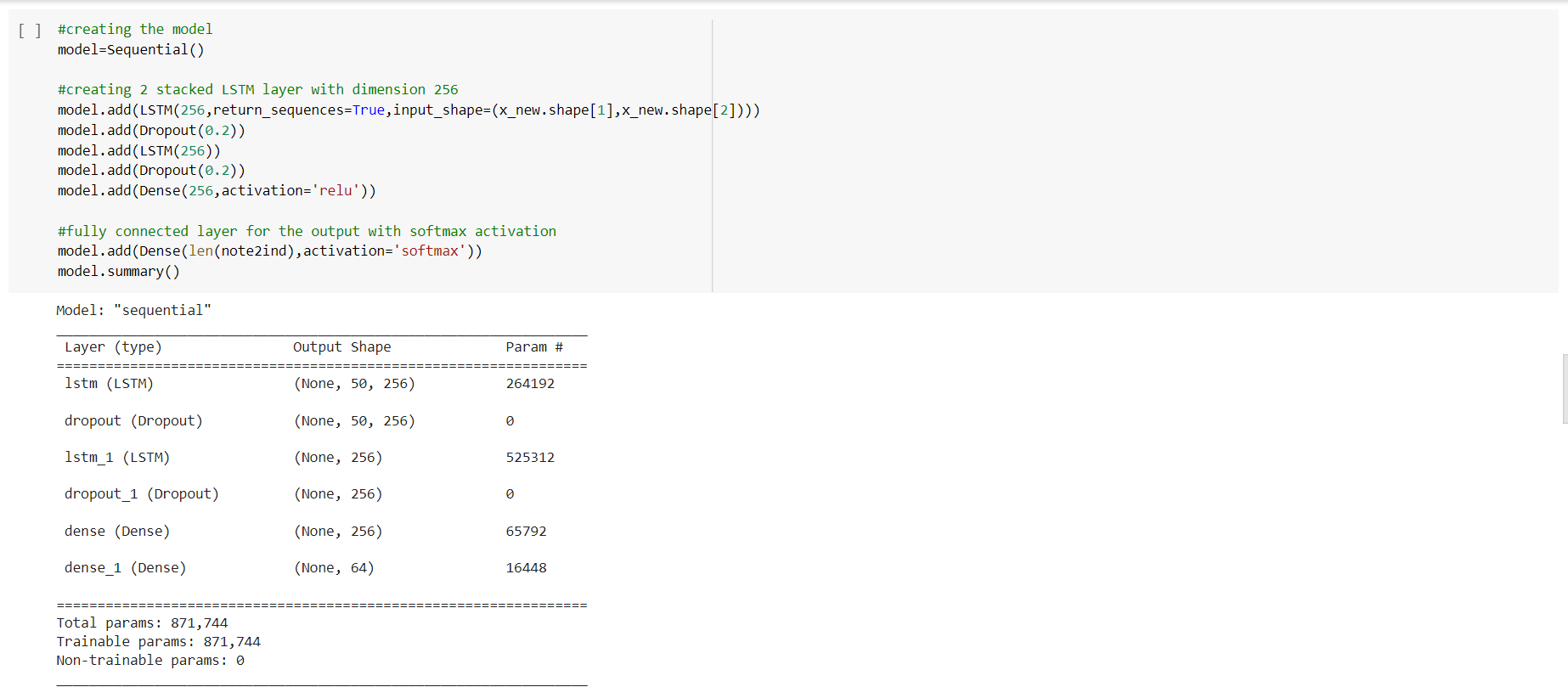
* + 2 stacked **LSTM** layers with a dropout rate of **0.2** are used
  + A fully connected **Dense** layer has been used for output
  + Output dimension of the Dense Layer is taken same as the length of our unique notes along with the

‘**softmax**’ activation function (Used for multi-class classification problems)

**Dropout** refers to ignoring units (i.e. neurons) during the training phase of certain set of neurons which is chosen at random. It basically prevents overfitting while training the model, while it does not affect the inference model.

**CODE SNIPPET**

**Building the Model**



**STEP 7**

**Training the Model**

* + After building the model, it is trained on the input and output data
  + For this, ‘**Adam**’ optimizer is used on **batch size** of **128** and for total **80 epochs**
  + After Training, model is **saved** for prediction

#### CODE SNIPPET



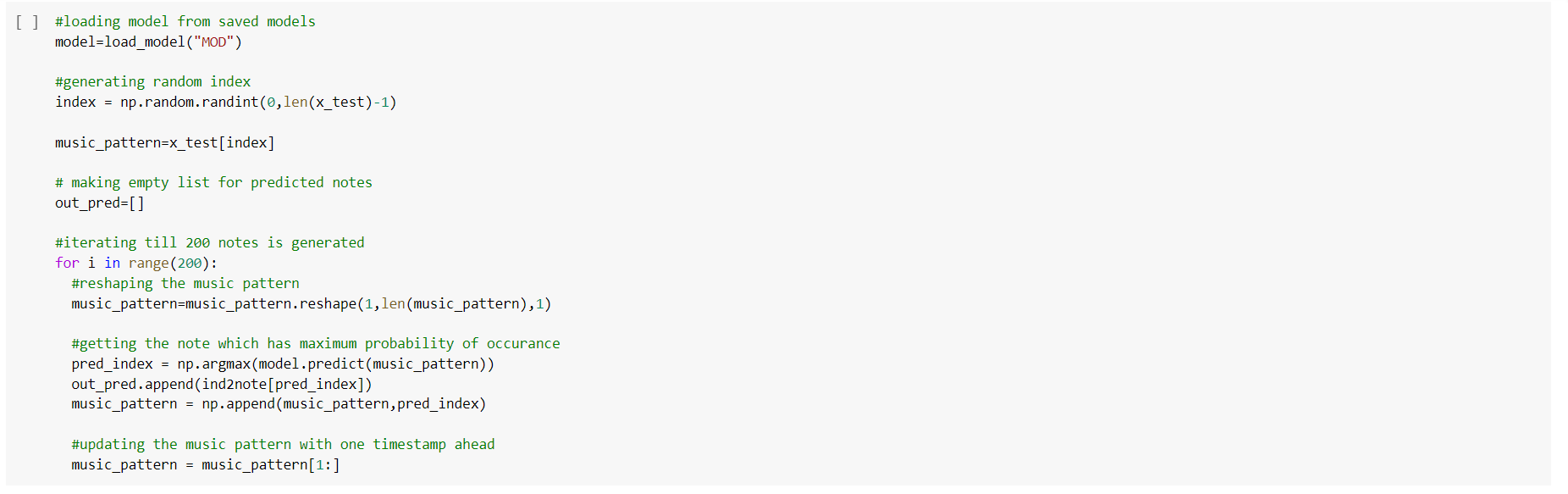
**STEP 8**

**Inference Phase**

* + Using the trained model, the notes will be predicted
  + A random integer(**index**) is generated for our testing input array which will be our testing input pattern
  + Our array is then re-shaped and the output is predicted
  + Using the ‘**np.argmax()**’ function, we get the data of the maximum probability value
  + This predicted index is converted to notes using ‘**ind2note**’(index to note) dictionary
  + Our next music pattern is one step ahead of the previous pattern
  + This process is repeated till we generate **200** notes
  + This parameter can be **changed** as per your requirements

**CODE SNIPPET**

**Inference Phase**



**STEP 9**

**Saving the File**

* + The predicted output notes are saved into a MIDI File

#### CODE SNIPPET

#### 

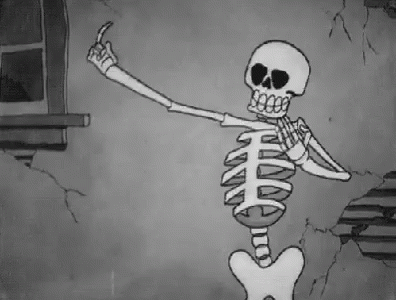
**OUTPUT AUDIO FILE LINK:**

[**https://drive.google.com/file/d/1A0se9zmoqUaVUjwIIiUAGMXGWdEm2uah/view?usp=sharing**](https://drive.google.com/file/d/1A0se9zmoqUaVUjwIIiUAGMXGWdEm2uah/view?usp=sharing)

**REFERENCE ARTICLE LINK:**

[**https://www.analyticsvidhya.com/blog/2021/12/step-by-step-guide-to-build-image-caption-generator-using-deep-learning/**](https://www.analyticsvidhya.com/blog/2021/12/step-by-step-guide-to-build-image-caption-generator-using-deep-learning/)

**THANK YOU**

****