

A Project Report  
on  
**Music Generation Using Machine Learning**

AI FOR DATA SCIENCE

By  
Jyothin Movva (2010030071)  
Satya Varsan KS (2010030151)  
Devaraj Acharya (2010030040)

under the supervision of  
**Dr. Arpita Gupta**

Assistant Professor



Department of Computer Science and Engineering

K L University Hyderabad,

Aziz Nagar, Moinabad Road, Hyderabad – 500075, Telangana, India.

April, 2022

## DECLARATION

PFSD project entitled “Human resource analysis using python” is a record of bonafide work of **Jyothin Movva (2010030071), KS Satyavarsan (2010030151), Devaraj Acharya (2010030040)** submitted in partial fulfillment for the award of B.Tech in the Department of Computer Science and Engineering to the K L University, Hyderabad. The results embodied in this report have not been copied from any other Departments/ University/ Institute.

Jyothin Movva (2010030071)

KS Satyavarsan (2010030151)

Devaraj Acharya (2010030040)

## **CERTIFICATE**

The Project Report entitled “Music Generation Using Machine Learning” is a record of bonafide work of Jyothin Movva(2010030071), Satya Varsan KS(2010030151), Devaraj Acharya (2010030040), submitted in partial fulfillment for the award of B.Tech in the Department of Computer Science and Engineering to the K L University, Hyderabad. The results embodied in this report have not been copied from any other Departments/University/Institute.

**Signature of the Supervisor**

Dr Arpita Gupta

**Signature of the HOD**

**Signature of the External Examiner**

## ACKNOWLEDGEMENT

First and foremost, we thank the lord almighty for all his grace & mercy showered upon us, for completing this project successfully.

We take a grateful opportunity to thank our beloved Founder and Chairman who has given constant encouragement during our course and motivated us to do this project. We are grateful to our Principal **Dr. L. Koteswara Rao** who has been constantly bearing the torch for all the curricular activities undertaken by us.

We pay our grateful acknowledgment & sincere thanks to our Head of the Department **Dr. Chiranjeevi Manike** for her exemplary guidance, monitoring, and constant encouragement throughout the course of the project. We thank **Ms.P. Sree Lakshmi** of our department who has supported us throughout this project holding the position of supervisor.

We wholeheartedly thank all the teaching and non-teaching staff of our department without whom we won't have made this project a reality. We would like to extend our sincere thanks, especially to our parents, our family members, and our friends who have supported us to make this project a grand success.

## **ABSTRACT**

- Automatic Music Generation is a process of composing a short piece of music with minimum human intervention
- The project achieves the generation of new music by taking in a large amount of data from the selected datasets to train a model.
- The model is then used to generate the new midi music file.
- The project is hosted in Django web-frame and is very user friendly and simple.

## TABLE OF CONTENTS

<b>S.NO</b>	<b>Topics</b>	<b>Page. No</b>
1.	Introduction	1
2.	Literature Survey	2-3
3.	Hardware & Software requirements 3.1 Hardware requirements 3.2 software requirements	7
4.	Functional & Non-functional Requirements 4.1 functional requirement 4.2 nonfunctional requirements	8
5.	Implementation 5.1. Work Flow 5.2. main.py 5.3. code explanation	9 – 12
6.	Results Discussion	13– 14
7.	Conclusion and Future Work	15
8.	References	16

# 1. INTRODUCTION

The days of debating whether or not artificial intelligence (AI) will have an influence on the music industry are long gone. Artificial intelligence is already being employed in a variety of applications. Now it's time to think about how it will impact the way we create and listen music. AI automates services, identifies patterns and insights in massive data sets, and helps generate efficiency in the music business, just like it does in other sectors. Companies in the music industry must acknowledge and prepare for the impact of ai on their business; those that do not will be left behind.

Automatic Music Generation is a process of composing a short piece of music with minimum human intervention. The project is about the same topic. Music generation is very important and It can be used in many applications.

This is achieved by recombination of musical phrases extracted from existing music, either live or pre-recorded. Music Generation a viable tool that can and is being used by producers to help in the creative process.

Music generation is one of the interesting applications of machine learning. Music itself is sequential data, it can be modelled using a sequential machine learning model such as the recurrent neural network. This modelling can help in learning the music sequence and generating the sequence of music data. In this Project, we are going to show how we can use neural networks, specifically RNN for automatic music generation.

## 2. LITERATURE SURVEY

S.no	Authors	Title	Publishing year
1	Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, Yi-Hsuan Yang	MuseGAN: Multi-track Sequential Generative Adversarial Networks for Symbolic Music Generation and Accompaniment	2017
2	Sageev Oore Ian Simon Sander Dieleman Douglas Eck Karen Simonyan	This Time with Feeling: Learning Expressive Musical Performance	2018
3	Nabil Hewahi ,Salman AlSaigal & Sulaiman AlJanahi	Generation of music pieces using machine learning: long short-term memory neural networks approach	2019
4	Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, Ilya Sutskever	Jukebox: A Generative Model for Music	2020



5	Li-Chia Yang, Szu-Yu Chou, Yi-Hsuan Yang	MIDINET: A CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORK FOR SYMBOLIC-DOMAIN MUSIC GENERATION	2017
---	--	--	------

### Articles used for Literature survey:

- **MuseGAN: Multi-track Sequential Generative Adversarial Networks for Symbolic Music Generation and Accompaniment**

MuseGAN is a project on music generation. In a nutshell, we aim to generate polyphonic music of multiple tracks (instruments). The proposed models are able to generate music either from scratch, or by accompanying a track given a priori by the user.

#### Pros:

An easy logical solution to generate the music. Using the chords changes to train the model is an effective way to train the model to generate new music

#### Cons:

The dataset uses only midi files which can be less rich compared to music format such as .wav

- **This Time with Feeling: Learning Expressive Musical Performance**

Pros:

Uses more musical convention to train the model instead of chord changes.

Cons:

With a tempo of 120 bpm each beat lasts for 500ms which corresponds to the change of 25ms. This will eventually add up to be a greater change.

- **Generation of music pieces using machine learning: long short-term memory neural networks approach**

Pros:

Easy to implement as the tools used are easy to understand. Uses a neural network to generate the music from the given dataset

Cons:

Having many LSTM layers makes the learning progress slower and less accurate than having one or two layers.

- **Jukebox: A Generative Model for Music**

Pros:

Music quality is comparatively better than other models. Each of these models has 72 layers of factorized self-attention on a context of 8192 codes.

Cons:

Suffer from hierarchy collapse due to use of successive encoders coupled with autoregressive decoders

- **MIDINET: a convolutional generative adversarial network for symbolic-domain music generation**

Pros:

MidiNet performs comparably with MelodyRNN models in being realistic and pleasant to listen to, yet MidiNet's melodies are reported to be much more interesting.

Cons:

The dataset uses only midi files which can be less rich compared to music format such as .wav

Dataset	Characteristics	Technique
Midi melody files for theme song	60 midi files	LSTM
Midi piano files	92 midi files	LSTM
Lakh pianoroll dataset	174,154 multitrack piano rolls	GAN (Generative Adversarial Networks)
Lo-Fi Hip Hop MIDI's	93 midi files	CNN and GAN
Multi-modal MIREX Emotion Dataset	193 midi files	

### **3. HARDWARE AND SOFTWARE REQUIREMENTS**

#### **3.1 HARDWARE REQUIREMENTS**

- Processor –
  1. AMD Athlon 3000G and above
  2. Intel i3 4th gen and above
- Ram – 4Gs
- Storage - 240gs SSD / 512gb HHD

#### **3.2 SOFTWARE REQUIREMENTS**

- Software –
  1. windows 10
  2. visual studio code
  3. python
- Python –
  1. Keras
  2. Music21
  3. Pandas
  4. Pickle

## **4. FUNCTIONAL & NON-FUNCTIONAL REQUIREMENTS**

### **4.1 FUNCTIONAL REQUIREMENTS**

The required python packages :

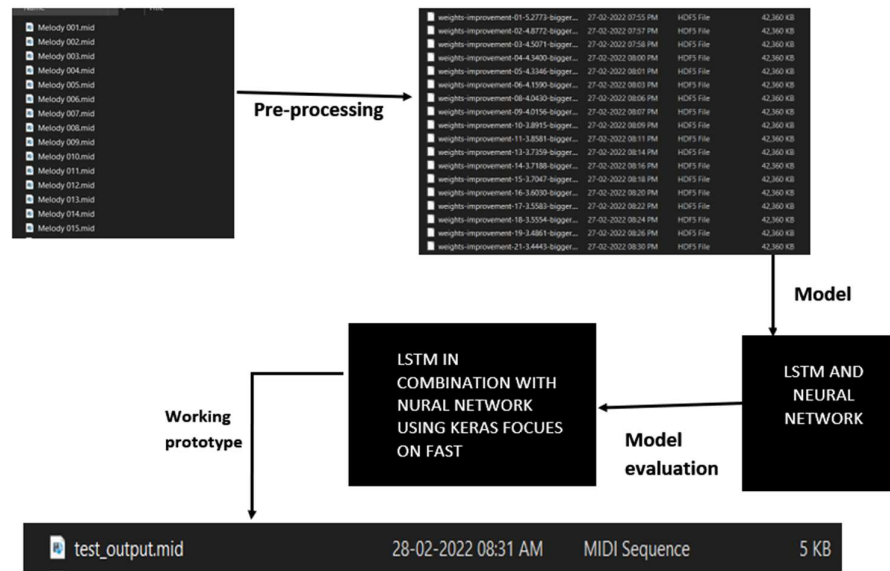
- Pandas – csv file handling
- Keras – NN and LSTM
- Music21 – midi file management
- Pickle – serialize and rdserializing

### **4.2 NON-FUNCTIONAL REQUIREMENTS**

- Requires the application to take input from a webpage
- Needs to generate a model from the input
- From the model the application needs to generate a new midi output file

## 5. IMPLEMENTATION

### 5.1 WORK FLOW



### 5.2 code

```
from django.shortcuts import redirect, render
from .models import Document
from .forms import DocumentForm
import pickle
import numpy
from music21 import instrument, note, stream, chord
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import LSTM
from keras.layers import BatchNormalization as BatchNorm
from keras.layers import Activation

def generate():
    with open('data/notes', 'rb') as filepath:

        notes = pickle.load(filepath)
```

```

pitchnames = sorted(set(item for item in notes))

n_vocab = len(set(notes))

network_input, normalized_input = prepare_sequences(notes, pitchnames, n_vocab)
model = create_network(normalized_input, n_vocab)
prediction_output = generate_notes(model, network_input, pitchnames, n_vocab)
create_midi(prediction_output)

def prepare_sequences(notes, pitchnames, n_vocab):
    note_to_int = dict((note, number) for number, note in enumerate(pitchnames))

    sequence_length = 100
    network_input = []
    output = []

    for i in range(0, len(notes) - sequence_length, 1):
        sequence_in = notes[i:i + sequence_length]
        sequence_out = notes[i + sequence_length]
        network_input.append([note_to_int[char] for char in sequence_in])
        output.append(note_to_int[sequence_out])

    n_patterns = len(network_input)

    normalized_input = numpy.reshape(network_input, (n_patterns, sequence_length, 1))

    normalized_input = normalized_input / float(n_vocab)

    return (network_input, normalized_input)

def create_network(network_input, n_vocab):
    """ create the structure of the neural network """
    model = Sequential()
    model.add(LSTM(
        512,
        input_shape=(network_input.shape[1], network_input.shape[2]),
        recurrent_dropout=0.3,
        return_sequences=True
    ))
    model.add(LSTM(512, return_sequences=True, recurrent_dropout=0.3,))
    model.add(LSTM(512))
    model.add(BatchNorm())
    model.add(Dropout(0.3))
    model.add(Dense(256))
    model.add(Activation('relu'))
    model.add(BatchNorm())
    model.add(Dropout(0.3))
    model.add(Dense(n_vocab))
    model.add(Activation('softmax'))
    model.compile(loss='categorical_crossentropy', optimizer='rmsprop')

```



```

model.load_weights('media/weights.hdf5')

return model

def generate_notes(model, network_input, pitchnames, n_vocab):

    start = numpy.random.randint(0, len(network_input)-1)

    int_to_note = dict((number, note) for number, note in enumerate(pitchnames))

    pattern = network_input[start]
    prediction_output = []

    for note_index in range(500):
        prediction_input = numpy.reshape(pattern, (1, len(pattern), 1))
        prediction_input = prediction_input / float(n_vocab)

        prediction = model.predict(prediction_input, verbose=0)

        index = numpy.argmax(prediction)
        result = int_to_note[index]
        prediction_output.append(result)

        pattern.append(index)
        pattern = pattern[1:len(pattern)]

    return prediction_output

def create_midi(prediction_output):
    offset = 0
    output_notes = []

    for pattern in prediction_output:
        if ( '.' in pattern) or pattern.isdigit():
            notes_in_chord = pattern.split('.')
            notes = []
            for current_note in notes_in_chord:
                new_note = note.Note(int(current_note))
                new_note.storedInstrument = instrument.Piano()
                notes.append(new_note)
            new_chord = chord.Chord(notes)
            new_chord.offset = offset
            output_notes.append(new_chord)
        else:
            new_note = note.Note(pattern)
            new_note.offset = offset
            new_note.storedInstrument = instrument.Piano()
            output_notes.append(new_note)

    offset += 0.5

```

```

midi_stream = stream.Stream(output_notes)

midi_stream.write('midi', fp='test_output.mid')

def my_view(request):
    message = ''

    if request.method == 'POST':
        generate()
        form = DocumentForm(request.POST, request.FILES)
        if form.is_valid():
            newdoc = Document(docfile=request.FILES['docfile'])
            newdoc.save()

            return redirect('my-view')
        else:
            message = 'The form is not valid. Fix the following error:'
    else:
        form = DocumentForm()

    documents = Document.objects.all()

    context = {'documents': documents, 'form': form, 'message': message}
    return render(request, 'list.html', context)

```

### 5.3 code explanation –

The code is taken from the views.py Django project directory. The code takes the model generated in the form of hd5f file. It uses the hd5f file to generate a new midi output file and stores it in the root directory of the Django project.

## 6. RESULT DISCUSSION

As we can see this is the output generated by the code. What happens is when the code gets executed, a link is generated in the output window which redirects the user to a website where two options are provided:

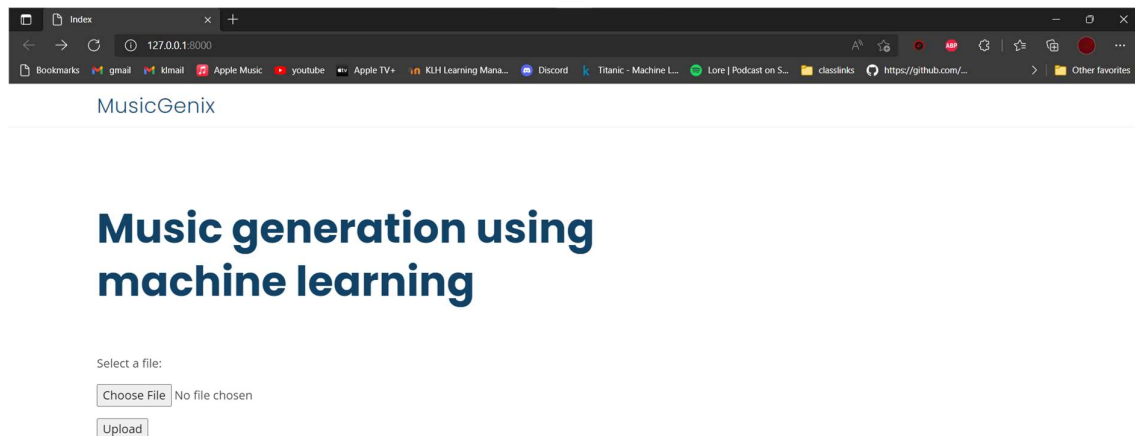
- Choose file:

This option allows the use to choose a file for uploading.



- Upload file:

This option uploads the file after which the code trains the dataset and returns the output (music file).

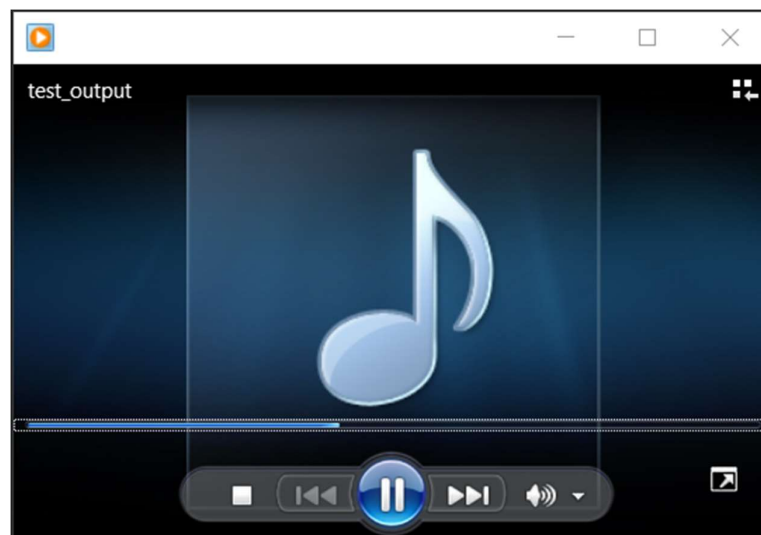
### Output Screenshots



*(Website hosted using Django framework)*

 manage.py	3/19/2022 09:31	Python Source File	1 KB
 test_output.mid	3/19/2022 11:12	MIDI Sequence	6 KB

*(Output file generated)*



*(Playable music file generated)*

**NOTE: The music file generated as output is a “.midi” file**

## **8. CONCLUSION AND FUTURE WORK**

- The project is fully functional and successfully generates a new music file
- Django or hosting it in a web server for testing will be appropriate representation of the intended real-life use
- Web-Interface can be changed/ modified to enhance the user-experience.
- Additional options can be added like “genre” to get the output based on chosen genre.

## 9. REFERENCES

- JamBot:

Gino Brunner, Yuyi Wang, Roger Wattenhofer and Jonas Wiesendanger\*  
Department of Information Technology and Electrical Engineering ETH Zürich  
Switzerland

- Convolutional generative adversarial networks with binary neurons for polyphonic music generation:

Hao-Wen Dong and Yi-Hsuan Yang Research Center for IT innovation, Academia Sinica, Taipei, Taiwan

- MuseGAN:

Hao-Wen Dong\* , 1 Wen-Yi Hsiao\* , 1,2 Li-Chia Yang,1 Yi-Hsuan Yang1  
1Research Center for Information Technology Innovation, Academia Sinica,  
Taipei, Taiwan 2Department of Computer Science, National Tsing Hua  
University, Hsinchu, Taiwan

- Lo-Fi Hip Hop Generation:

Zachary Katsnelson