Mehwish Faqir

2030-0260

lab12task2

```
1
2
3  import numpy as np # linear algebra
4  import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
5
6  # Input data files are available in the read-only "../input/" directory
7  # For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory
8
9  import os
10 for dirname, _, filenames in os.walk('/kaggle/input'):
11     for filename in filenames:
12         print(os.path.join(dirname, filename))
13
14 # You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Sa
15 # You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
16
17 #Installing dependencies
18 !pip install music21
19 !apt-get install -y lilypond
```

```
sbin/ldconfig.real: File /usr/lib/x86_64-linux-gnu/libnvidia-ptxjitcompiler.so.450.119.04 is empty, not checked.
sbin/ldconfig.real: File /usr/lib/x86_64-linux-gnu/libnvidia-allocator.so.1 is empty, not checked.
sbin/ldconfig.real: File /usr/lib/x86_64-linux-gnu/libGLX_nvidia.so.450.119.04 is empty, not checked.
sbin/ldconfig.real: File /usr/lib/x86_64-linux-gnu/libnvoptix.so.1 is empty, not checked.
sbin/ldconfig.real: File /usr/lib/x86_64-linux-gnu/libnvidia-glcore.so.450.119.04 is empty, not checked.
sbin/ldconfig.real: File /usr/lib/x86_64-linux-gnu/libnvidia-cfg.so.1 is empty, not checked.
sbin/ldconfig.real: File /usr/lib/x86_64-linux-gnu/libvdpau_nvidia.so.1 is empty, not checked.
sbin/ldconfig.real: File /usr/lib/x86_64-linux-gnu/libnvidia-eglcore.so.450.119.04 is empty, not checked.
```

```
#Importing Libraries
import tensorflow
import numpy as np
import pandas as pd
from collections import Counter
import random
import IPython
from IPython.display import Image, Audio
import music21
from music21 import *
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
import tensorflow.keras.backend as K
from tensorflow.keras.optimizers import Adamax
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
%matplotlib inline
import sys
import warnings
warnings.filterwarnings("ignore")
warnings.simplefilter("ignore")
np.random.seed(42)
```

```
2021-10-11 11:17:01.123545: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libcudar
```

LOADING DATA

For this project, I will be using MIDI files of classical piano music. The dataset includes various artists. I will be working with Frédéric Chopin's compositions.

- First of all, I make a list of all the songs in the Chopin folder parsed as music21 stream.
- Then I will be creating a function to extract chords and notes out of the data creating a corpus.

Laoding and parsing data

```
#Loading the list of chopin's midi files as stream
filepath = "../input/classical-music-midi/chopin/"
#Getting midi files
all_midis= []
for i in os.listdir(filepath):
    if i.endswith(".mid"):
        tr = filepath+i
        midi = converter.parse(tr)
        all_midis.append(midi)
```

Next, I get the components out of these streams of MIDI files. The midi files only have the piano included as mentioned in the dataset. So the components of the file would be either piano chords or piano notes.

Note: The musical notes are the building blocks of the music. It pertains to a pitch associated with a specific audio vibration. Western music utilizes twelve musical notes.

Chord: A group of notes that sound good together is a chord.

The music21 stream that was created in the above cell contains both, chords and notes, we will extract them in the form of notes and obtain a series of notes in the musical composition.

The function to get the notes:

```
#Helping function
def extract_notes(file):
    notes = []
    pick = None
    for j in file:
        songs = instrument.partitionByInstrument(j)
        for part in songs.parts:
            pick = part.recurse()
            for element in pick:
                if isinstance(element, note.Note):
                    notes.append(str(element.pitch))
                elif isinstance(element, chord.Chord):
                    notes.append(".".join(str(n) for n in element.normalOrder))

    return notes
#Getting the list of notes as Corpus
Corpus= extract_notes(all_midis)
print("Total notes in all the Chopin midis in the dataset:", len(Corpus))
```

```
Total notes in all the Chopin midis in the dataset: 57887
```

```
print("First fifty values in the Corpus:", Corpus[:50])
```

```
First fifty values in the Corpus: ['6.10.1', 'F#2', 'F3', 'G#3', 'F#3', 'B-3', 'C#3', '6.10.1', 'F#2', 'F3', 'G#3', 'F#3', 'B-3', 'C#3',
```

All these values indicate the notes, as mentioned above.

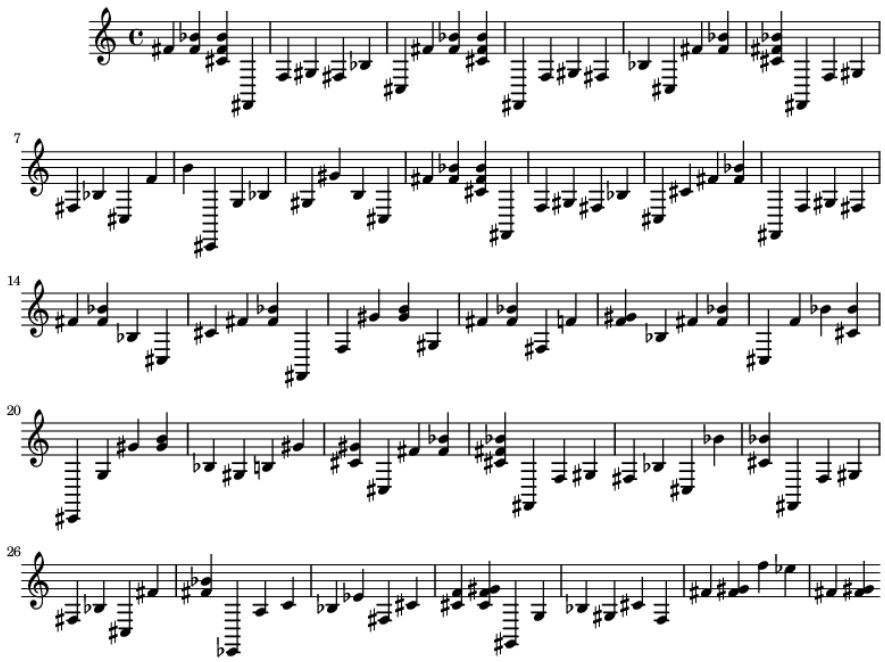
Printing the music sheet

```
#First Lets write some functions that we need to look into the data
def show(music):
    display(Image(str(music.write("lily.png"))))

def chords_n_notes(Snippet):
    Melody = []
    offset = 0 #Incremental
    for i in Snippet:
        #If it is chord
        if ("." in i or i.isdigit()):
            chord_notes = i.split(".") #Seperating the notes in chord
            notes = []
            for j in chord_notes:
                inst_note=int(j)
                note_snip = note.Note(inst_note)
                notes.append(note_snip)
                chord_snip = chord.Chord(notes)
                chord_snip.offset = offset
                Melody.append(chord_snip)
        # pattern is a note
        else:
            note_snip = note.Note(i)
            note_snip.offset = offset
            Melody.append(note_snip)
        # increase offset each iteration so that notes do not stack
        offset += 1
    Melody_midi = stream.Stream(Melody)
    return Melody_midi

Melody_Snippet = chords_n_notes(Corpus[:100])
show(Melody_Snippet)
```

```
Changing working directory to: `/tmp/music21'
Processing `/tmp/music21/tmpa4rhfc3w.ly'
Parsing...
Interpreting music...[8][16][24][32][32]
Preprocessing graphical objects...
Calculating line breaks...
Drawing systems...
Layout output to `tmpa4rhfc3w.ly.eps'...
Converting to PNG...
Layout output to `tmpa4rhfc3w.ly-1.eps'...
Layout output to `tmpa4rhfc3w.ly-2.eps'...
Layout output to `tmpa4rhfc3w.ly-3.eps'...
Layout output to `tmpa4rhfc3w.ly-4.eps'...
Layout output to `tmpa4rhfc3w.ly-5.eps'...
Writing tmpa4rhfc3w.ly-systems.texi...
Writing tmpa4rhfc3w.ly-systems.tex...
Writing tmpa4rhfc3w.ly-systems.count...
Success: compilation successfully completed
```



Playing the above sheet music

As I could not play a midi file on the Kaggle interface, I have created a ".wav" filetype of the same outside of this code. I am using it to create an audio interface. Let us have a listen to the data corpus.

```
#to play audio or corpus
print("Sample Audio From Data")
IPython.display.Audio("../input/music-generated-lstm/Corpus_Snippet.wav")
```

```
Sample Audio From Data
```

0:00 / 0:51

Examine all the notes in the Corpus

```
#Creating a count dictionary
count_num = Counter(Corpus)
print("Total unique notes in the Corpus:", len(count_num))
```

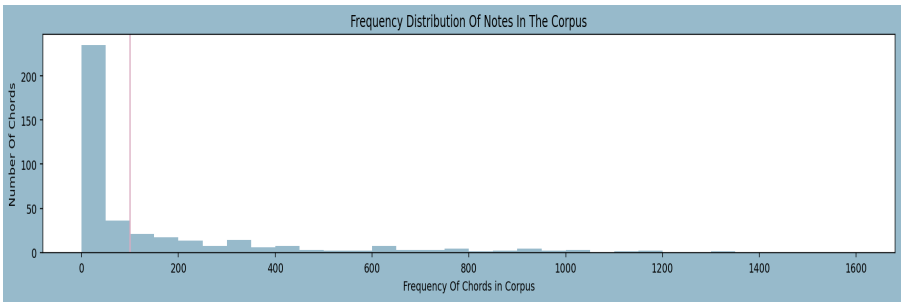
```
Total unique notes in the Corpus: 397
```

```
#Exploring the notes dictionary
Notes = list(count_num.keys())
Recurrence = list(count_num.values())
#Average recurrenc for a note in Corpus
def Average(lst):
    return sum(lst) / len(lst)
print("Average recurrenc for a note in Corpus:", Average(Recurrence))
print("Most frequent note in Corpus appeared:", max(Recurrence), "times")
print("Least frequent note in Corpus appeared:", min(Recurrence), "time")
```

```
Average recurrenc for a note in Corpus: 145.8110831234257
Most frequent note in Corpus appeared: 1627 times
Least frequent note in Corpus appeared: 1 time
```

Clearly, there are some very rare notes in the melody; some so rare that it was played only once in the whole data. This would create a lot of problems. (I did run into most of them while writing this piece) To spare us the error reports, let us have a look at the frequency of the notes. And for simplicity, I shall be eliminating some of the least occurring notes. I am sure Chopin wouldn't mind me messing with his masterpiece for science or would he? Either way, I may never know!

```
# Plotting the distribution of Notes
plt.figure(figsize=(18,3),facecolor="#97BACB")
bins = np.arange(0,(max(Recurrence)), 50)
plt.hist(Recurrence, bins=bins, color="#97BACB")
plt.axvline(x=100,color="#DBACC1")
plt.title("Frequency Distribution Of Notes In The Corpus")
plt.xlabel("Frequency Of Chords in Corpus")
plt.ylabel("Number Of Chords")
plt.show()
```



I have decided, I will be taking out the notes that were played less than 100 times. I mean, if Chopin liked them he would have played it a lot more often. So I create a list of rare notes in the next section.

```
#Getting a list of rare chords
rare_note = []
for index, (key, value) in enumerate(count_num.items()):
    if value < 100:
        m =  key
        rare_note.append(m)

print("Total number of notes that occur less than 100 times:", len(rare_note))
```

```
Total number of notes that occur less than 100 times: 271
```

```
#Eleminating the rare notes
for element in Corpus:
    if element in rare_note:
        Corpus.remove(element)

print("Length of Corpus after elemination the rare notes:", len(Corpus))
```

```
Length of Corpus after elemination the rare notes: 53712
```

```
# Storing all the unique characters present in my corpus to bult a mapping dic.
symb = sorted(list(set(Corpus)))

L_corpus = len(Corpus) #length of corpus
L_symb = len(symb) #length of total unique characters

#Building dictionary to access the vocabulary from indices and vice versa
mapping = dict((c, i) for i, c in enumerate(symb))
reverse_mapping = dict((i, c) for i, c in enumerate(symb))

print("Total number of characters:", L_corpus)
print("Number of unique characters:", L_symb)
```

```
Total number of characters: 53712
Number of unique characters: 266
```

Encoding and Splitting the Corpus as Labels and Targets

```
#Splitting the Corpus in equal length of strings and output target
length = 40
features = []
targets = []
for i in range(0, L_corpus - length, 1):
    feature = Corpus[i:i + length]
    target = Corpus[i + length]
    features.append([mapping[j] for j in feature])
    targets.append(mapping[target])


L_datapoints = len(targets)
print("Total number of sequences in the Corpus:", L_datapoints)
```

```
Total number of sequences in the Corpus: 53672
```

```
# reshape X and normalize
X = (np.reshape(features, (L_datapoints, length, 1)))/ float(L_symb)
# one hot encode the output variable
y = tensorflow.keras.utils.to_categorical(targets)
```

Splitting Train and Seed datasets

```
#Taking out a subset of data to be used as seed
X_train, X_seed, y_train, y_seed = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
#Initialising the Model
model = Sequential()
#Adding layers
model.add(LSTM(512, input_shape=(X.shape[1], X.shape[2]), return_sequences=True))
model.add(Dropout(0.1))
model.add(LSTM(256))
model.add(Dense(256))
model.add(Dropout(0.1))
model.add(Dense(y.shape[1], activation='softmax'))
#Compiling the model for training
opt = Adamax(learning_rate=0.01)
model.compile(loss='categorical_crossentropy', optimizer=opt)

```

```
2021-10-11 11:22:26.554021: I tensorflow/compiler/jit/xla_cpu_device.cc:41] Not creating XLA devices, tf_xla_enable_xla_devices not set
2021-10-11 11:22:26.557517: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libcuda.
2021-10-11 11:22:26.594074: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:941] successful NUMA node read from SysFS had negativ
2021-10-11 11:22:26.594881: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1720] Found device 0 with properties:
pciBusID: 0000:00:04.0 name: Tesla P100-PCIE-16GB computeCapability: 6.0
coreClock: 1.3285GHz coreCount: 56 deviceMemorySize: 15.90GiB deviceMemoryBandwidth: 681.88GiB/s
2021-10-11 11:22:26.594975: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libcudar
2021-10-11 11:22:26.617347: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libcubla
2021-10-11 11:22:26.617445: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libcubla
2021-10-11 11:22:26.631473: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libcufft
2021-10-11 11:22:26.637487: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libcuran
2021-10-11 11:22:26.662421: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libcusol
2021-10-11 11:22:26.669506: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libcuspa
2021-10-11 11:22:26.669841: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libcudnn
2021-10-11 11:22:26.670000: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:941] successful NUMA node read from SysFS had negativ
2021-10-11 11:22:26.670742: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:941] successful NUMA node read from SysFS had negativ
```

```
2021-10-11 11:22:26.672337: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1862] Adding visible gpu devices: 0
2021-10-11 11:22:26.673444: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow binary is optimized with oneAPI Deep Ne
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2021-10-11 11:22:26.673656: I tensorflow/compiler/jit/xla_gpu_device.cc:99] Not creating XLA devices, tf_xla_enable_xla_devices not set
2021-10-11 11:22:26.673818: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:941] successful NUMA node read from SysFS had negativ
2021-10-11 11:22:26.674421: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1720] Found device 0 with properties:
pciBusID: 0000:00:04.0 name: Tesla P100-PCIE-16GB computeCapability: 6.0
coreClock: 1.3285GHz coreCount: 56 deviceMemorySize: 15.90GiB deviceMemoryBandwidth: 681.88GiB/s
2021-10-11 11:22:26.674465: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libcudar
2021-10-11 11:22:26.674492: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libcubla
2021-10-11 11:22:26.674510: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libcubla
2021-10-11 11:22:26.674528: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libcufft
2021-10-11 11:22:26.674554: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libcuran
2021-10-11 11:22:26.674581: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libcusol
2021-10-11 11:22:26.674601: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libcuspa
2021-10-11 11:22:26.674619: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libcudnn
2021-10-11 11:22:26.674696: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:941] successful NUMA node read from SysFS had negativ
2021-10-11 11:22:26.675343: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:941] successful NUMA node read from SysFS had negativ
2021-10-11 11:22:26.675871: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1862] Adding visible gpu devices: 0
2021-10-11 11:22:26.676907: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libcudar
2021-10-11 11:22:28.126215: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1261] Device interconnect StreamExecutor with strength 1
2021-10-11 11:22:28.126268: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1267]      0
2021-10-11 11:22:28.126280: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1280] 0:   N
2021-10-11 11:22:28.128644: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:941] successful NUMA node read from SysFS had negativ
2021-10-11 11:22:28.129422: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:941] successful NUMA node read from SysFS had negativ
2021-10-11 11:22:28.130083: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:941] successful NUMA node read from SysFS had negativ
2021-10-11 11:22:28.130711: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1406] Created TensorFlow device (/job:localhost/replica:0
```

```
#Model's Summary
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 40, 512)           1052672
_________________________________________________________________
dropout (Dropout)            (None, 40, 512)           0
_________________________________________________________________
lstm_1 (LSTM)                (None, 256)               787456
_________________________________________________________________
dense (Dense)                (None, 256)               65792
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 266)               68362
=================================================================
Total params: 1,974,282
Trainable params: 1,974,282
Non-trainable params: 0
_________________________________________________________________
```

```
#Training the Model
history = model.fit(X_train, y_train, batch_size=256, epochs=200)
```

```
Epoch 1/200
2021-10-11 11:22:29.171831: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:116] None of the MLIR optimization passes are e
2021-10-11 11:22:29.182449: I tensorflow/core/platform/profile_utils/cpu_utils.cc:112] CPU Frequency: 2000179999 Hz
2021-10-11 11:22:31.475252: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libcub
2021-10-11 11:22:32.222805: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libcub
2021-10-11 11:22:32.267251: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libcud
168/168 [==============================] - 13s 29ms/step - loss: 4.9002
Epoch 2/200
168/168 [==============================] - 5s 29ms/step - loss: 4.6921
Epoch 3/200
168/168 [==============================] - 5s 30ms/step - loss: 4.6850
Epoch 4/200
168/168 [==============================] - 5s 29ms/step - loss: 4.6382
Epoch 5/200
168/168 [==============================] - 5s 29ms/step - loss: 4.5880
Epoch 6/200
168/168 [==============================] - 5s 29ms/step - loss: 4.5722
Epoch 7/200
168/168 [==============================] - 5s 29ms/step - loss: 4.5714
Epoch 8/200
168/168 [==============================] - 5s 29ms/step - loss: 4.5634
Epoch 9/200
168/168 [==============================] - 5s 30ms/step - loss: 4.5575
Epoch 10/200
168/168 [==============================] - 5s 29ms/step - loss: 4.5603
Epoch 11/200
```

```
168/168 [==============================] - 5s 30ms/step - loss: 4.5546
Epoch 12/200
168/168 [==============================] - 5s 29ms/step - loss: 4.5539
Epoch 13/200
168/168 [==============================] - 5s 29ms/step - loss: 4.5599
Epoch 14/200
168/168 [==============================] - 5s 30ms/step - loss: 4.5414
Epoch 15/200
168/168 [==============================] - 5s 29ms/step - loss: 4.5435
Epoch 16/200
168/168 [==============================] - 5s 30ms/step - loss: 4.5047
Epoch 17/200
168/168 [==============================] - 5s 29ms/step - loss: 4.4722
Epoch 18/200
168/168 [==============================] - 5s 30ms/step - loss: 4.4684
Epoch 19/200
168/168 [==============================] - 5s 29ms/step - loss: 4.4658
Epoch 20/200
168/168 [==============================] - 5s 29ms/step - loss: 4.4497
Epoch 21/200
168/168 [==============================] - 5s 29ms/step - loss: 4.4505
Epoch 22/200
168/168 [==============================] - 5s 29ms/step - loss: 4.4483
Epoch 23/200
168/168 [==============================] - 5s 30ms/step - loss: 4.4532
Epoch 24/200
168/168 [==============================] - 5s 29ms/step - loss: 4.4307
Epoch 25/200
168/168 [==============================] - 5s 29ms/step - loss: 4.4320
Epoch 26/200
168/168 [==============================] - 5s 29ms/step - loss: 4.4231
```

EVALUATING MODELS

Now that I have my model trained on the MIDI files of piano music, let us see how it performs.

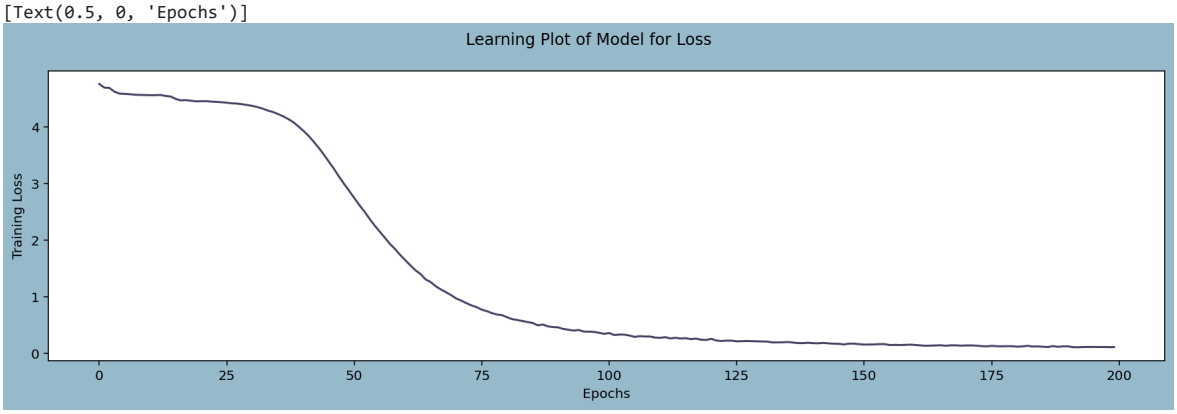
To evaluate my model, I shall be having a look at:

- The performance of the model via Learning Curves
- The melody created

Plotting the learning curve for the loss function

```
#Plotting the learnings
history_df = pd.DataFrame(history.history)
fig = plt.figure(figsize=(15,4), facecolor="#97BACB")
fig.suptitle("Learning Plot of Model for Loss")
pl=sns.lineplot(data=history_df["loss"],color="#444160")
pl.set(ylabel ="Training Loss")
pl.set(xlabel ="Epochs")
```

```
[Text(0.5, 0, 'Epochs')]
```



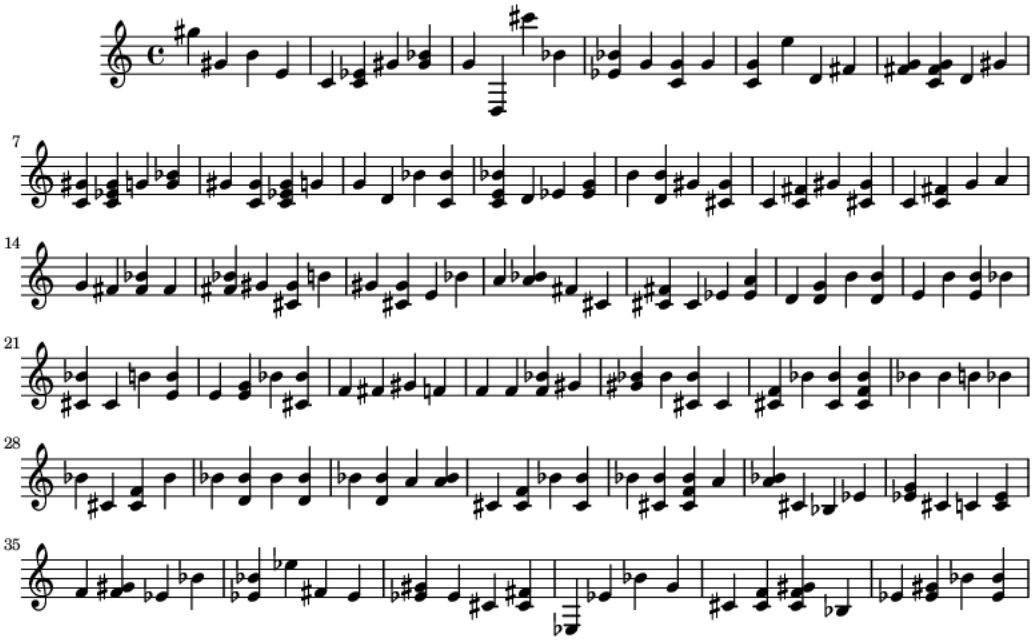
Generating the Melody

A function to obtain the generated music

```
def Malody_Generator(Note_Count):
    seed = X_seed[np.random.randint(0,len(X_seed)-1)]
    Music = ""
    Notes_Generated=[]
    for i in range(Note_Count):
        seed = seed.reshape(1,length,1)
        prediction = model.predict(seed, verbose=0)[0]
        prediction = np.log(prediction) / 1.0 #diversity
        exp_preds = np.exp(prediction)
        prediction = exp_preds / np.sum(exp_preds)
        index = np.argmax(prediction)
        index_N = index/ float(L_symb)
        Notes_Generated.append(index)
        Music = [reverse_mapping[char] for char in Notes_Generated]
        seed = np.insert(seed[0],len(seed[0]),index_N)
        seed = seed[1:]
    #Now, we have music in form or a list of chords and notes and we want to be a midi file.
    Melody = chords_n_notes(Music)
    Melody_midi = stream.Stream(Melody)
    return Music,Melody_midi


#getting the Notes and Melody created by the model
Music_notes, Melody = Malody_Generator(100)
show(Melody)
```

```
Changing working directory to: `/tmp/music21'
Processing `/tmp/music21/tmpnt8eigub.ly'
Parsing...
Interpreting music...[8][16][24][32][40]
Preprocessing graphical objects...
Calculating line breaks...
Drawing systems...
Layout output to `tmpnt8eigub.ly.eps'...
Converting to PNG...
Layout output to `tmpnt8eigub.ly-1.eps'...
Layout output to `tmpnt8eigub.ly-2.eps'...
Layout output to `tmpnt8eigub.ly-3.eps'...
Layout output to `tmpnt8eigub.ly-4.eps'...
Layout output to `tmpnt8eigub.ly-5.eps'...
Layout output to `tmpnt8eigub.ly-6.eps'...
Writing tmpnt8eigub.ly-systems.texi...
Writing tmpnt8eigub.ly-systems.tex...
Writing tmpnt8eigub.ly-systems.count...
Success: compilation successfully completed
```



This sure looks like music! To check if it sounds like music we have to listen to the MIDI file. Playing midi is crumblesome. I have saved and converted a few generated melodies to ".wav" format outside of this notebook. So let us have a listen.

Melody Generated Sample 1

```
#To save the generated melody
Melody.write('midi','Melody_Generated.mid')
#to play audio or corpus
IPython.display.Audio("../input/music-generated-lstm/Melody_Generated 2.wav")
```

0:00 / 0:52

Melody Generated Sample 2

Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.