##Importing Required Packages

```
!pip install mido

Collecting mido
  Downloading mido-1.3.3-py3-none-any.whl.metadata (6.4 kB)
Requirement already satisfied: packaging in
/usr/local/lib/python3.12/dist-packages (from mido) (25.0)
Downloading mido-1.3.3-py3-none-any.whl (54 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0.0/54.6 kB ? eta -:--:--
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 54.6/54.6 kB 1.8 MB/s eta
0:00:00
ido
Successfully installed mido-1.3.3

import mido
from mido import MidiFile, MidiTrack, Message
from keras.layers import LSTM, Dense, Activation, Dropout, Flatten
from keras.preprocessing import sequence
from keras.models import Sequential
from tensorflow.keras.optimizers import Adam
from sklearn.preprocessing import MinMaxScaler, StandardScaler
import numpy as np
```

Pada bagian ini, saya menginstal dan memanggil library yang saya butuhkan untuk menjalankan seluruh proses pemodelan:

- **mido** digunakan untuk membaca, memproses, dan menulis file MIDI.
- **Keras/TensorFlow** digunakan untuk membangun dan melatih model LSTM.
- **scikit-learn (sklearn)** digunakan untuk melakukan scaling data agar lebih stabil saat diproses oleh model.
- **NumPy** digunakan untuk manipulasi array dan pengolahan data numerik.

##Load MID file/files

```
!unzip /content/chillhopdata.zip -d chillhop

Archive:  /content/chillhopdata.zip
  inflating: chillhop/1.mid
  inflating: chillhop/10.mid
  inflating: chillhop/11.mid
  inflating: chillhop/12.mid
  inflating: chillhop/13.mid
  inflating: chillhop/14.mid
  inflating: chillhop/15.mid
  inflating: chillhop/16.mid
  inflating: chillhop/17.mid
  inflating: chillhop/18.mid
  inflating: chillhop/19.mid
```

```
  inflating: chillhop/2.mid
  inflating: chillhop/20.mid
  inflating: chillhop/3.mid
  inflating: chillhop/4.mid
  inflating: chillhop/5.mid
  inflating: chillhop/6.mid
  inflating: chillhop/7.mid
  inflating: chillhop/8.mid
  inflating: chillhop/9.mid
  inflating: chillhop/Cymatics - Eternity MIDI 1 - C Maj.mid
  inflating: chillhop/Cymatics - Eternity MIDI 10 - F Min.mid
  inflating: chillhop/Cymatics - Eternity MIDI 11 - A Maj.mid
  inflating: chillhop/Cymatics - Eternity MIDI 12 - A Min.mid
  inflating: chillhop/Cymatics - Eternity MIDI 13 - A Min.mid
  inflating: chillhop/Cymatics - Eternity MIDI 14 - A Min.mid
  inflating: chillhop/Cymatics - Eternity MIDI 15 - A Maj.mid
  inflating: chillhop/Cymatics - Eternity MIDI 16 - A Maj.mid
  inflating: chillhop/Cymatics - Eternity MIDI 17 - A Maj.mid
  inflating: chillhop/Cymatics - Eternity MIDI 18 - A Maj.mid
  inflating: chillhop/Cymatics - Eternity MIDI 19 - A Min.mid
  inflating: chillhop/Cymatics - Eternity MIDI 2 - C Min.mid
  inflating: chillhop/Cymatics - Eternity MIDI 20 - A Min.mid
  inflating: chillhop/Cymatics - Eternity MIDI 21 - A Min.mid
  inflating: chillhop/Cymatics - Eternity MIDI 22 - B Min.mid
  inflating: chillhop/Cymatics - Eternity MIDI 3 - D Maj.mid
  inflating: chillhop/Cymatics - Eternity MIDI 4 - D Maj.mid
  inflating: chillhop/Cymatics - Eternity MIDI 5 - D Min.mid
  inflating: chillhop/Cymatics - Eternity MIDI 6 - D Min.mid
  inflating: chillhop/Cymatics - Eternity MIDI 7 - E Min.mid
  inflating: chillhop/Cymatics - Eternity MIDI 8 - F Maj.mid
  inflating: chillhop/Cymatics - Eternity MIDI 9 - F Min.mid
  inflating: chillhop/Cymatics - Lofi MIDI 1 - C Maj.mid
  inflating: chillhop/Cymatics - Lofi MIDI 10 - D Maj.mid
  inflating: chillhop/Cymatics - Lofi MIDI 11 - E Maj.mid
  inflating: chillhop/Cymatics - Lofi MIDI 12 - E Min.mid
  inflating: chillhop/Cymatics - Lofi MIDI 13 - E Min.mid
  inflating: chillhop/Cymatics - Lofi MIDI 14 - F Min.mid
  inflating: chillhop/Cymatics - Lofi MIDI 15 - F Maj.mid
  inflating: chillhop/Cymatics - Lofi MIDI 16 - F Maj.mid
  inflating: chillhop/Cymatics - Lofi MIDI 17 - G Maj.mid
  inflating: chillhop/Cymatics - Lofi MIDI 18 - G Maj.mid
  inflating: chillhop/Cymatics - Lofi MIDI 19 - G Maj.mid
  inflating: chillhop/Cymatics - Lofi MIDI 2 - C Maj.mid
  inflating: chillhop/Cymatics - Lofi MIDI 20 - G Min.mid
  inflating: chillhop/Cymatics - Lofi MIDI 21 - A Min.mid
  inflating: chillhop/Cymatics - Lofi MIDI 22 - B Min.mid
  inflating: chillhop/Cymatics - Lofi MIDI 3 - C Min.mid
  inflating: chillhop/Cymatics - Lofi MIDI 4 - D Maj.mid
  inflating: chillhop/Cymatics - Lofi MIDI 5 - D Maj.mid
```

```
  inflating: chillhop/Cymatics - Lofi MIDI 6 - D Min.mid
  inflating: chillhop/Cymatics - Lofi MIDI 7 - D Maj.mid
  inflating: chillhop/Cymatics - Lofi MIDI 8 - D Maj.mid
  inflating: chillhop/Cymatics - Lofi MIDI 9 - D Maj.mid
  inflating: chillhop/E-Piano Chords MIDI.mid
  inflating: chillhop/E-Piano MIDI (2).mid
  inflating: chillhop/E-Piano MIDI.mid
  inflating: chillhop/Lofi Piano MIDI.mid
  inflating: chillhop/Piano 1 MIDI.mid
  inflating: chillhop/Piano 2 MIDI.mid
  inflating: chillhop/Piano Chords MIDI (2).mid
  inflating: chillhop/Piano Chords MIDI (3).mid
  inflating: chillhop/Piano Chords MIDI.mid
  inflating: chillhop/Piano MIDI (2).mid
  inflating: chillhop/Piano MIDI (3).mid
  inflating: chillhop/Piano MIDI (4).mid
  inflating: chillhop/Piano MIDI (5).mid
  inflating: chillhop/Piano MIDI (6).mid
  inflating: chillhop/Piano MIDI (7).mid
  inflating: chillhop/Piano MIDI (8).mid
  inflating: chillhop/Piano MIDI 1.mid
  inflating: chillhop/Piano MIDI 2.mid
  inflating: chillhop/Piano MIDI.mid
  inflating: chillhop/Rhodes MIDI (2).mid
  inflating: chillhop/Rhodes MIDI (3).mid
  inflating: chillhop/Rhodes MIDI (4).mid
  inflating: chillhop/Rhodes MIDI (5).mid
  inflating: chillhop/Rhodes MIDI (6).mid
  inflating: chillhop/Rhodes MIDI (7).mid
  inflating: chillhop/Rhodes MIDI (8).mid
  inflating: chillhop/Rhodes MIDI (9).mid
  inflating: chillhop/Rhodes MIDI.mid
  inflating: chillhop/merge_from_ofoct.mid
```

```python
import os

notes = []
for song in os.listdir("/content/chillhop"):
  mid = MidiFile('/content/chillhop/' + song)
  for msg in mid:
    if not msg.is_meta and msg.channel == 0 and msg.type =='note_on':
      data = msg.bytes()
      notes.append(data[1])
```

Pada bagian ini, saya membuka setiap file MIDI dalam folder dan mengambil pitch dari setiap pesan note_on. Saya hanya menyimpan pitch tersebut ke dalam list notes.

##**Scale Data**

```
scaler = MinMaxScaler()
notes = list(scaler.fit_transform(np.array(notes).reshape(-1, 1)))
```

Agar model LSTM lebih stabil, saya mengubah range nada menjadi 0–1 menggunakan MinMaxScaler. Saya juga mengubah kembali hasil scaling menjadi list biasa.

##Create Train Data

```
notes = [list(note) for note in notes]

X = []
y = []

n_prev=30

for i in range(len(notes) - n_prev):
  X.append(notes[i:i+n_prev])
  y.append(notes[i+n_prev])

X_test = X[-300:]
X = X[:-300]
y = y[:-300]
```

Pada bagian ini, saya membentuk data dalam format sekuensial. Setiap 30 nada sebelumnya saya gunakan sebagai input (`X`), sedangkan nada ke-31 saya gunakan sebagai target (`y`). Dengan pendekatan ini, model belajar memprediksi nada berikutnya berdasarkan pola dari 30 nada sebelumnya.

Setelah data sekuensial terbentuk, saya memisahkan **300 sampel terakhir** sebagai data uji. Sisa datanya tetap digunakan untuk proses training agar model dapat belajar dari mayoritas pola nada yang tersedia.

##Build LSTM

```
model = Sequential()
model.add(LSTM(256, input_shape=(n_prev, 1), return_sequences=True))
model.add(Dropout(0.6))
model.add(LSTM(128, input_shape=(n_prev, 1), return_sequences=True))
model.add(Dropout(0.6))
model.add(LSTM(64, input_shape=(n_prev, 1), return_sequences=False))
model.add(Dropout(0.6))
model.add(Dense(1))
model.add(Activation('linear'))
model.summary()

optimizer = Adam(learning_rate=0.001)
model.compile(loss='mse', optimizer=optimizer)
```

```
Model: "sequential_2"
```

```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ lstm_6 (LSTM)                   │ (None, 30, 256)        │       264,192 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dropout_6 (Dropout)             │ (None, 30, 256)        │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ lstm_7 (LSTM)                   │ (None, 30, 128)        │       197,120 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dropout_7 (Dropout)             │ (None, 30, 128)        │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ lstm_8 (LSTM)                   │ (None, 64)             │        49,408 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dropout_8 (Dropout)             │ (None, 64)             │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_2 (Dense)                 │ (None, 1)              │            65 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ activation_2 (Activation)       │ (None, 1)              │             0 │
└─────────────────────────────────┴────────────────────────┴───────────────┘

 Total params: 510,785 (1.95 MB)

 Trainable params: 510,785 (1.95 MB)

 Non-trainable params: 0 (0.00 B)
```

Model yang saya bangun terdiri dari beberapa lapisan yang disusun bertingkat untuk menangkap pola nada secara mendalam:

- **LSTM 256 unit** sebagai lapisan pertama untuk menangkap pola jangka panjang.
- **Dropout 60%** untuk mencegah overfitting.

- **LSTM 128 unit** sebagai lapisan kedua untuk memperhalus representasi fitur.
- **Dropout 60%** kembali digunakan untuk menjaga generalisasi model.
- **LSTM 64 unit** sebagai lapisan terakhir sebelum output.
- **Dense(1)** digunakan untuk menghasilkan satu nilai pitch sebagai output.
- **Aktivasi linear** diterapkan karena model ini melakukan prediksi nilai numerik (regresi).

Untuk proses kompilasi, saya menggunakan:

- **Optimizer Adam** karena stabil dan efisien untuk model sekuensial.
- **Learning rate 0.001** agar proses pembelajaran berjalan seimbang.
- **Loss function MSE (Mean Squared Error)** karena model melakukan regresi pada nilai pitch.

##**Training**

```
model.fit(np.array(X), np.array(y), batch_size=16, epochs=10,
verbose=1)

Epoch 1/10
195/195 ━━━━━━━━━━━━━━━━━━━━ 31s 133ms/step - loss: 0.0306
Epoch 2/10
195/195 ━━━━━━━━━━━━━━━━━━━━ 37s 113ms/step - loss: 0.0191
Epoch 3/10
195/195 ━━━━━━━━━━━━━━━━━━━━ 42s 118ms/step - loss: 0.0172
Epoch 4/10
195/195 ━━━━━━━━━━━━━━━━━━━━ 23s 118ms/step - loss: 0.0165
Epoch 5/10
195/195 ━━━━━━━━━━━━━━━━━━━━ 40s 114ms/step - loss: 0.0162
Epoch 6/10
195/195 ━━━━━━━━━━━━━━━━━━━━ 41s 113ms/step - loss: 0.0159
Epoch 7/10
195/195 ━━━━━━━━━━━━━━━━━━━━ 23s 117ms/step - loss: 0.0150
Epoch 8/10
195/195 ━━━━━━━━━━━━━━━━━━━━ 40s 110ms/step - loss: 0.0158
Epoch 9/10
195/195 ━━━━━━━━━━━━━━━━━━━━ 23s 115ms/step - loss: 0.0147
Epoch 10/10
195/195 ━━━━━━━━━━━━━━━━━━━━ 22s 114ms/step - loss: 0.0153

<keras.src.callbacks.history.History at 0x7c2ce741e0c0>
```

Pada bagian ini, saya melakukan proses pelatihan model dengan beberapa pengaturan sebagai berikut:

- **Batch size 16**, sehingga model memperbarui bobot setiap memproses 16 sampel.
- **Epoch 10**, yaitu jumlah iterasi penuh model dalam mempelajari seluruh dataset.
- **Input berupa array NumPy**, karena format ini dibutuhkan oleh Keras untuk memproses data sekuensial dengan LSTM.

Dengan pengaturan tersebut, model belajar mengenali pola dari data nada secara bertahap berdasarkan input yang telah disiapkan.

##Generating & Saving LSTM Music

```
prediction = model.predict(np.array(X_test))
prediction = np.squeeze(prediction)
prediction = np.squeeze(scaler.inverse_transform(prediction.reshape(-
1,1)))
prediction = [int(i) for i in prediction]

mid = MidiFile()
track = MidiTrack()
t = 0
for note in prediction:
    # 147 means note_on
    # 67 is velosity
    note = np.asarray([147, note, 67])
    bytes = note.astype(int)
    msg = Message.from_bytes(bytes[0:3])
    t += 1
    msg.time = t
    track.append(msg)
mid.tracks.append(track)
mid.save('LSTM_music.mid')

10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 176ms/step
```

Pada tahap prediksi, model menghasilkan nilai pitch dalam bentuk skala 0–1 karena sebelumnya data telah dinormalisasi. Nilai prediksi tersebut kemudian saya kembalikan ke skala pitch asli menggunakan *inverse transform*. Setelah itu, saya mengonversi setiap nilai menjadi integer agar valid sebagai pitch MIDI.

Pada tahap akhir, saya membuat sebuah objek MIDI baru. Untuk setiap pitch hasil prediksi, saya membuat pesan `note_on` dan mengatur nilai waktu (*timestamp*) sehingga nada dimainkan secara berurutan. Seluruh pesan MIDI tersebut saya masukkan ke dalam satu track, lalu saya simpan sebagai file **LSTM_music.mid**.

Melalui proses ini, saya berhasil menghasilkan musik MIDI berdasarkan prediksi model LSTM yang telah dilatih.