**Code:**

from pydub import AudioSegment import numpy as np

import matplotlib.pyplot as plt

from scipy.signal import convolve

# Load MP3 file

audio = AudioSegment.from\_mp3("/content/Yeh Jeevan Hai (Piya Ka Ghar) -(Raag.Fm).mp3") # Convert to mono and extract raw samples

audio = audio.set\_channels(1)

samples = np.array(audio.get\_array\_of\_samples()).astype(np.float32)

# Define convolution kernel

kernel = np.array([1, 0, 1, 0, ], dtype=np.float32)

# Perform convolution

convoluted = convolve(samples, kernel, mode='same')

# Normalize to int16 range for saving

convoluted = convoluted / np.max(np.abs(convoluted)) convoluted\_int16 = (convoluted \* 32767).astype(np.int16)

# Save as WAV using PyDub convoluted\_audio = AudioSegment( convoluted\_int16.tobytes(),

frame\_rate=audio.frame\_rate,

sample\_width=2, # 16-bit samples = 2 bytes channels=1

)

# Export to file

convoluted\_audio.export("output\_convoluted.wav", format="wav") print("Convoluted audio saved as 'output\_convoluted.wav'.")

# Plot original and convoluted signals (optional)

samples\_norm = samples / np.max(np.abs(samples)) convoluted\_norm= convoluted

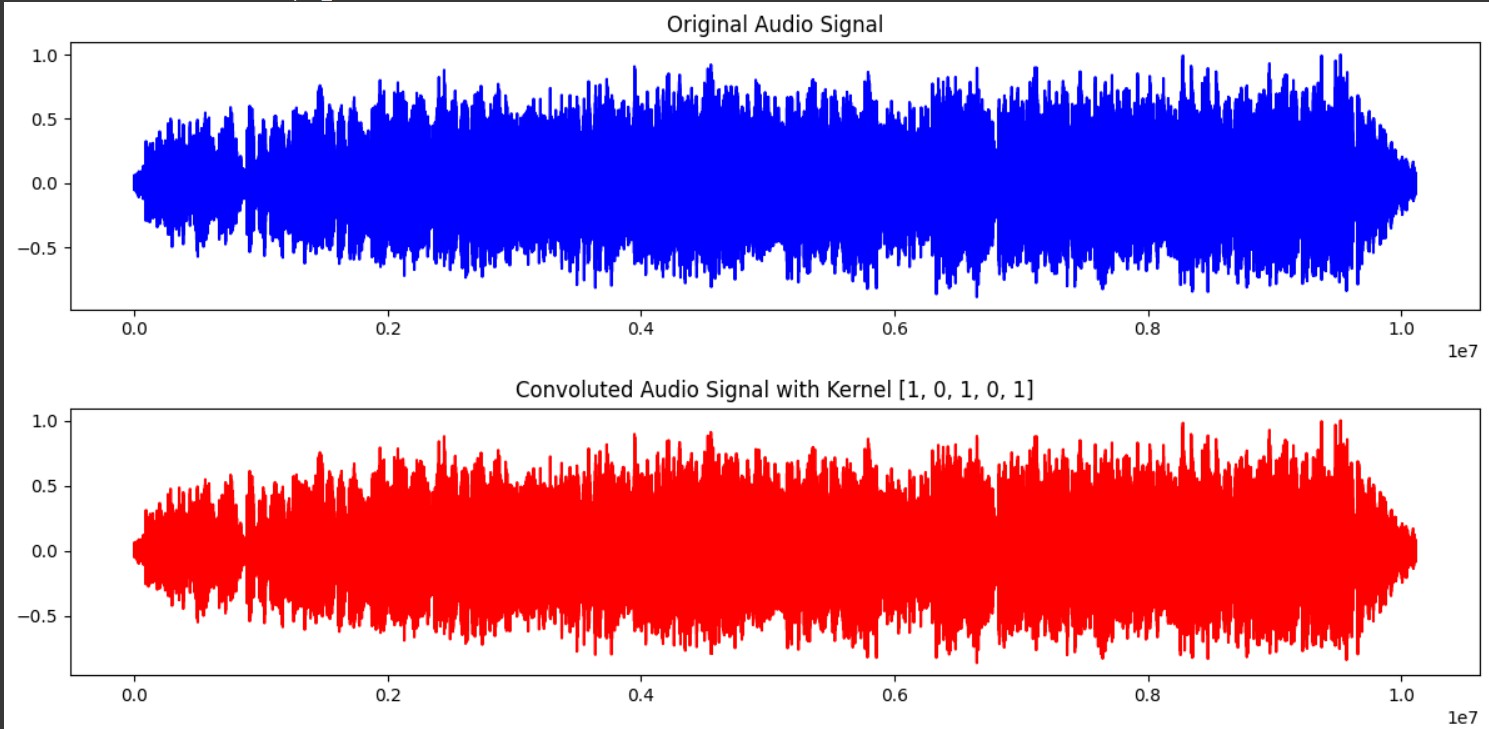
plt.figure(figsize=(12, 6))

plt.subplot(2, 1, 1)

plt.plot(samples\_norm, color='blue') plt.title("Original Audio Signal") plt.subplot(2, 1, 2) plt.plot(convoluted\_norm, color='red')

plt.title("Convoluted Audio Signal with Kernel [1, 0, 1, 0, 1]") plt.tight\_layout()

plt.show()

**Output:**

**Conclusion:**This experiment showed that convolution can modify audio waveforms, producing effects like echo and filtering. By converting audio to mono, extracting samples, and applying a predefined kernel, we created and compared altered and original signal.