**Sentongo Hamza**

**Master’s in Data Science**

Question 1

A green sound wave graph

AI-generated content may be incorrect.

I successfully removed the mean from this audio file, and I scaled successfully the amplitude between -1 and 1. The resulting waveform is centered around zero with normalized amplitude as shown in the above figure. The purpose of this is to standardize the amplitude for consistent processing.

**Code**

import numpy as np

import matplotlib.pyplot as plt

from scipy.io import wavfile

Fs, y\_initial = wavfile.read('./Kuusi.wav')

y = y\_initial.astype(np.float64)

y\_mean\_removed = y - np.mean(y)

y = y\_mean\_removed / np.max(np.abs(y\_mean\_removed))

duration = len(y) / Fs

time = np.linspace(0, duration, len(y))

plt.plot(time, y, label='Normalized Signal', color='green', linewidth=0.5)

plt.title('Normalized Audio Signal')

plt.ylabel('Amplitude')

plt.grid(axis='y', linestyle='--')

plt.ylim(-1.1, 1.1)

plt.xlim(0, duration)

plt.show()

**Question2**

The message can still be understood when resampled down to **about 2000 Hz**

It becomes difficult to understand at **1000Hz,** **500Hz and 250hz**

A screenshot of a screen

AI-generated content may be incorrect.

**Code**

import librosa

import soundfile as sf

import matplotlib.pyplot as plt

import librosa.display

y, sr = librosa.load('Kuusi.wav', sr=8000)

y\_2000 = librosa.resample(y, orig\_sr=sr, target\_sr=2000)

sf.write('Kuusi\_2000Hz.wav', y\_2000, 2000)

y\_1000 = librosa.resample(y, orig\_sr=sr, target\_sr=1000)

sf.write('Kuusi\_1000Hz.wav', y\_1000, 1000)

y\_500 = librosa.resample(y, orig\_sr=sr, target\_sr=500)

sf.write('Kuusi\_500Hz.wav', y\_500, 500)

y\_250 = librosa.resample(y, orig\_sr=sr, target\_sr=250)

sf.write('Kuusi\_250Hz.wav', y\_250, 250)

plt.figure(figsize=(10, 6))

plt.subplot(4,1,1)

librosa.display.waveshow(y\_2000, sr=2000)

plt.title('Resampled at 2000 Hz')

plt.subplot(4,1,2)

librosa.display.waveshow(y\_1000, sr=1000)

plt.title('Resampled at 1000 Hz')

plt.subplot(4,1,3)

librosa.display.waveshow(y\_500, sr=500)

plt.title('Resampled at 500 Hz')

plt.subplot(4,1,4)

librosa.display.waveshow(y\_250, sr=250)

plt.title('Resampled at 250 Hz')

plt.xlabel('Time (s)')

plt.tight\_layout()

plt.show()

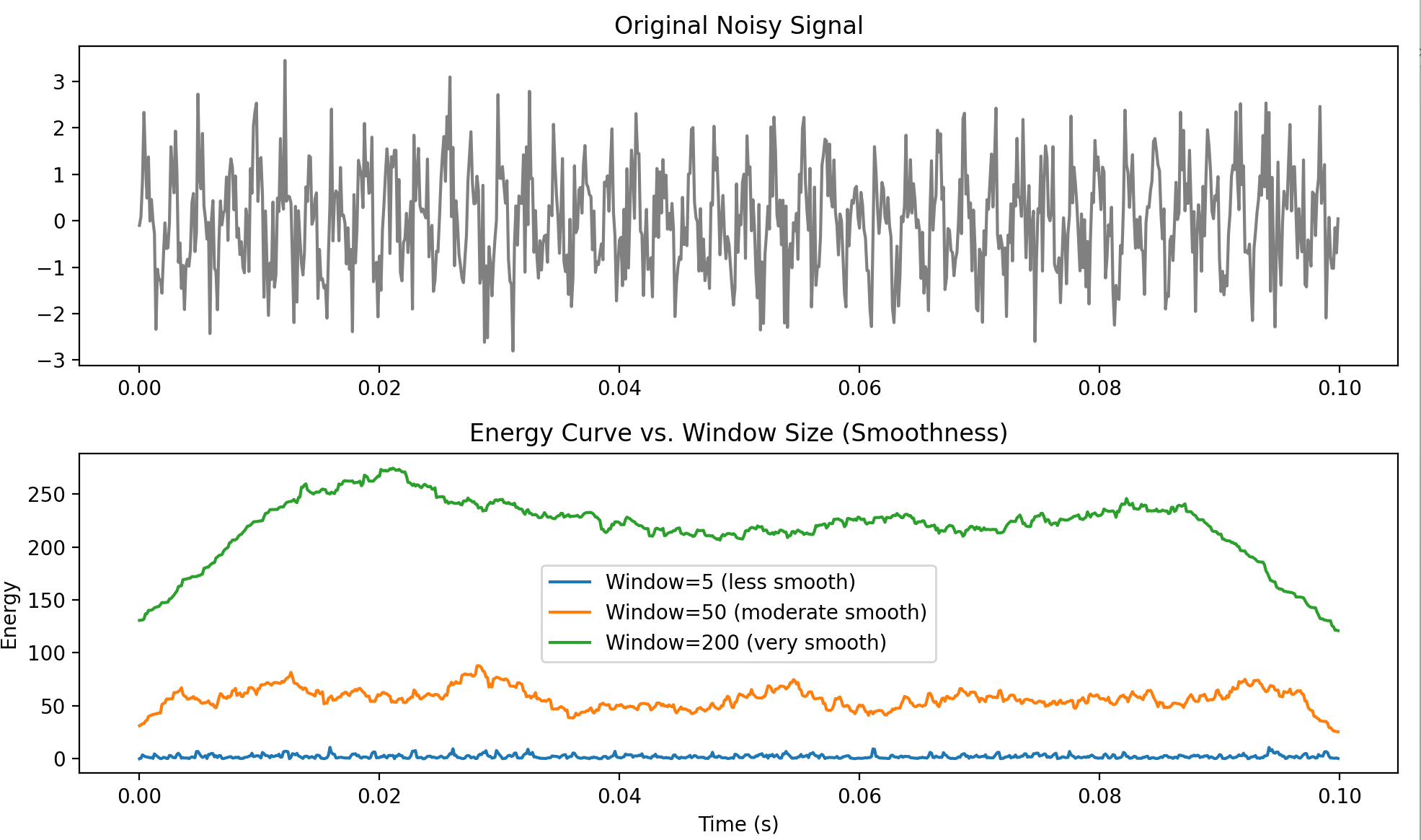
**Question 3**

**Part 1**

**A graph showing a number of different colored lines

AI-generated content may be incorrect.**

**Part 2**

****

As the window size increases, the energy curve becomes smoother and less affected by short-term noise fluctuations. This makes the signal energy representation more stable and interpretable for machine processing instead of reacting to random noise spikes. As shown in the above figure

**Code**

#Jari Turunen, TUNI

import numpy as np

from numpy import cos, sin, pi, absolute, arange, mean

from matplotlib import pyplot as plt

from scipy.stats import skew, kurtosis

fs = 8000

freq = 440

end\_time = 0.1

time = np.arange(0, end\_time, 1/fs)

print(len(time))

y = sin(2\*pi\*freq\*time) + np.random.normal(loc=0.0, scale=0.8, size=[1, len(time)])

y = y.squeeze()

print(y.shape)

len1 = 4

len2 = 10

x = y.copy()\*0

x2 = y.copy()\*0

x3 = y.copy()\*0

x4 = y.copy()\*0

for i in range(len(y)):

    print("%d / %d\n" % (i, len(y)))

    start = i - len1

    if start < 1:

        start = 1

    start2 = i - len2

    if start2 < 1:

        start2 = 1

    ending = i + len1

    if ending > len(y):

        ending = len(y)

    ending2 = i + len2

    if ending2 > len(y):

        ending2 = len(y)

    if len(y[start:ending]) < 2:

        x[i] = 0

    else:

        x[i] = np.mean(y[start:ending])

    if len(y[start2:ending2]) < 2:

        x2[i] = 0

    else:

        x2[i] = np.mean(y[start2:ending2])

        x3[i] = skew(y[start2:ending2], axis=0, bias=True)

        x4[i] = kurtosis(y[start2:ending2], axis=0, bias=True)

plt.figure(figsize=(10,5))

plt.plot(y)

plt.plot(x, 'r')

plt.plot(x2, 'g')

plt.plot(x3, 'b')

plt.plot(x4, 'y')

plt.legend(['Original', str(len1\*2+1)+'-sample filtered', str(len2\*2+1)+'-sample filtered'])

plt.title("Signal and Windowed Means/Statistics")

plt.show()

def energy\_curve(signal, window):

    en = np.zeros(len(signal))

    half\_win = window // 2

    for i in range(len(signal)):

        start = max(0, i - half\_win)

        end = min(len(signal), i + half\_win)

        segment = signal[start:end]

        en[i] = np.sum((segment - np.mean(segment))\*\*2)

    return en

E1 = energy\_curve(y, 5)

E2 = energy\_curve(y, 50)

E3 = energy\_curve(y, 200)

plt.figure(figsize=(10,6))

plt.subplot(2,1,1)

plt.plot(time, y, color='gray')

plt.title("Original Noisy Signal")

plt.subplot(2,1,2)

plt.plot(time, E1, label="Window=5 (less smooth)")

plt.plot(time, E2, label="Window=50 (moderate smooth)")

plt.plot(time, E3, label="Window=200 (very smooth)")

plt.xlabel("Time (s)")

plt.ylabel("Energy")

plt.title("Energy Curve vs. Window Size (Smoothness)")

plt.legend()

plt.tight\_layout()

plt.show()

**Part 3**

The function skew() measures the *asymmetry* of the data distribution.

The function kurtosis() measures how *peaked* or *flat* the distribution is compared to a normal distribution.

The energy function measures the total signal power within a window.

The python moment() function provides statistical features like variance, mean and it tells us how the values of a signal are distributed around the mean.

Part 4

A graph of stock market growth

AI-generated content may be incorrect.

**Code**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

data = pd.read\_pickle("AMZN.pkl")

print(data.head())

y = data['Open'].values

time = np.arange(len(y))

def energy\_curve(signal, window):

    en = np.zeros(len(signal))

    half\_win = window // 2

    for i in range(len(signal)):

        start = max(0, i - half\_win)

        end = min(len(signal), i + half\_win)

        segment = signal[start:end]

        en[i] = np.sum((segment - np.mean(segment))\*\*2)

    return en

E\_small = energy\_curve(y, 5)

E\_medium = energy\_curve(y, 30)

E\_large = energy\_curve(y, 100)

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

plt.subplot(2,1,1)

plt.plot(time, y, color='gray')

plt.title("Amazon Stock 'Open' Price Fluctuations")

plt.subplot(2,1,2)

plt.plot(time, E\_small, label='Window=5')

plt.plot(time, E\_medium, label='Window=30')

plt.plot(time, E\_large, label='Window=100')

plt.xlabel("Time (days)")

plt.ylabel("Energy")

plt.title("Energy Curve Smoothness for Different Window Sizes")

plt.legend()

plt.tight\_layout()

plt.show()

**Question 4**

Machine learning is a field in technology which uses scientific methods to automate machines and make them learn and perform actions with minimal error.

Neural networks have been developed over the years but the basic principle of them being able to minimize error in their predictions with respect to what the actual outcome would be.

Different architectures have been developed to solve different problems, for example image processing, signal processing, Natural language processing and others.

I think these architectures have only been decided through making research and experiments in order to come up with an architecture that fits a specific problem