FSM Mid-Internship Review

Predicting tool wear and surface roughness for a lathe machine

SAURABH MOURYA

MADHAV INSTITUTE OF TECHNOLOGY AND SCIENCE, GWALIOR

Under Mentorship of

Mr. Shridhar Katwe

IITD-AIA FOUNDATION FOR SMART MANUFACTURING

Objectives

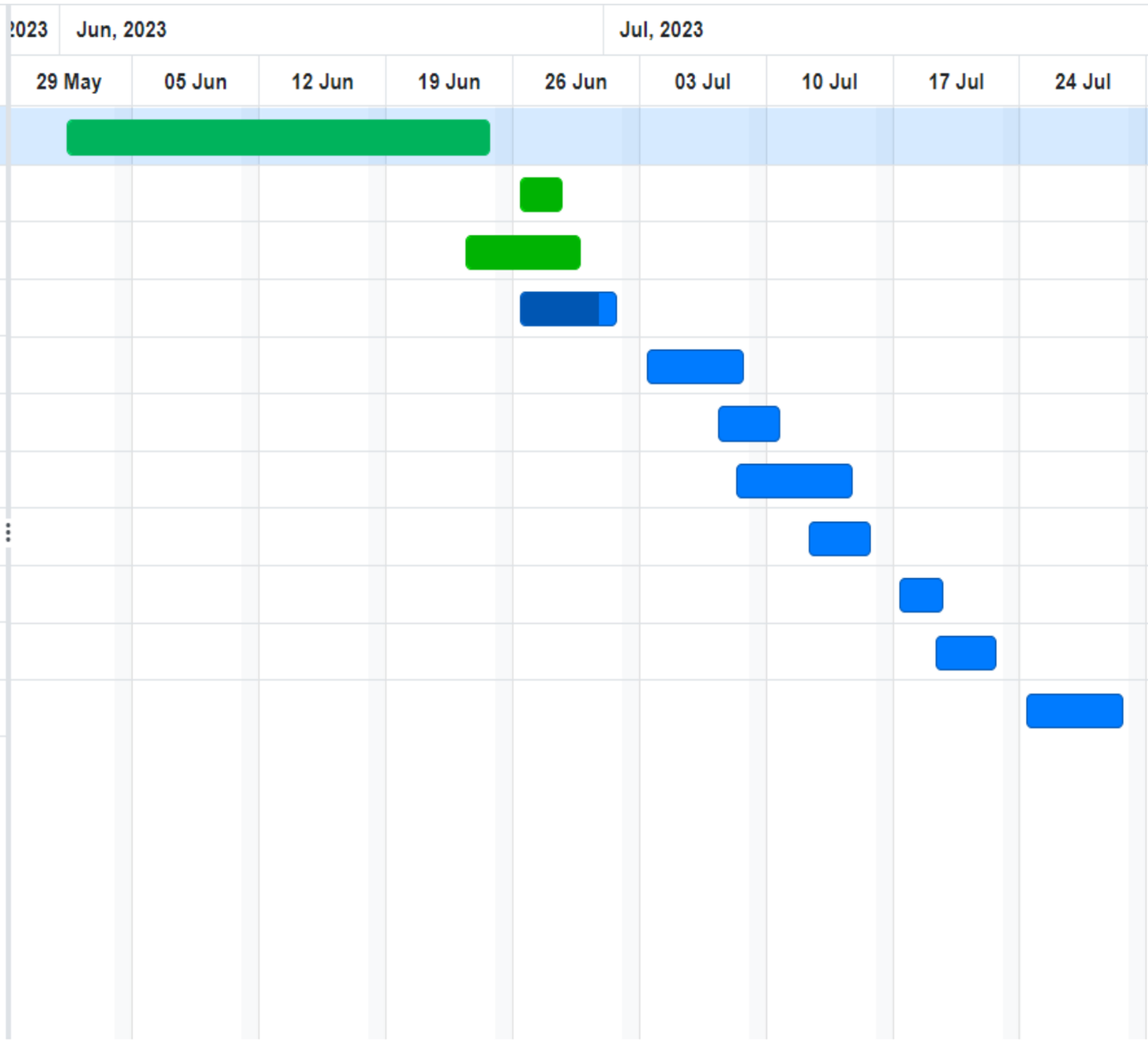
Phase 1:
Data pre-processing and analysis. **80%**

Phase 2:
Model training and testing. **0%**

Phase 3:
Model implementation in a web-app. **0%**

Timeline - Gantt chart

Name	Start Date	End Date	Duration
Learning: PandasNumPy, PyTorch, TensorFlow	Jun 01, 2023	Jun 24, 2023	21 days
Research and Familiarization:	Jun 26, 2023	Jun 28, 2023	3 days
Research on machine learning models for pred...	Jun 23, 2023	Jun 29, 2023	6 days
Data Preprocessing	Jun 26, 2023	Jul 01, 2023	6 days
Model Selection and Training	Jul 03, 2023	Jul 08, 2023	6 days
Model Validation and Interpretation	Jul 07, 2023	Jul 10, 2023	3 days
Implementation and Integration	Jul 08, 2023	Jul 14, 2023	6 days
Testing and Debugging	Jul 12, 2023	Jul 15, 2023	4 days
Documentation and Reporting	Jul 17, 2023	Jul 19, 2023	3 days
Further Testing, Analysis and Optimization	Jul 19, 2023	Jul 22, 2023	4 days
Final Documentation and Reporting	Jul 24, 2023	Jul 29, 2023	6 days



Screenshots of development

Importing data and converting it to the proper Time-Series format.

```
dataframes = []  # To store the imported data from each file

for i in range(1, 61):
    file = f"Data/{i}.xlsx"
    df = pd.read_excel(file)  # Use pd.read_excel() for Excel files
    df = df.dropna(axis='columns', how='all')
    df = df.dropna(axis='rows', how='all')
    df.columns = ['Time', 'X', 'Y', 'Z']
    df = df.iloc[1:]  # Exclude the original header row from the data
    df['Time'] = pd.to_datetime(df['Time'], unit='s').dt.time  # Convert 'Time' column to da
    dataframes.append(df)
    print(i) #too keep an eye on progress
exp = pd.read_excel("Data/Experiment Summary.xlsx")
```

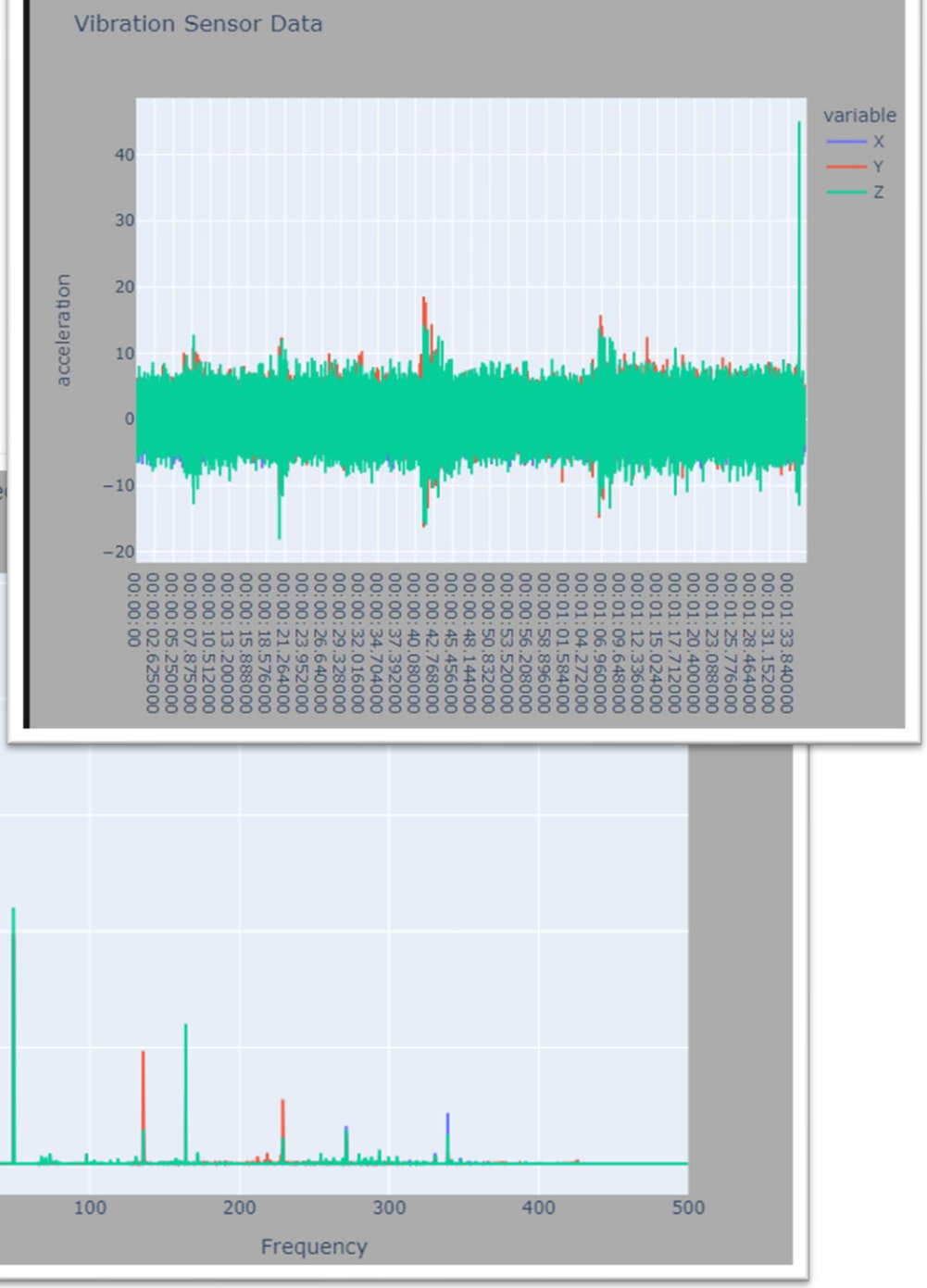
Converting the data from Time-domain to Frequency-Domain to identify the frequency components.

```
# Calculating PSD

def calculate_psd(dataframe, fs=1000):
    time = dataframe['Time']
    x = dataframe['X']
    y = dataframe['Y']
    z = dataframe['Z']

    # Convert time values to seconds
    time_seconds = [(t.hour * 3600 + t.minute * 60 + t.second + t.microsecond / 1e6) for t in time]

    # Apply Hanning window function
    window = np.hanning(len(time_seconds))
    x_windowed = x * window
    y_windowed = y * window
    z_windowed = z * window

    # Calculate PSD using periodogram
    f, psd_x = signal.periodogram(x_windowed, fs)
    _, psd_y = signal.periodogram(y_windowed, fs)
    _, psd_z = signal.periodogram(z_windowed, fs)

    return f, psd_x, psd_y, psd_z
```

```python
# Time_domain plot


def time_domain(num):
    df = dataframes[num]

    # Create a line plot using Plotly
    fig = px.line(df, x='Time', y=['X', 'Y', 'Z'], title='Vibration Sensor Data', labels={'value':'acceleration'})

    # Display the plot
    fig.update_layout(
        height=600,
        showlegend=True,
        paper_bgcolor= 'darkgrey')
    fig.show()
```

```python
#Plotting PSD


def freq_domain(frequencies, psd_x, psd_y, psd_z):
    # Convert complex PSD values to magnitude
    psd_x_mag = np.abs(psd_x)
    psd_y_mag = np.abs(psd_y)
    psd_z_mag = np.abs(psd_z)

    # Create line plots for X, Y, and Z axes
    fig = go.Figure()

    fig.add_trace(go.Scatter(x=frequencies, y=psd_x_mag, mode='lines', name='X'))
    fig.add_trace(go.Scatter(x=frequencies, y=psd_y_mag, mode='lines', name='Y'))
    fig.add_trace(go.Scatter(x=frequencies, y=psd_z_mag, mode='lines', name='Z'))

    fig.update_layout(
        title='Power Spectral Density',
        xaxis=dict(title='Frequency'),
        yaxis=dict(title='acceleration'),
        showlegend=True,
        paper_bgcolor= 'darkgrey',
        height=600


    )

    fig.show()
```

```
Experiment      1.000
Rpm           190.000
Feed            0.140
Depth           0.200
Ra              3.324
Name: 0, dtype: float64
```



Thank You

mouryas911@gmail.com