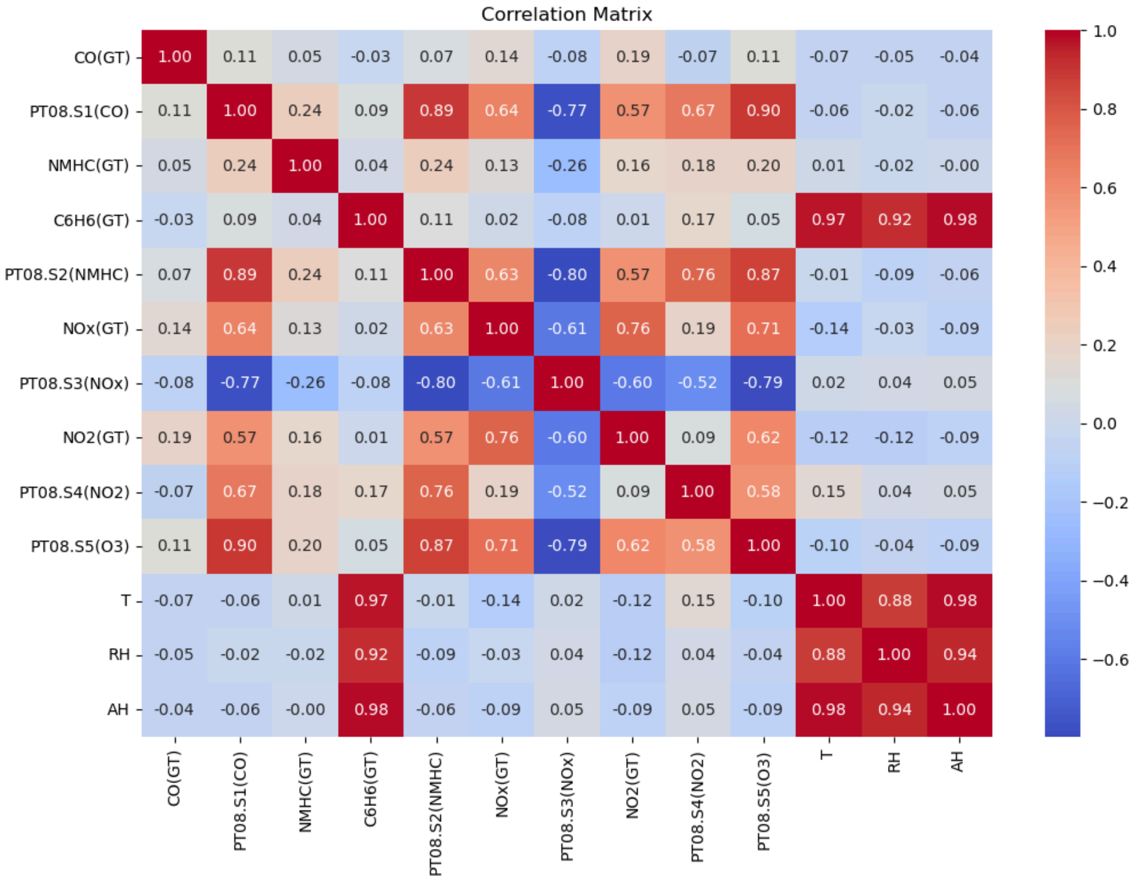
Nandhakumar Vadivel
50604851
Project 3

Part -1

1. Nature of the Dataset

The dataset is the **Air Quality Dataset**, which contains hourly averaged responses from an array of 5 metal oxide chemical sensors embedded in an Air Quality Chemical Multisensor Device. The data spans from March 2004 to February 2005 and contains 9358 instances. The dataset includes numerical values for various chemical components such as CO, NOx, Benzene, and others, measured by sensors, as well as ground truth values provided by a reference analyzer. This dataset is ideal for time series analysis as it captures sensor data continuously over time.

2. Correlation Plot and Feature Selection



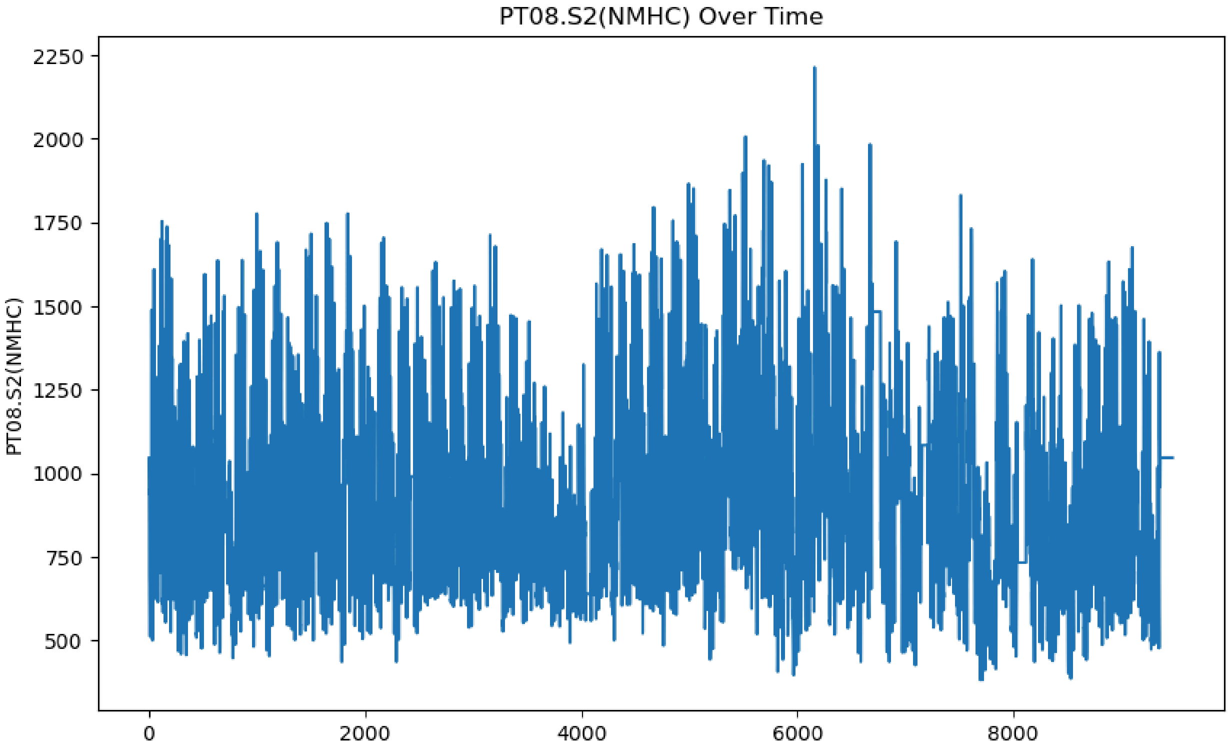
The correlation matrix (below) was computed to understand the relationships between the features and the target label PT08.S4(NO2).

Correlation Plot:

From the correlation analysis, the following features were found to have the **highest correlation** with the target variable PT08.S4(NO2):

- **PT08.S2(NMHC)**: Highly positively correlated.
- **PT08.S1(CO)**: Strong positive correlation.
- **PT08.S5(O3)**: Strong positive correlation.
- **NOx(GT)**: Positively correlated, indicating its influence on the target.

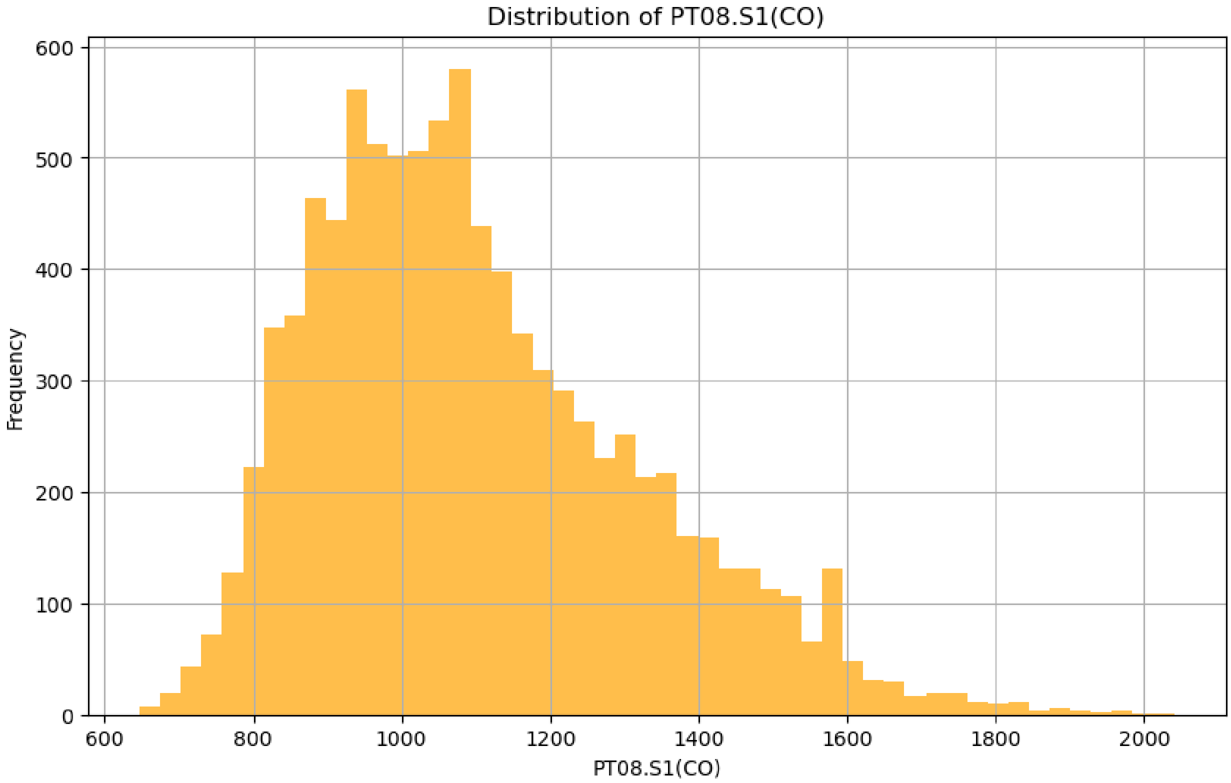
3.



Time Series of PT08.S2(NMHC)

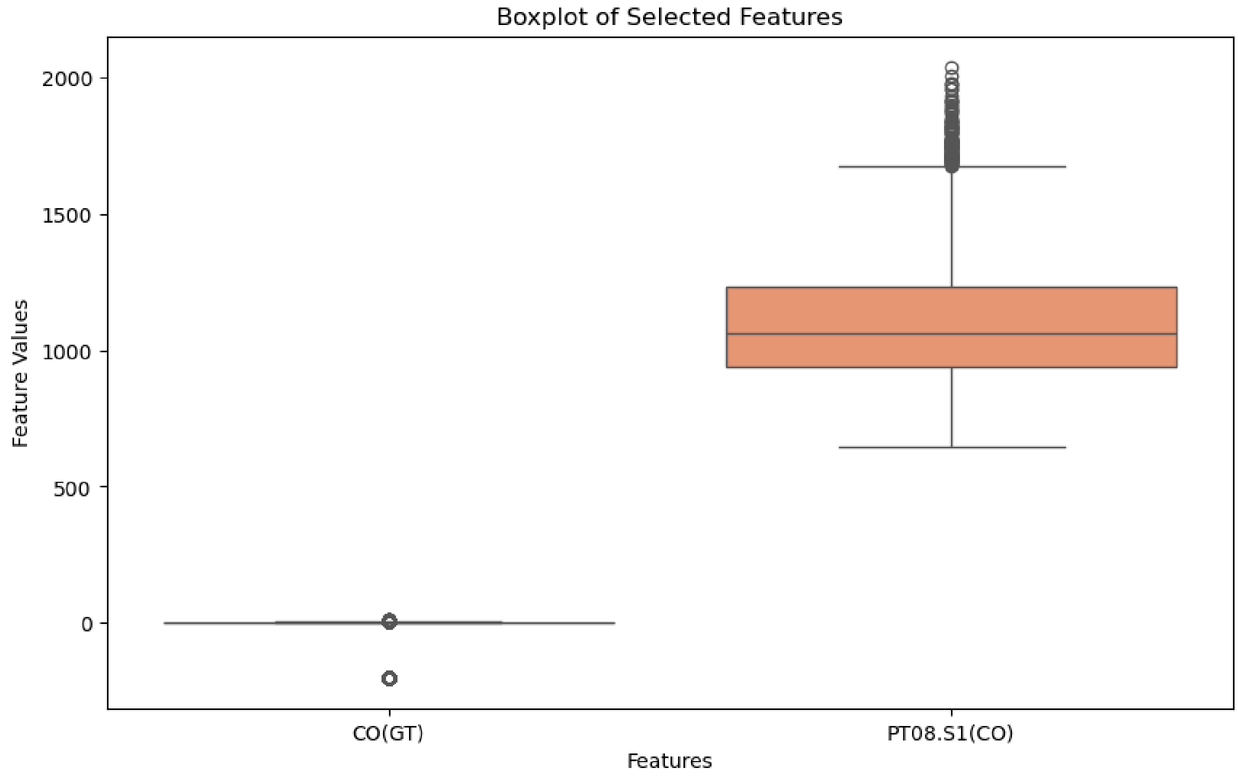
- The above graph illustrates how the sensor response for PT08.S2(NMHC) fluctuates over time.

- The data exhibits significant variation, with peaks and troughs observed across the year. This variability suggests that NMHC levels are highly dynamic, likely influenced by environmental or seasonal factors.
- This pattern is crucial for time series modeling as it hints at the presence of temporal trends or periodicity.



Distribution of PT08.S1(CO)

- This histogram shows the distribution of sensor readings for PT08.S1(CO), which measures Carbon Monoxide levels.
- Most of the values are concentrated between **800 and 1200**, with a long tail extending toward higher values. This indicates that while most observations are within a normal range, there are occasional spikes in CO levels.
- Understanding the distribution helps identify outliers and informs us about the typical behavior of this feature.



CO(GT):

- The distribution of CO(GT) is extremely narrow, with most values clustered around 0.
- A few outliers are visible below the whiskers, indicating significantly lower CO levels for some samples.
- The overall range and variance for this feature are minimal, suggesting it might not vary much across the dataset.

PT08.S1(CO):

- The distribution for PT08.S1(CO) is much wider, with values spread across a higher range (approximately 500 to 2000).
- The box shows the interquartile range (IQR), and the median is closer to the middle of the box, indicating a fairly symmetric distribution.
- Numerous outliers are present above the whisker, suggesting that some samples have abnormally high PT08.S1(CO) values.

4. Purpose of Breaking Data into Sequences

Breaking data into sequences is crucial in time series analysis because:

1. **Temporal Dependencies**: It preserves the temporal structure of the data, allowing models to learn patterns over time.
2. **Real-World Simulation**: Models predict future outcomes based on past observations. For this analysis, a sequence length of 10 means the model uses the last 10 observations to make predictions.
3. **Improved Learning**: Sequential data helps recurrent models like RNNs or LSTMs capture temporal dynamics effectively.

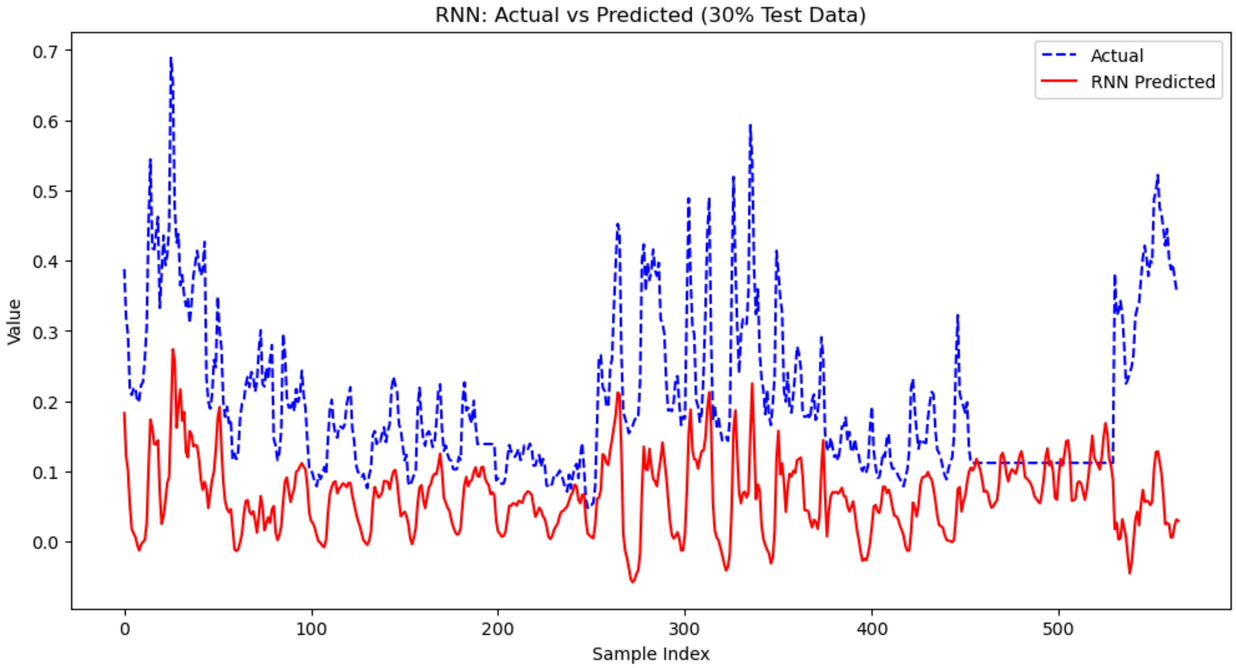
Part -2

1.In two lines write your understanding of RNN and LSTM models.

Recurrent Neural Networks (RNNs): RNNs are designed to process sequential data by maintaining a hidden state that captures temporal dependencies in the sequence. However, they often suffer from vanishing gradient issues when dealing with long sequences.

Long Short-Term Memory Networks (LSTMs): LSTMs are an improvement over RNNs, incorporating gates (input, forget, and output) to better handle long-term dependencies in sequences, making them more robust for time series forecasting.

2. Include both the graphs in your report and write your analysis.



1. Actual Data (Blue Dashed Line):

- **Observation:**
  - The actual data ranges between **0.1 and 0.7**, with visible sharp peaks and troughs.
  - Peaks are observed around indices **0 to 50**, reaching as high as **0.7**.
  - The data gradually decreases after index **100**, reaching a low around **0.1** at index **200**.
  - Towards the end (index ~500), the data shows another increase, reaching around **0.6**.

---

2. RNN Predictions (Red Line):

- **Observation:**
  - The RNN predictions range between **0.05 and 0.3**, consistently underestimating the actual data values.
  - The predictions fail to capture the magnitude of the sharp peaks and troughs.
  - At peaks in index **50** , where the actual value reaches **0.7**, the RNN predicts only around **0.25**.

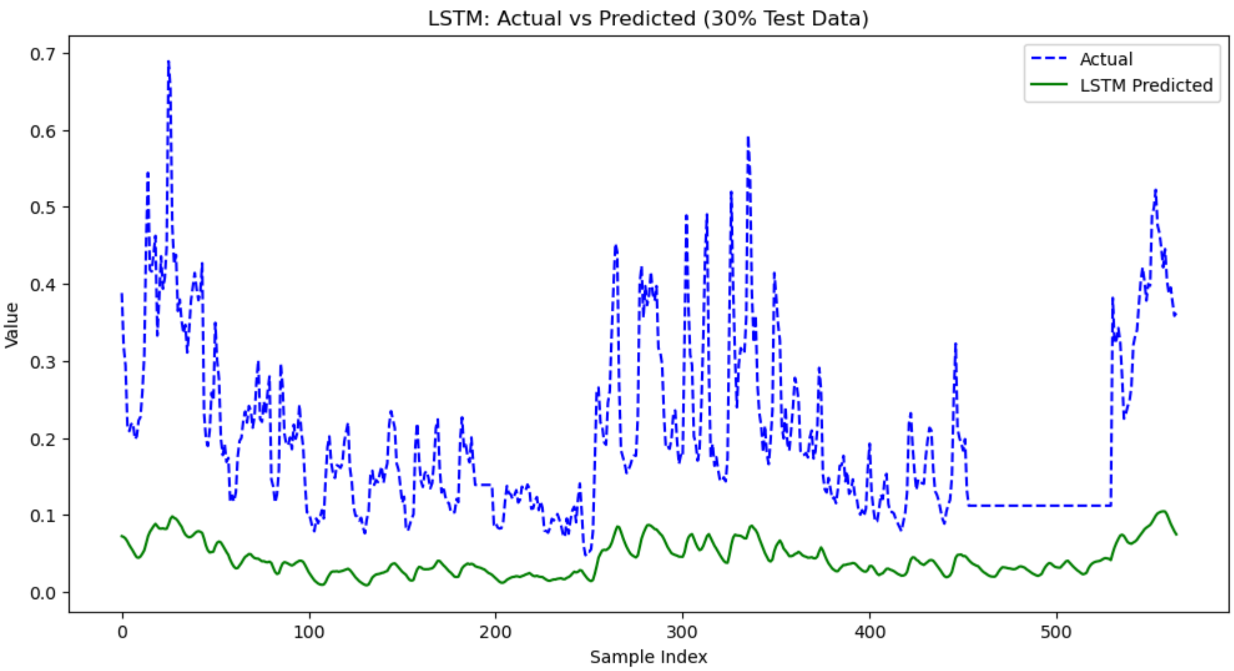
    - At troughs (e.g., index **200**), where the actual value drops to **0.1**, the RNN predicts approximately **0.15**, overestimating the trough.

Key Mismatches:

1. **Index ~50 (Peak)**:
    - **Actual Value**: ~**0.7**.
    - **RNN Prediction**: ~**0.25**.
    - **Interpretation**: RNN underestimates the peak significantly.
2. **Index ~200 (Trough)**:
    - **Actual Value**: ~**0.1**.
    - **RNN Prediction**: ~**0.15**.
    - **Interpretation**: RNN slightly overestimates the trough, showing less sensitivity to sharp drops.
3. **Index ~500 (Rising Trend)**:
    - **Actual Value**: ~**0.6**.
    - **RNN Prediction**: ~**0.3**.
    - **Interpretation**: RNN again underestimates the rising trend.

Overall Observations:

- The RNN model struggles to capture the **magnitude** of variations in the actual data, smoothing out sharp peaks and troughs.
- While it somewhat tracks the overall trend, the scale of predictions is consistently **lower than the actual values**.



Actual Data (Blue Dashed Line):

- **Range**:
  - The actual data fluctuates significantly between **0.1** and **0.7** across the test samples.
- **Peaks and Troughs**:
  - **Index ~50**: Peak at approximately **0.7**.
  - **Index ~200**: Trough at approximately **0.1**.
  - **Index ~500**: Rising again to around **0.6**.

---

2. LSTM Predictions (Green Line):

- **Range**:
  - The LSTM predictions are smoother, ranging between **0.05 and 0.15**, failing to capture the actual data's dynamic variations.
- **Key Points of Mismatch**:
  - **Index ~50 (Peak)**:
    - **Actual**: ~**0.7**.
    - **LSTM Prediction**: ~**0.1**.
    - **Interpretation**: LSTM significantly underestimates the peak.

        - **Index ~200 (Trough)**:
            - **Actual**: **~0.1**.
            - **LSTM Prediction**: **~0.05**.
            - **Interpretation**: LSTM slightly underestimates but fails to match the sharp drop.
        - **Index ~500 (Rising Trend)**:
            - **Actual**: **~0.6**.
            - **LSTM Prediction**: **~0.12**.
            - **Interpretation**: LSTM underestimates the rising trend.

---

3. Observations:

1. **Smoothness of Predictions**:
    - The LSTM predictions exhibit a narrow range, staying within **0.05 to 0.15**, irrespective of the actual data's sharp peaks and troughs.
    - The model smooths out variations and misses the dynamic behavior of the actual values.
2. **Underfitting**:
    - The consistent low predictions indicate that the LSTM model might be **underfitting** the data, likely due to insufficient learning capacity or overly regularized settings.
3. **Mismatch with Dynamic Behavior**:
    - The actual data exhibits sharp and frequent changes (e.g., between indices **50 to 100**), which the LSTM fails to replicate, producing almost flat predictions.

Conclusion:

- The **LSTM model** does not perform well on this dataset, as it smooths out the predictions and fails to align with the actual dynamic variations.
- The underfitting suggests that the model might need:
    - **Increased complexity** (e.g., more layers or units).
    - **Hyperparameter tuning** to better capture short-term variations

3. Which model do you think performed well and why?

RNN:

- **Strengths**:
  - The RNN captures the overall trends in the data better than LSTM, especially short-term fluctuations.
  - Its predictions follow the actual values' dynamic behavior more closely, even if they underestimate the magnitude of peaks and troughs.
- **Weaknesses**:
  - RNN tends to overfit and exaggerates variance, leading to significant mismatches in some areas.

LSTM:

- **Strengths**:
  - LSTM predictions are smooth and stable, avoiding the noise seen in RNN predictions.
- **Weaknesses**:
  - LSTM severely underfits the data, failing to capture the dynamic nature and magnitude of actual values.
  - It produces overly smooth predictions that ignore the sharp changes in peaks and troughs.

The RNN performed better overall.

- While it is noisy and imperfect, the RNN captures the short-term trends better and aligns closer to the actual data's dynamics.
- LSTM, on the other hand, fails to capture both the trends and magnitude, likely due to underfitting or improper tuning of hyperparameters.

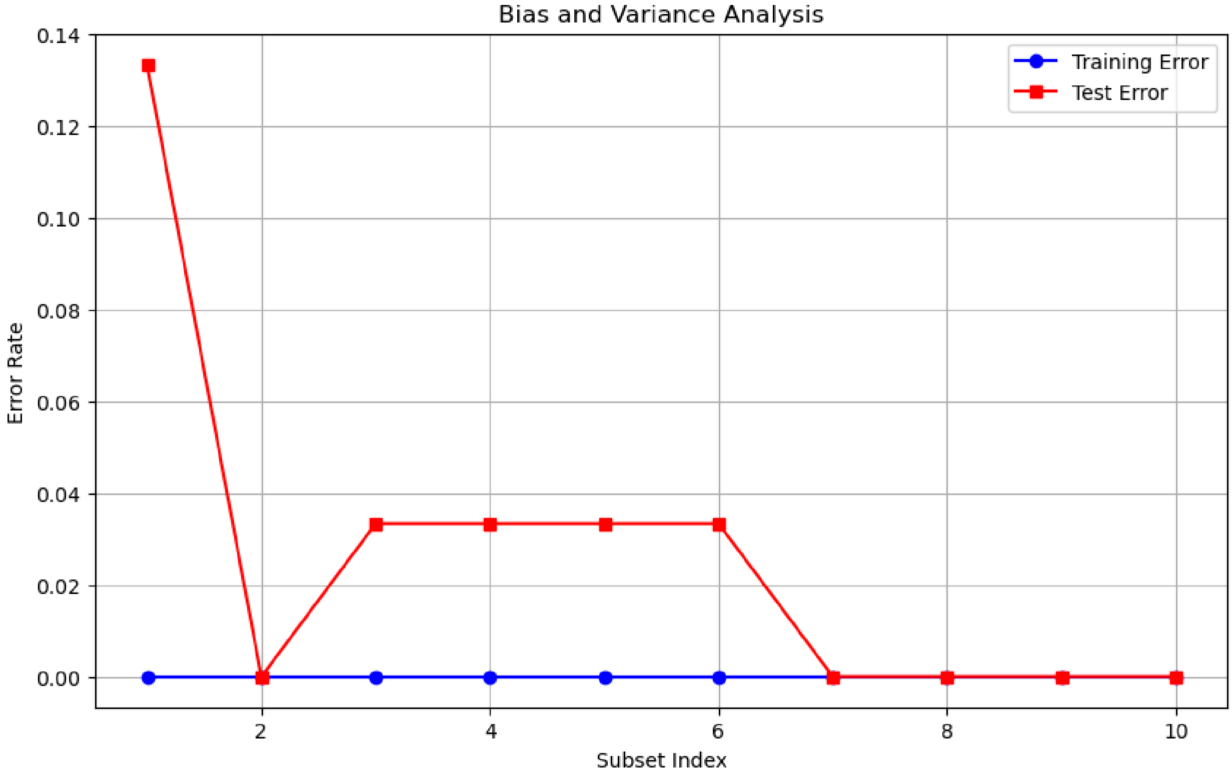
Part-3

1. What is Bias?

- Bias refers to the error introduced by approximating a real-world problem (which may be complex) with a simpler model.
- High bias typically results from **underfitting**, where the model fails to capture the underlying patterns of the data.
- For example, a linear model trying to fit non-linear data would have high bias.

2. What is Variance?

- Variance refers to the error introduced due to the model's sensitivity to small fluctuations in the training data.
- High variance usually results from **overfitting**, where the model captures noise in the training data rather than the actual signal.



Observations from the Graph:

1. **Training Error (Blue Line)**:

  - The training error remains consistently low (close to **0**), indicating the model fits the training data well. This suggests **low bias**.

2. **Test Error (Red Line)**:
   - Initially, the test error starts high (around **0.14** at index 1), then drops significantly by index **2**.
   - Between indices **3 to 7**, the test error stabilizes around **0.04**, indicating the model generalizes reasonably well during this range.
   - At index **8 to 10**, the test error further decreases to near **0**, suggesting a potentially overfitted model.