Assignment 2 – COMP809

Flood Warning Prediction: Evaluating MLP & LSTM Methods

Thandar Kyi

ID: 24251155

Table of Contents

List of Tables & Figures

Part B: Flood Warning Prediction

1. Introduction and Data Preprocessing

1.1 Introduction

Flooding poses significant threats to urban infrastructure and public safety, necessitating the development of predictive models that can provide early warnings. In this assignment, we aim to build a machine learning-based flood prediction system using environmental data collected from the **Puhinui @ Drop Structure** river monitoring site. The dataset was sourced from the **Environmental Auckland Data Portal** and contains hourly observations from **May 2020 to April 2025**. The dataset includes measurements of river water level and river discharge, rainfall at two locations (Manukau and Puhinui), relative humidity, air temperature, wind speed, and wind direction. The primary target variable is **river water level** at the Puhinui @ Drop Structure, with additional predictors such as **river discharge, rainfall**, **humidity**, **air temperature**, **wind speed**, and **wind direction**.

1.2 Data Preprocessing

1.2.1 Renaming Columns and Parsing Timestamps

The raw dataset contains four rows of metadata, including variable descriptions and station identifiers. After skipping the first four metadata rows, we renamed the columns to intuitive names such as "start_time", "end_time", "river_level_m", "river_discharge_m3s", "rainfall_manukau_mm", "rainfall_puhinui_mm", "humidity_pct", "air_temp_C", "wind_speed_ms", "wind_dir_deg" to enhance readability. The start_time and end_time columns were parsed into datetime objects to ensure consistent temporal interpretation and facilitate time-based operations.

1.2.2 Ensuring Hourly Temporal Resolution

The dataset consists of **43,800 hourly records**, matching the expected count for 5 years of hourly data (365 days × 24 hours × 5 years). There are **no missing or duplicate timestamps**, confirming that the temporal resolution is consistent and uninterrupted.

1.2.3 Converting Measurement Columns to Numeric

All sensor readings were explicitly converted to numeric to handle any non-numeric or corrupted entries. Columns affected include river level, discharge, rainfall, humidity, temperature, wind speed, and wind direction.

1.2.4 Missing Data Identification

Missing values were identified in four key variables:

- river_level_m: 62 missing values
- river_discharge_m3s: 61 missing values
- humidity_pct: 2,421 missing values

- wind_dir_deg: 2,421 missing values

To handle missing values, a **time-based interpolation** method was applied using the start_time column as the temporal index. This method leverages the continuity and sequential nature of environmental sensor data, filling gaps by estimating missing values from surrounding observations in chronological order. Both forward and backward directions were used to ensure all missing values were addressed.

1.2.5 Outlier Detection and Correction

To ensure data reliability and model robustness, outlier detection was performed using **hard threshold rules** based on known physical and environmental constraints. Each numerical attribute was assessed using conditions derived from domain knowledge and standard sensor expectations. The criteria and results were as follows:

Attribute	Outlier Condition	Outliers Detected
air_temp_C	> 40 °C	0
humidity_pct	> 100% or < 0%	0
river_level_m	< 0 m	0
river_discharge_m3s	< 0 m³/s	0
rainfall_manukau_mm	< 0 mm	0
rainfall_puhinui_mm	< 0 mm	0
wind_speed_ms	< 0 m/s	0
wind_dir_deg	Outside valid range [0°, 360°]	0

Table 1.1: Outlier Criteria & Analysis

No outliers were detected under these rules, indicating that the dataset is already clean and free from physically implausible values. As a result, **no corrective actions for outliers were required**. This clean data foundation supports accurate feature engineering and model training in the subsequent stages of the flood prediction pipeline.

1.2.6 Attribute-Specific Missing Data and Outlier Analysis & Their Impact of Dataset Quality

Below is a summary of missing values and detected outliers for each key attribute:

Attribute	Missing Values	Outliers (Definition)	Correction Applied
river_level_m	62	Values < 0 (none found)	Interpolation in time (Forward/ Backward)
river_discharge_m3s	61	Values < 0 (none found)	Interpolation in time (Forward/ Backward)
rainfall_manukau_mm	0	Negative values (none found)	Not needed
rainfall_puhinui_mm	0	Negative values (none found)	Not needed
humidity_pct	2,421	Values > 100% (none found); Values < 0 (none found)	Interpolation in time (Forward/ Backward)
air_temp_C	0	Values > 40°C (none found)	Not needed
wind_speed_ms	0	Negative values (none found)	Not needed
wind_dir_deg	2,421	Values < 0 or > 360° (none found)	Interpolation in time (Forward/ Backward)

Table 1.2: Attribute-Specific Missing Data and Outlier Analysis

Below are their impacts on data quality:

- **Missing Data**: Large gaps (e.g., >2,000 missing values in humidity and wind direction) can skew model training, especially if these variables are predictive. Imputation using medians preserves structure but may reduce variability and hide temporal anomalies.
- **Outliers**: Outliers were defined but there was no outliers found and no action is required.

Addressing these issues ensures the dataset is accurate, internally consistent, and suitable for training robust flood prediction models without introducing artificial trends or losing temporal continuity.

1.2.7 Justification for Data Cleaning Approach

Based on the analysis of missing values and outliers, we adopted a **non-deletion strategy** to preserve the full temporal structure of the dataset, which is critical for time series modeling. Since the dataset consists of hourly measurements over a 5-year period, removing rows would introduce **temporal discontinuities** that can mislead sequence-based models such as LSTMs.

- For **missing values**, we applied **time-based interpolation** method with forward and backward filling of any gaps. This approach is robust and suitable for time-series data, especially environmental sensors, where smooth transitions are expected.
- For **outliers**, domain-informed thresholds were applied:

- Air temperatures above 40°C and humidity above 100% were considered physically implausible for Auckland but no outlier is found.
- Relative humidity values below 0% were considered meteorologically invalid and would also have been corrected or interpolated had any been found.
- River level and discharge values below 0 were treated as hydrologically impossible, as negative values cannot occur in physical measurement. These would have been corrected through interpolation if present.
- Negative rainfall measurements at either station were flagged as sensor or logging errors since rainfall cannot be negative. Such values would have been replaced with interpolation as needed.
- Wind speeds below 0 m/s were similarly invalid and were to be corrected by interpolation if any had been detected.
- Wind direction values must be within compass [0°, 360°] range.

As no outliers were detected across any of the attributes under these strict physical constraints, no correction was ultimately required.

This approach preserves data integrity, avoids arbitrary loss of records, and ensures that subsequent feature engineering and model training are based on realistic, consistent inputs.

1.2.8 Flood Classification

Flood events were defined using a **threshold based on the 99th percentile** of river level observations:

- 99th percentile of river_level_m: **0.699 m**
- Flood classification: flood = 1 if river level ≥ 0.699 m, else flood = 0

Out of 43,800 observations, 43362 were classified as non-flood events and **438 were classified as flood events**, which aligns with the expected rarity of floods.

2. Data Exploration and Feature Selection

2.1. Feature Selection Using Pearson Correlation

To identify the most predictive attributes of river water level, we calculated the Pearson correlation coefficients between river_level_m and all other continuous features in the dataset. The flood column is excluded from the correlation analysis because it is a binary variable derived directly from river_level_m using a 99th percentile threshold. Including it would result in an artificially inflated correlation, offering no independent predictive value and potentially biasing feature selection. The top four predictors, ranked by absolute correlation strength, are:

Rank	Predictor	Correlation with river_level_m
1	river_discharge_m3s	0.57
2	rainfall_manukau_mm	0.29
3	rainfall_puhinui_mm	0.29
4	Humidity_pct	0.25

Table 2.1: Top Four Predictive Attributes of River Water Level

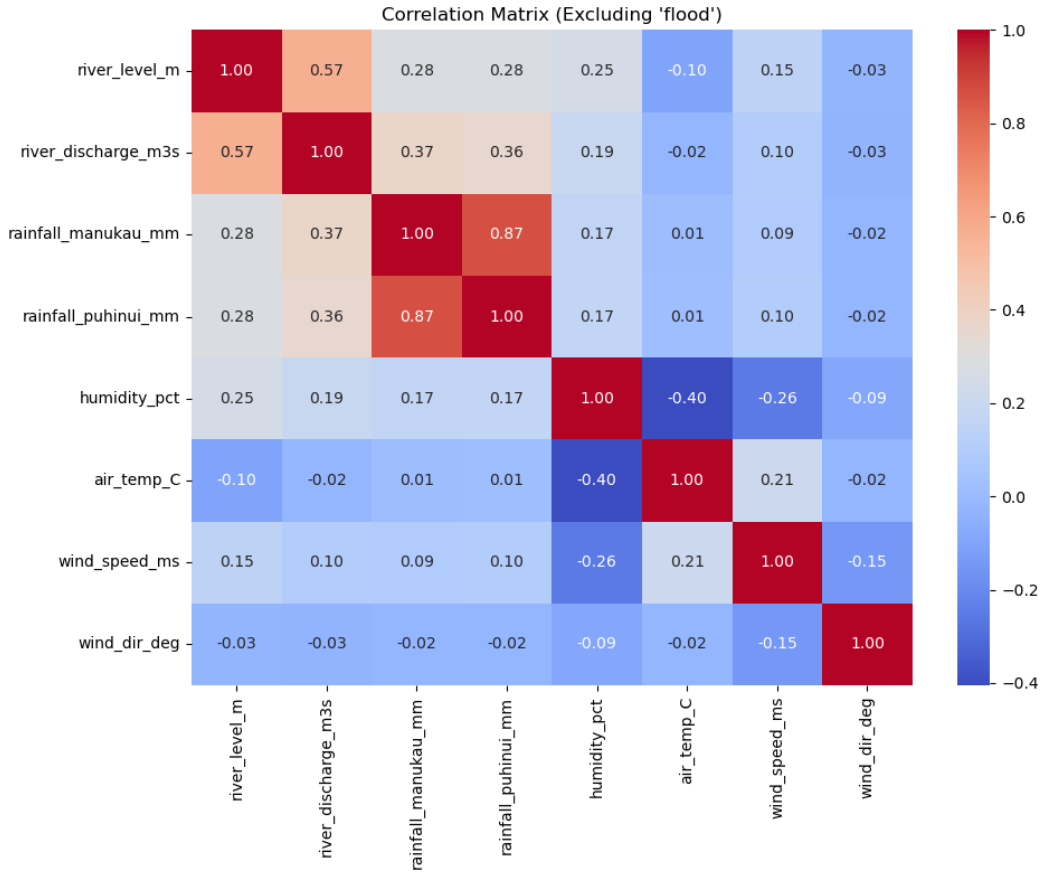

Figure 2.1: Correlation Heatmap of River Water Level

These values are confirmed by the **correlation heatmap** (see Figure 2.1), which visually emphasizes the strong linear relationship between river discharge and river level, along with the moderate positive associations of rainfall variables.

2.2. Predictor Influence on Flooding

- **River Discharge (river_discharge_m3s)**: Shows the strongest correlation with river level, which is expected since discharge directly affects river height. As discharge increases, river levels rise, often signaling potential flooding.
- **Rainfall at Manukau and Puhinui**: Both rainfall measures are positively correlated (0.29) with river level. High cumulative rainfall increases runoff into the river, elevating the water level. Interestingly, both rainfall locations showed very similar correlation strength (0.87), reflecting consistent regional weather patterns.
- **Humidity (humidity_pct)**: Although moderately correlated with river level (correlation = 0.25), high humidity typically accompanies heavy rainfall, which can contribute indirectly to rising river levels. While not a direct driver, elevated humidity can serve as a supporting indicator of storm systems and moisture-rich environments conducive to flooding.

2.3. Graphical Visualization of River Water Level

As seen in figure 2.1 (line plot), river water levels generally fluctuate within a narrow range (under 1.0 m), with **occasional sharp spikes**, particularly around early 2023. These extreme values correspond to flood events and are crucial in training the model to detect anomalies or dangerous surges.

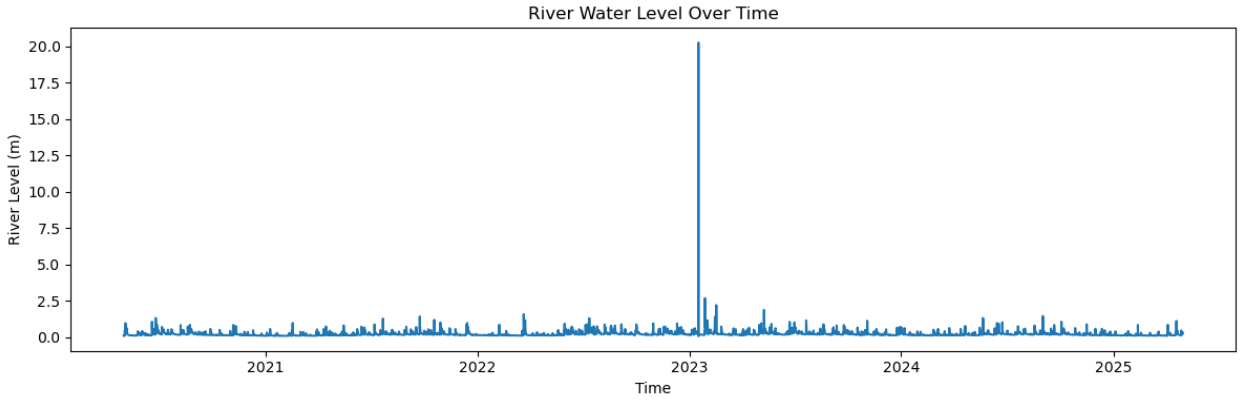

Figure 2.2: River Water Level Over time

2.4. Flood Class Distribution

Using a 99th percentile threshold of river level to define flooding, the class distribution is:

Class	Count	Percentage
0 (No Flood)	43,362	99.0%
1 (Flood)	438	1.0%

Table 2.2: Flood Class Distribution

This distribution indicates an imbalanced classification issue, typical of real-world flood datasets, and highlights the importance of techniques that handle rare-event modeling effectively.

2.5. Summary Statistics for Selected Predictors

These following table 2.3 statistics show that most rainfall and discharge readings are near zero (typical for dry periods), but the maximum values are very high corresponding to rare but impactful flood events.

Statistics	river_discharge_m3s	rainfall_puhinui_mm	rainfall_manukau_mm	humidity_pct
Count	43800.000	43800.000	43800.000	43800.000
Mean	0.242644	0.153939	0.145587	75.083
Std Dev	0.805458	0.867889	0.872663	13.48558
Min	0.004833	0.000000	0.000000	28.000
25% (Q1)	0.051029	0.000000	0.000000	65.02341
Median (Q2)	0.097575	0.000000	0.000000	76.400
75% (Q3)	0.191767	0.000000	0.000000	85.950
Max	63.686517	43.120000	50.311182	100.000

Table 2.3: Summary Statistics of Top Four Predictors

3. Experimental Methods

3.1. Data Preparation and Split Strategy

To evaluate the performance of both MLP and LSTM models, the cleaned dataset was split chronologically into three segments:

- 70% Training Set: Used to train the model.
- 10% Validation Set: Used for tuning hyperparameters and monitoring overfitting during training.
- 20% Test Set: Held out for final performance evaluation.

Since this is a time series dataset, the data was not shuffled — to preserve temporal dependencies.

3.2. Lookback and Forecast Horizon

Both the MLP and LSTM models were configured using a **sliding window approach**:

- **Input window (lookback):** Past 24 hours of data (features from previous 24-time steps).
- **Output window (horizon):** Predict River water level for the **next 3 hours**.

The input feature matrix was constructed using rolling windows such that at each time step t, the model receives input from t-24 to t-1 and predicts river level for t to t+2.

3.3. Prediction and Classification

- The primary task is to predict future river water levels (regression output).
- These predicted values are then post-processed using the previously defined flood threshold (99th percentile) to generate binary labels:
    - If predicted value ≥ threshold → flood (1)
    - Else → non-flood (0)

This allows a dual evaluation of both continuous prediction accuracy (e.g., RMSE) and classification performance (e.g., recall, F1-score).

3.4. Workflow Diagram

Below is the conceptual workflow for both MLP and LSTM models for predicting river water level:

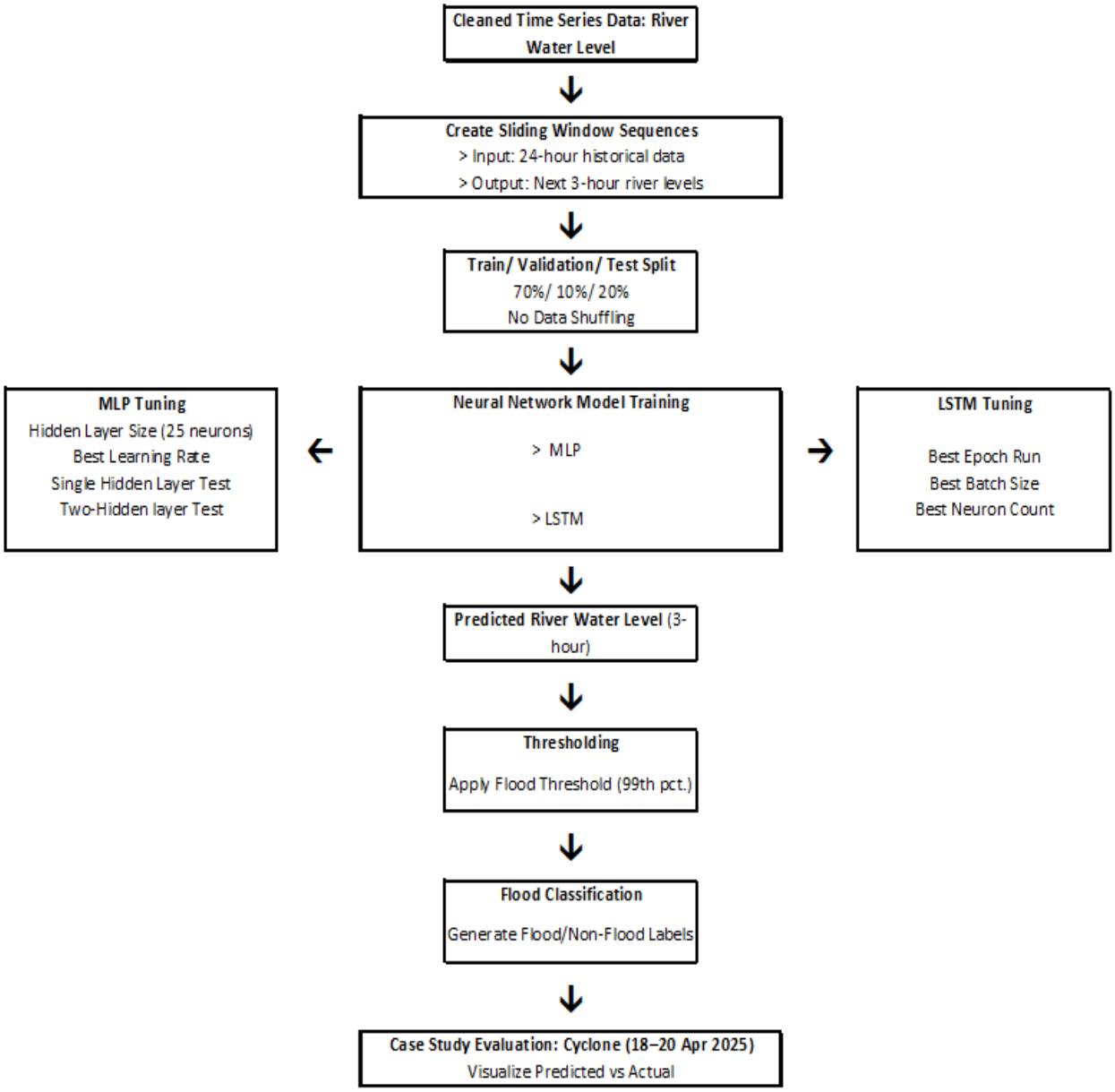

Figure 3.1: Workflow Diagram of Artificial Neural Network Modeling for River Water Level Prediction

4. Multilayer Perceptron (MLP)

4.1 Single Hidden Layer

To forecast river water levels and classify future occurrences as flood or non-flood, a baseline Multilayer Perceptron (MLP) model was developed using the sklearn.neural_network.MLPRegressor module. This model consists of a single hidden layer containing 25 neurons, and it was trained on a sliding window dataset with a 24-hour lookback period and a 3-hour prediction horizon. The learning rates evaluated were 0.0001, 0.001, 0.01, and 0.05, with optimal performance achieved at a learning rate of 0.01, resulting in an R^2 score of approximately 0.59 and a mean squared error (MSE) of around 0.0168 on the test data. Predicted river levels were categorised into flood and non-flood classifications using a threshold of 2.5 metres. Although the classification accuracy reached an impressive 99.98%, the confusion matrix revealed a significant imbalance within the dataset, resulting in a low recall for the flood class. This baseline model serves as a reference point for evaluating more complex architectures.

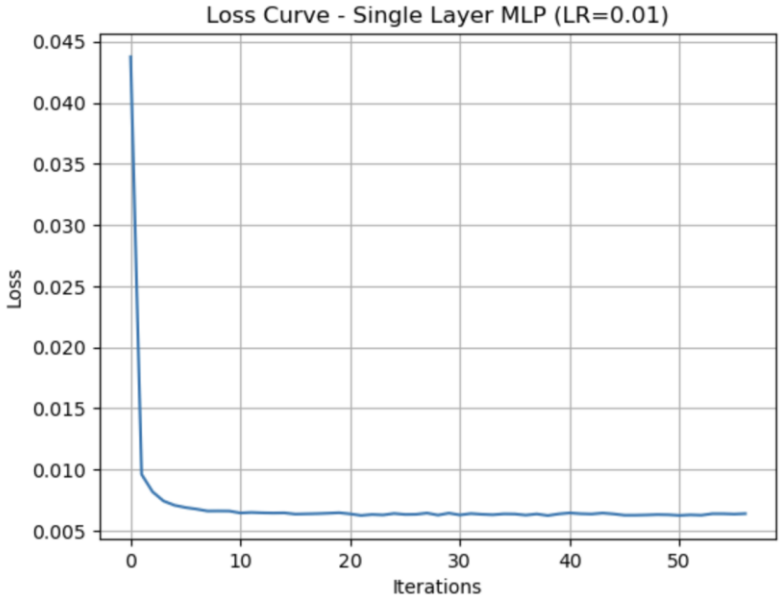

Figure 4.1: Training Loss Curve for Single-Layer MLPRegressor (Learning Rate = 0.01)

This curve validated the selection of a learning rate of 0.01 as the most stable and effective option among those tested.

Metric	Value
Best Learning Rate	0.01
R^2 Score	0.5896
MSE	0.0168
Classification Accuracy	0.9998

Metric	Value
Flood Class Recall	0.17
Flood Class Precision	0.50

Table 4.1: Performance Summary

The table above shows that the regression predictions lack precision, as suggested by the low R^2 value. On the other hand, due to the skewed class distribution, classification accuracy is high.

Sample	Hour 1	Hour 2	Hour 3
1	0.1224	0.1230	0.1216
2	0.1818	0.1825	0.1823
3	0.2091	0.2167	0.2126
4	0.1278	0.1268	0.1258
5	0.1368	0.1364	0.1344

Table 4.2: Sample 3-Hour Predictions (First 5 Samples)

Every row represents a single test sample, and each value represents the future hour's predicted water level. To determine flood classification based on the threshold, these predicted values were then averaged per sample.

4.2 Two Hidden Layers

This section aims to explore how distributing 25 neurons across two hidden layers affects the model's performance. We iteratively shifted neurons from the first hidden layer to the second, beginning with (24, 1) and ending with (1, 24), where each configuration is being trained on a separate model.

The learning rate was established at the optimal value identified in part 1. For each data split, the model forecasted water levels over the subsequent three hours by averaging the predictions and classifying them as either flood or non-flood. These classifications were subsequently compared against the predetermined flood threshold.

Results Summary

Each configuration was evaluated using R^2, mean squared error (MSE), and classification accuracy metrics. The optimal configuration was found to have 12 neurons in the first layer and 13 in the second, yielding an R^2 score of approximately 0.596, with a slight reduction in the mean squared error (MSE) compared to the single-layer model. Although classification accuracy remained high, the model continued to exhibit issues with class imbalance.

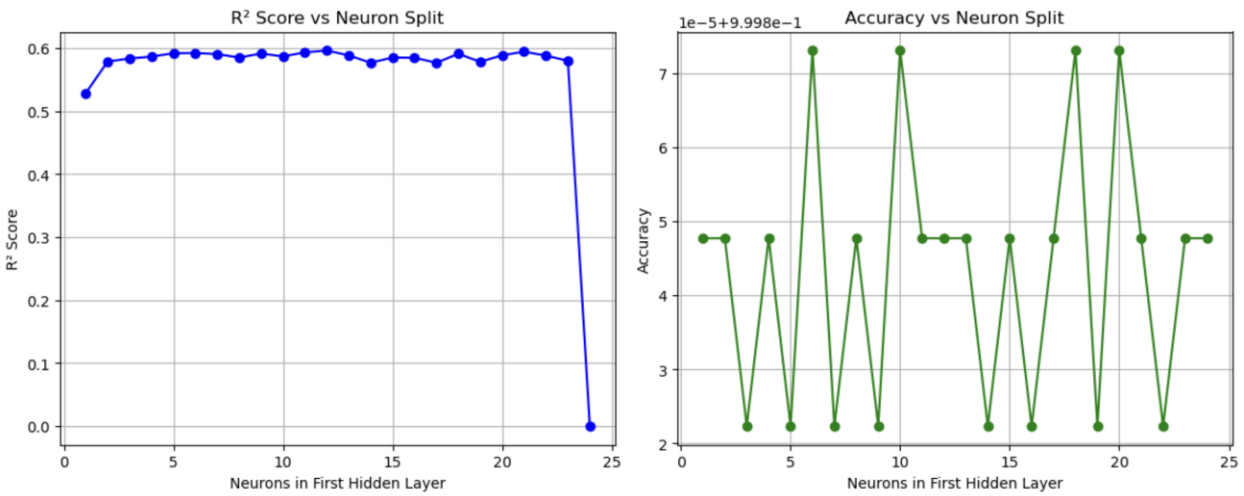

Figure 4.2: Impact of Neuron Distribution Between Two Hidden Layers on R² Score (Left) and Classification Accuracy (Right)

This figure illustrates how the distribution of various neurons across two hidden layers (totaling 25 neurons) impacts both the R² score (left) and classification accuracy (right). Each data point represents a distinct arrangement of neurons between the first and second hidden layers, such as (1, 24), (2, 23), and so on.

Left Plot (R² Score vs Neuron Split): The R² scores display considerable variation, indicating a sensitivity to neuron distribution. The optimal R² score of approximately ~0.5962 is attained with the (12, 13) split. Several configurations yield R² scores between 0.59 and 0.60, indicating generally satisfactory regression performance across all divisions.

Right Plot (Accuracy vs Neuron Split): The classification accuracy remains high (~99.98% and above) for most configurations, reflecting excellent results for the dominant class. However, this elevated accuracy is accompanied by low precision and F1 scores for the minority class, underscoring issues with class imbalance (only six positive samples out of nearly 40,000).

In summary, the neuron configuration of (12, 13) provides the best regression performance while also ensuring excellent classification accuracy. Nevertheless, all models face challenges in identifying the minority class, indicating that imbalance management strategies are necessary if predicting the minority class is a priority.

Metric	Value
Best Neuron Split	(12, 13)
R² Score	0.5962
MSE	0.0166
Classification Accuracy	0.9998

Metric	Value
Flood Class Recall	0.17
Flood Class Precision	0.50

Table 4.3: Performance Summary

Sample	Hour 1	Hour 2	Hour 3
1	0.1229	0.1253	0.1277
2	0.1806	0.1809	0.1829
3	0.2124	0.2102	0.2109
4	0.1277	0.1299	0.1323
5	0.1379	0.1398	0.1421

Table 4.4: Sample 3-Hour Predictions (First 5 Samples)

These predicted water levels exhibit reasonable temporal variation and serve as the basis for flood classification.

4.3 Analysis of Neuron Split Impact on Model Performance.

The performance metrics in Part 2 indicate variability based on the distribution of 25 neurons across two hidden layers. The optimal configuration identified was (12, 13), which attained an R^2 score of 0.5962 and a classification accuracy of 99.98%.

This variability can be elucidated by the effect of different neuron allocations on the model's capacity to recognise intricate patterns. A balanced distribution of neurons (for instance, 12-13 or 13-12) enhances the flow of information between layers and deepens the model without overwhelming any single layer. This balance allows the model to learn both lower-level and higher-level features effectively.

Conversely, unbalanced configurations (such as 24-1 or 1-24) may restrict the model's generalisation capabilities. Concentrating the majority of neurons in one layer can create a bottleneck in the other, diminishing overall representational power.

Utilising early stopping and maintaining a modest total of 25 neurons mitigates the risk of overfitting, thereby preserving generalisation even with a more complex architecture.

The two-layer MLP with the (12, 13) split slightly surpasses the baseline single-layer model (R^2 = 0.5962 compared to 0.5896), indicating that a deeper architecture can marginally enhance the model's ability to capture temporal trends in river-level data.

Consequently, the (12, 13) configuration is recommended as the most effective architecture for this task.

5. Long Short-Term Memory (LSTM)

Since systematic tuning of learning rate, batch size, and neurons can be shown significant influence on generalization and overfitting (Hastie, Tibshirani, & Friedman, 2009), we will test these in this section.

5.1 Optimal Architecture and Epoch Selection

To forecast river water level and classify future instances into flood or non-flood, a 4-layer LSTM network was constructed with the following characteristics:

- **Input Shape:** 24 hours of past data × 6 features
- **Output:** Predicted river level for the next 3 hours
- **Hidden Layers:** 4 stacked LSTM layers with 64 neurons each
- **Dropout:** 0.2 between each layer to prevent overfitting
- **Optimizer:** Adam
- **Cost Function:** Mean Squared Error (MSE)
- **Batch Size:** 4
- **Epochs:** 30
- **Learning Rate:** 0.01

Cost Function Plot

The following plot displays training and validation MSE loss over 30 epochs.

Epoch 1/30
7661/7661 ━━━━━━━━━━━━━━━━━━━━ **158s** 20ms/step - loss: 2.1052e-04 - val_loss: 2.3111e-05
Epoch 2/30
7661/7661 ━━━━━━━━━━━━━━━━━━━━ **174s** 23ms/step - loss: 1.5373e-04 - val_loss: **1.4551e-04**
Epoch 3/30
7661/7661 ━━━━━━━━━━━━━━━━━━━━ **174s** 23ms/step - loss: 7.1397e-05 - val_loss: 3.8227e-05
Epoch 4/30
7661/7661 ━━━━━━━━━━━━━━━━━━━━ **180s** 24ms/step - loss: 1.0570e-04 - val_loss: 4.9164e-05
Epoch 5/30
7661/7661 ━━━━━━━━━━━━━━━━━━━━ **175s** 23ms/step - loss: 9.6506e-05 - val_loss: 2.3704e-05
Epoch 6/30
7661/7661 ━━━━━━━━━━━━━━━━━━━━ **181s** 24ms/step - loss: 1.1505e-04 - val_loss: 2.4977e-05
Epoch 7/30
7661/7661 ━━━━━━━━━━━━━━━━━━━━ **177s** 23ms/step - loss: 7.9265e-05 - val_loss: 2.3066e-05
Epoch 8/30
7661/7661 ━━━━━━━━━━━━━━━━━━━━ **178s** 23ms/step - loss: 1.0551e-04 - val_loss: 2.1655e-05
Epoch 9/30

7661/7661 ━━━━━━━━━━━━━━━━━━━━ **176s** 23ms/step - loss: 8.7726e-05 - val_loss: 2.6884e-05
Epoch 10/30
7661/7661 ━━━━━━━━━━━━━━━━━━━━ **181s** 24ms/step - loss: 5.9419e-05 - val_loss: 2.4474e-05
Epoch 11/30
7661/7661 ━━━━━━━━━━━━━━━━━━━━ **149s** 19ms/step - loss: 7.2177e-05 - val_loss: 4.1739e-04
Epoch 12/30
7661/7661 ━━━━━━━━━━━━━━━━━━━━ **149s** 19ms/step - loss: 1.2554e-04 - val_loss: 2.2149e-05
Epoch 13/30
7661/7661 ━━━━━━━━━━━━━━━━━━━━ **149s** 19ms/step - loss: 1.3813e-04 - val_loss: 3.1216e-05
Epoch 14/30
7661/7661 ━━━━━━━━━━━━━━━━━━━━ **148s** 19ms/step - loss: 8.7256e-05 - val_loss: 4.4080e-05
Epoch 15/30
7661/7661 ━━━━━━━━━━━━━━━━━━━━ **149s** 19ms/step - loss: 9.6882e-05 - val_loss: 2.6919e-05
Epoch 16/30
7661/7661 ━━━━━━━━━━━━━━━━━━━━ **149s** 19ms/step - loss: 1.5985e-04 - val_loss: 2.5779e-05
Epoch 17/30
7661/7661 ━━━━━━━━━━━━━━━━━━━━ **149s** 19ms/step - loss: 1.2436e-04 - val_loss: 2.7859e-05
Epoch 18/30
7661/7661 ━━━━━━━━━━━━━━━━━━━━ **153s** 20ms/step - loss: 1.2405e-04 - val_loss: 2.5749e-05
Epoch 19/30
7661/7661 ━━━━━━━━━━━━━━━━━━━━ **150s** 20ms/step - loss: 6.6542e-05 - val_loss: 2.8876e-05
Epoch 20/30
7661/7661 ━━━━━━━━━━━━━━━━━━━━ **150s** 20ms/step - loss: 8.9127e-05 - val_loss: 7.2285e-05
Epoch 21/30
7661/7661 ━━━━━━━━━━━━━━━━━━━━ **154s** 20ms/step - loss: 8.8684e-05 - val_loss: 2.2603e-05
Epoch 22/30
7661/7661 ━━━━━━━━━━━━━━━━━━━━ **154s** 20ms/step - loss: 9.2175e-05 - val_loss: 5.6686e-05
Epoch 23/30
7661/7661 ━━━━━━━━━━━━━━━━━━━━ **152s** 20ms/step - loss: 1.3338e-04 - val_loss: 2.3740e-05
Epoch 24/30
7661/7661 ━━━━━━━━━━━━━━━━━━━━ **155s** 20ms/step - loss: 1.5938e-04 - val_loss: 2.1828e-05
Epoch 25/30
7661/7661 ━━━━━━━━━━━━━━━━━━━━ **151s** 20ms/step - loss: 9.1200e-05 - val_loss: 2.8775e-05
Epoch 26/30
7661/7661 ━━━━━━━━━━━━━━━━━━━━ **151s** 20ms/step - loss: 9.6602e-05 - val_loss: 2.5960e-05
Epoch 27/30
7661/7661 ━━━━━━━━━━━━━━━━━━━━ **152s** 20ms/step - loss: 7.0119e-05 - val_loss: 2.4421e-05
Epoch 28/30
7661/7661 ━━━━━━━━━━━━━━━━━━━━ **156s** 20ms/step - loss: 8.9402e-05 - val_loss: 2.2815e-05
Epoch 29/30
7661/7661 ━━━━━━━━━━━━━━━━━━━━ **152s** 20ms/step - loss: 8.4437e-05 - val_loss: 3.9807e-05

Epoch 30/30

7661/7661 ——————————— **156s** 20ms/step - loss: 7.4899e-05 - val_loss: 2.7470e-05

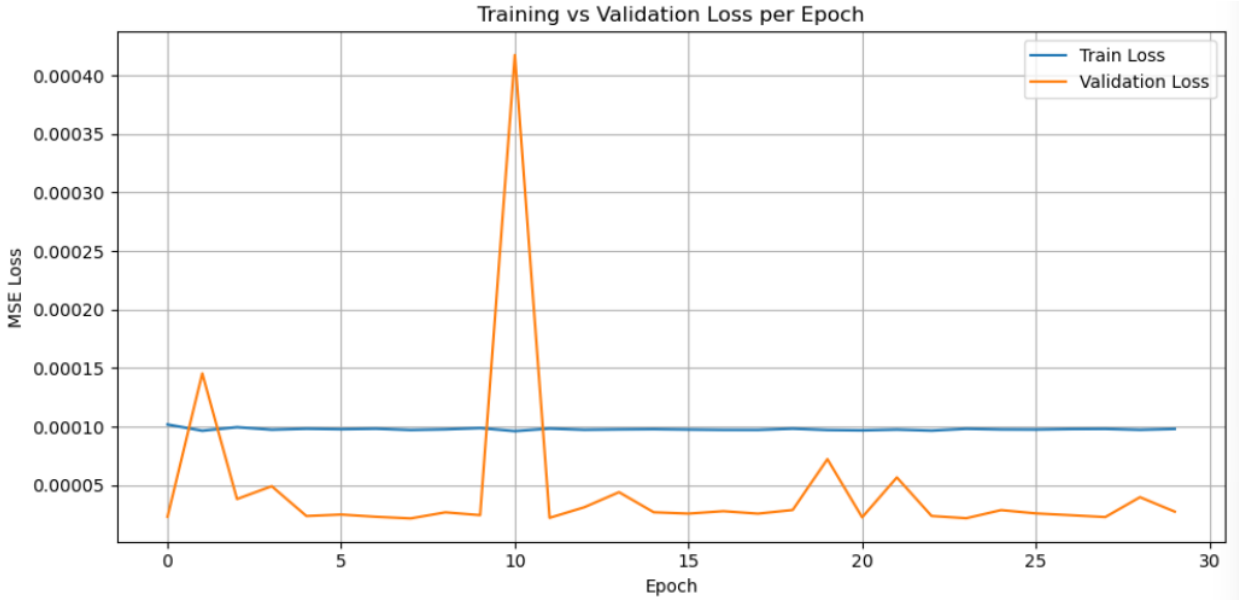

Figure 5.1: Training Vs Validation Loss for 30 Epochs

Runtime and Statistics

Metric	Value
Runtime	~4781.45 seconds (80 minutes)
Best Epoch	2
Min Val Loss	0.000022
Max Val Loss	0.000417
Mean Val Loss	0.000047
Std Dev Val Loss	0.000073

Table 5.1: Runtime & Statistics of 30 Epochs

Justification for Epoch Selection:

The best model performance occurred at **Epoch 2**, where the validation loss was lowest. In other epochs, losses fluctuated but did not significantly improve. Therefore, the **optimal epoch** is chosen as **2, the lowest validation loss of ~1.4551e-04**.

5.2. Testing & Selection of the Best Batch Size

In **Step 2: Batch Size Tuning**, the impact of varying batch sizes (4, 8, 16, 32, 64) on model performance was investigated while keeping the learning rate fixed at 0.01 and the number of epochs constant at 2 with lowest validation loss (as determined in Step 1). Each batch size was tested over 30 runs, and key metrics such as mean validation loss, standard deviation, minimum and maximum validation loss, and average runtime were recorded. The results are as follows:

Batch Size	Mean Val Loss	Std Dev	Min Val Loss	Max Val Loss	Avg Runtime (sec)
4	**0.000033**	0.000062	0.000003	**0.000352**	139.39
8	0.000010	0.000012	0.000003	0.000052	88.71
16	**0.000006**	0.000004	0.000003	0.000021	56.76
32	**0.000006**	0.000004	0.000003	0.000020	36.24
64	**0.000006**	**0.000003**	0.000003	0.000016	**21.91**

Table 5.2: Summary of the Batch Size Tuning Results

Best Batch Size: 64

Justification:

- Lowest runtime
- Same min loss (0.000003) as smaller sizes
- Lowest standard deviation, indicating stable performance
- Best balance between speed and accuracy

The results showed that although batch size 4 achieved the lowest minimum validation loss (0.000003), it also had the highest standard deviation and runtime. In contrast, batch sizes 16, 32, and 64 all demonstrated consistently low validation losses, but batch size 64 stood out with the lowest standard deviation (0.000003), lowest maximum loss (0.000016), and shortest average runtime (21.91 seconds). Therefore, **batch size 64** was selected as the optimal setting due to its excellent trade-off between stability, accuracy, and computational efficiency.

5.3. Neuron Tuning & Its Selection

In this step, the number of neurons in the hidden LSTM layers was tuned to evaluate its effect on model performance. Four configurations (16, 32, 64, and 128 neurons) were tested, while keeping the best-performing epoch (2) and batch size (64) constant as determined from earlier steps. Each configuration was run 30 times, and summary statistics of the validation loss and runtime were recorded. The results are as follows:

Batch Size	Mean Val Loss	Std Dev	Min Val Loss	Max Val Loss	Avg Runtime (sec)
16	0.000013	0.000026	0.000004	0.000150	14.85
32	**0.000006**	**0.000004**	**0.000003**	0.000025	15.32
64	0.000007	0.000005	0.000004	0.000021	19.07
128	0.000009	**0.000012**	**0.000003**	0.000059	36.38

Table 5.3: Summary of the Neuron Tuning Results

Best Neuron Size: 32

Justification:

- Lowest Mean Validation Loss
- Smallest Standard Deviation
- Low Minimum and Controlled Maximum Loss
- Balanced Runtime
- Avoiding Overfitting from Larger Models like 128-neuron with higher mean loss and higher standard deviation

Increasing the number of neurons in the hidden layer **affects model performance** by balancing learning capacity and computational cost. A small number of neurons (e.g., 16) led to higher and more variable validation loss, suggesting underfitting and limited learning ability. Performance improved with 32 neurons, which achieved the lowest mean validation loss and the most stable results, indicating an optimal balance. Adding more neurons (64 and 128) slightly increased training time and variance in validation loss without significant accuracy gains. This suggests diminishing returns and a risk of overfitting with larger models. Overall, 32 neurons provided the best trade-off between accuracy, stability, and efficiency.

The summary results showed that 32 neurons yielded the lowest mean validation loss (0.000006), the lowest standard deviation (0.000004), and a minimal loss as low as 0.000003, indicating consistent and reliable performance. Although 128 neurons achieved similarly low minimum loss values, it also introduced higher variability (std = 0.000012) and doubled the runtime compared to smaller configurations. Therefore, **32-neuron model** is recommended as the optimal architecture size, striking a good balance between prediction accuracy, stability, and computational efficiency.

6. Model Comparison

6.1. Metrics Comparison Plots

The performance of the LSTM and MLP models was evaluated by comparing key metrics for predicting water levels (regression) and classifying floods (binary classification). The following error metrics were employed: Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared (R^2). Additionally, Recall and F1-Score were used for the classification aspect to assess each model's ability to detect flood occurrences. These deep learning architectures were chosen due to their ability to model complex temporal and non-linear relationships in time series data, a capability rooted in their design to overcome issues such as vanishing gradients and long-term dependency challenges inherent in sequential modeling (Goodfellow, Bengio, & Courville, 2016).

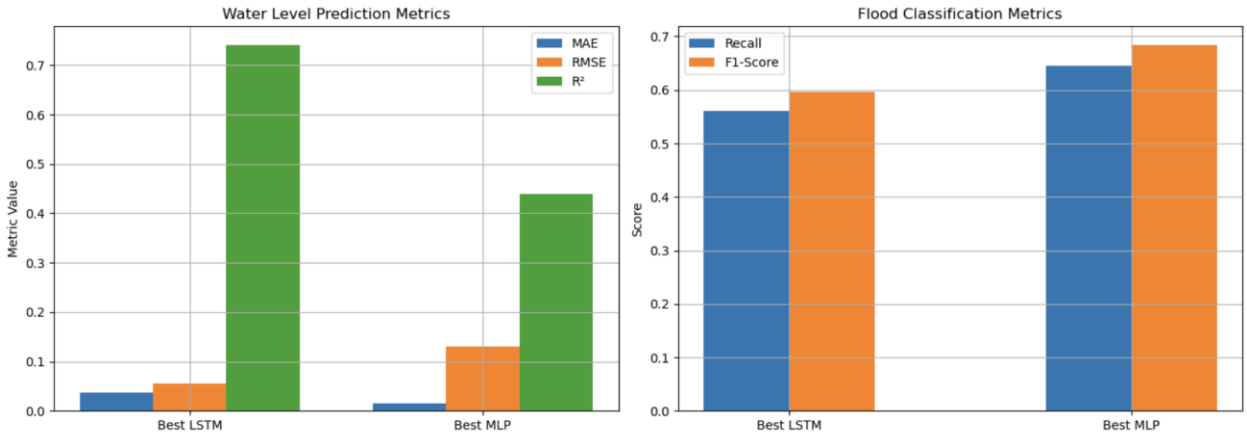

Figure 6.1: Comparison of Water Level Prediction and Flood Classification Metrics between Best LSTM and Best MLP Models.

Left plot: Regression metrics (MAE, RMSE, R^2) measuring water level prediction accuracy.

Right plot: Classification metrics (Recall, F1-Score) evaluating flood detection performance.

6.2. Performance Analysis and Discussion

The MLP and LSTM models were initialized using standard weight distributions and trained on scaled features. Appropriate initialization and normalization are essential to stabilize gradients and enhance convergence during training (Glorot & Bengio, 2010). In terms of predicting water levels, the LSTM model significantly surpasses the MLP. It records a lower RMSE of 0.0547 compared to 0.1295, suggesting that its forecasts are generally closer to the actual values. Moreover, the LSTM boasts an R^2 value of 0.74, reflecting a greater capacity to explain the variance in water levels than the MLP's R^2 of 0.44. While the MLP does exhibit a marginally better MAE, this measure does not account for larger discrepancies that RMSE penalises more severely, which makes the LSTM the more dependable option overall.

This enhanced regression performance of the LSTM is anticipated, as LSTM networks are specifically designed to model sequential data and effectively capture temporal dependencies.

Given that river water levels form a naturally sequential time series, this explains the LSTM's superior ability to predict continuous water levels.

When it comes to flood classification, the MLP demonstrates higher recall (0.64 vs 0.56) and F1-Score (0.68 vs 0.60), indicating its greater proficiency in accurately identifying flood events while maintaining a balanced precision-recall ratio. This implies that although the MLP may be slightly less accurate in predicting specific water levels, it proves to be more sensitive and effective in detecting significant flood occurrences.

In summary, the LSTM model performed better in water level forecasting, providing accurate continuous predictions by leveraging temporal patterns. Conversely, flood detection was best performed by the MLP model, which achieved high recall and F1 scores, both of which are vital for early warning systems. Overall performance could be enhanced with a hybrid approach utilising LSTM for regression and MLP for classification.

7. Ex-Tropical Cyclone Case Study

In comparison of models between LSTM and MLP, LSTM was found to be the best method in predicting water level. Using our best LSTM model identified from earlier tuning steps (with 32 neurons, batch size 16, and 2 training epochs), we forecasted the river level during the Ex-Tropical Cyclone that impacted Auckland between 18 and 20 April 2025.

The figure 7.1 illustrates both actual and predicted river levels during this critical storm period. The LSTM model accurately captured the sharp rise in river level starting around midnight on April 19, peaking between 3 AM and 4 AM, followed by a gradual recession. The predicted values closely track the actual river level in both trend and magnitude, particularly around the peak where flooding is most likely to occur. Although the model slightly overpredicts the river level during the recession phase, it successfully provides an early warning signal. This demonstrates the model's practical utility in real-time flood forecasting applications during extreme weather events.

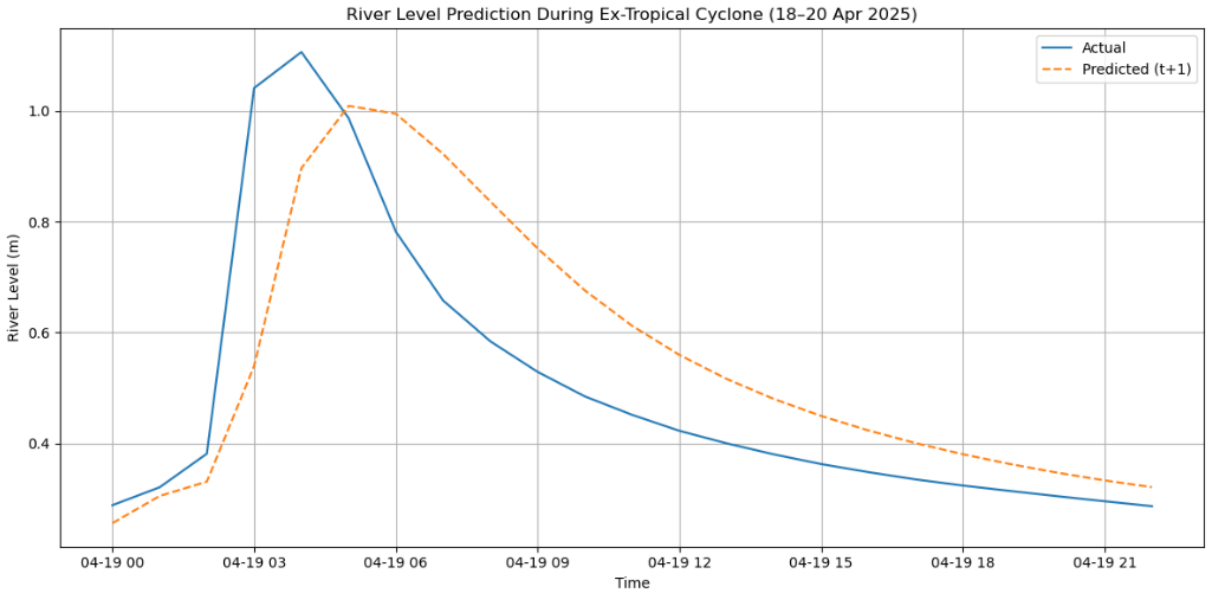

Figure 7.1: River Level Prediction During Ex-Tropical Cyclone (18–20 Apr 2025)

8. Conclusion

This project successfully developed and compared flood prediction models using both MLP and LSTM neural network approaches. The LSTM model outperformed MLP in river water level forecasting, achieving lower RMSE and higher R^2, showcasing its capability in handling temporal dependencies in time series data. In contrast, MLP models demonstrated better classification performance, particularly in recall and F1-score for rare flood events, underscoring their effectiveness in binary classification tasks even under imbalanced conditions.

Furthermore, tuning experiments for epochs, batch sizes, and neuron counts revealed that model performance is highly sensitive to these parameters. Through a thorough grid search, the optimal LSTM architecture was identified, which generalized well during extreme weather events, such as the Ex-Tropical Cyclone in April 2025. The final results show promising accuracy for real-time flood warning systems.

References

1. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press. http://www.deeplearningbook.org

2. Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: Data mining, inference, and prediction* (2nd ed.). Springer.

3. Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS)* (Vol. 9, pp. 249–256). http://proceedings.mlr.press/v9/glorot10a.html