CISC484 Project 5 Report

Peter Chapman

University of Delaware

CISC484: Intro to Machine Learning

Project 5: Weather Parameters Prediction

December 12, 2025

Project Overview

The objective of this project was to use the Long Short-Term Memory (LSTM) recurrent neural network (RNN) model to predict certain weather parameters. Using weather data from the University of South Alabama, this project attempted to make weather predictions about the next four hours using the weather from the previous 12 hours. As requested in the project description, four separate LSTM models were created for the four different weather variables of interest: temperature, wind speed, air pressure, and humidity.

Long Short-Term Memory (LSTM) Model

The created LSTM models were heavily influenced by the demo code provided. The models accept the previous 12 hours of values as input and predict the next four hours of values as outputs. The objective was to enable short-term weather forecasting by learning patterns and trends in the data. This specific model type, the Long Short-Term Memory model, was developed for time-series predictions, which changed the outlook compared to the demo code model.

The LSTM model architecture contains several layers, with many coming from the demo code. The first layer consists of 50 units, and it returns sequences. It has 12 time steps (one for each hour) and allows information to flow to the subsequent layers. The second and third layers, which were asked for from the demo code, both contain 50 units and also return sequences. They continue the learning from the first layer, refining the patterns that were originally found. The fourth layer also contains 50 units, but it does not return sequences. Instead, it outputs a single vector representation. The fifth layer, the dropout layer, attempts to prevent overfitting by randomly dropping neurons during the training process using a 20% dropout rate (unless otherwise denoted). The sixth layer is a dense ReLU activation layer with 25 neurons. This layer allows for complex relationships to be learned through non-linear transformations. A second dense layer follows, and it essentially processes these learned features even more. The final layer, the output layer, uses four neurons to make predictions for the next four hours.

For training, an RMSprop optimizer was used, with an original placeholder of 0.001 for the learning rate. This rate was adjusted for each model, depending on the training, testing, and prediction results. For loss, the mean squared error function was selected due to the fact that it was asked for in the assignment description. The placeholder number of epochs for training was 30, an increase from the 10 epochs included in the demo code. Once again, this number was adjusted, depending on the variable of interest and the success of the training and predicting processes. The number of batches was finalized across all four models to be 32. Metrics tracked for each variable include mean absolute error (MAE), mean squared error (MSE), and root mean square error (RMSE).

These design choices were inspired by the demo code but also the context of this problem. Having multiple LSTM layers allows the model to learn hierarchical temporal features. Including a dropout layer ideally prevents overfitting on the training data. Using ReLU activation introduces non-linearity for complex pattern recognition. Using mean squared error (MSE) for loss is appropriate considering the numerical context of this problem and the problem included in the demo code. The

RMSprop optimizer adapts learning rates for each parameter, which can be very effective for recurrent neural networks.

The data used came from the .csv data set provided in the project description. Variables of interest from the project description included temperature, pressure, wind speed, and humidity. The four chosen variables from the data set were air temperature at two meters in degrees Celsius, wind speed at two meters in what is believed to be meters per second (m / s), relative humidity at two meters (%), and air pressure at two meters in what is believed to be millimeters of mercury (mmHg). The 'Hour' column was used as the datetime index for parsing. Some variables had misleading or incorrect data, so that was addressed before the models were built.

Normalization was performed using the MinMaxScaler, which scaled all values to a range of [0, 1]. This step improved the neural network training by preventing gradient explosion/vanishing, ensuring that all features are on similar scales, and speeding up convergence. Time sequences were also created to reflect the project context of time-series data. Windows of 12 consecutive hours were created as inputs, and the outputs were windows of four consecutive hours.

The training and testing data split followed the traditional method from class. 80% of the data was assigned to the training process, while 20% of the data was saved for the testing process. This provided sufficient training data while reserving enough data for reliable evaluation of the model. Most importantly, the data was not shuffled, instead having a chronological split. This was important for the specific project context of time-series predictions. The goal was to create predictions for four hours based on the previous 12. This meant that the first 12 hours needed to be kept together, and the predictions needed to be the following four hours. Shuffling the data would have prevented the ability to create a strong predictive model for the variables of interest.

Model Results

Temperature

A data issue arose when training a model for temperature: any observations with missing values from the dataset were automatically coded as -273.15 degrees Celsius. This led the first model created to have results that were not beneficial at all. This occurred because the wrong column name from the .csv file was included. The actual variable of interest, air temperature in degrees Celsius at two meters, was not called "Temp_C" as assumed. Instead, it was called "AirT_2m". Outside of this adjustment, the standard 30 epochs, 32 batches, and dropout rate of 20% were used for training the model. 10,355 data points were sent to the training process and 2,589 were saved for testing.

Figure 1 below shows the training and testing losses over the 30 epochs performed for the air temperature model. For both mean square error and root mean square error, the training loss gradually reduced as the number of epochs increased. On the other hand, the validation loss fluctuated heavily, showing no general trend as the number of epochs increased. This suggests that the model may have struggled a little bit with overfitting.

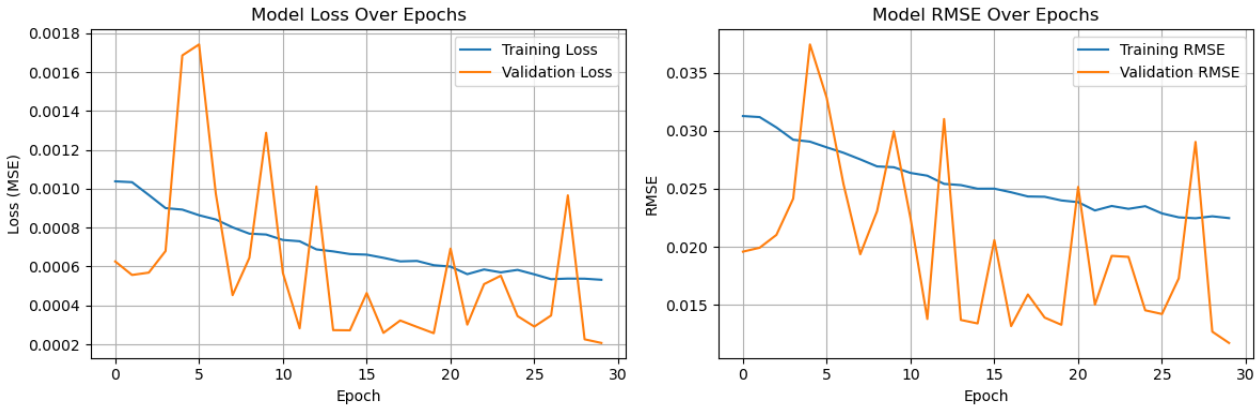

Figure 1: Training and testing errors for the temperature model.

Figure 2 below shows the predicted temperature values vs. the actual temperature values for the testing data. Generally, the LSTM model did an excellent job of predicting the temperatures at two meters in degrees Celsius. The curve for predicted temperature values closely follows the curve for the actual values. The two curves do deviate a little bit, but that is to be expected for a predictive model such as this one. In particular, the predicted values differ from the actual values as the sample index approaches 100. This could suggest that the model would struggle with predictions as the number of samples increases past 100.

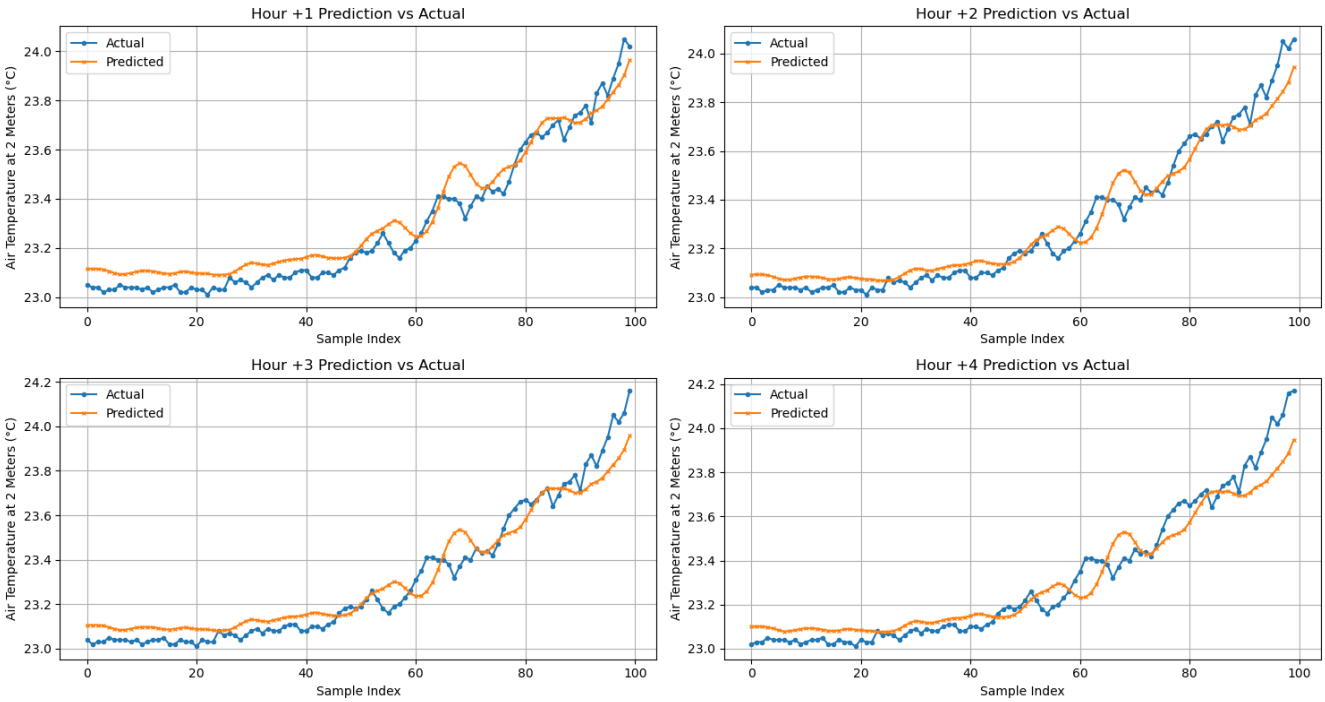

Figure 2: Predictions vs. actual values at each hour for temperature at two meters.

Wind Speed

A few issues came up when training the model and making predictions for wind speed. First, it appeared as though each data point was assigned a wind speed value of 0.0 meters per second if the true value was missing. This was considered bizarre, as the true wind speed is very rarely the exact value of zero. Therefore, the data was adjusted to exclude any data points that included a wind speed of 0.0. The finalized data for this variable included 6,076 data points for training and 1,520 data points for testing.

Additionally, the model initially struggled with training and testing loss with the original setup used for the temperature training process. Therefore, the dropout rate was reduced to 10% and the number of epochs was enlarged to 60. Figure 3 below shows the training and testing losses for the wind speed model. Fortunately, the training loss gradually decreased as the number of epochs increased. Just like the testing process for temperature, the testing loss fluctuated back and forth as the number of epochs increased.

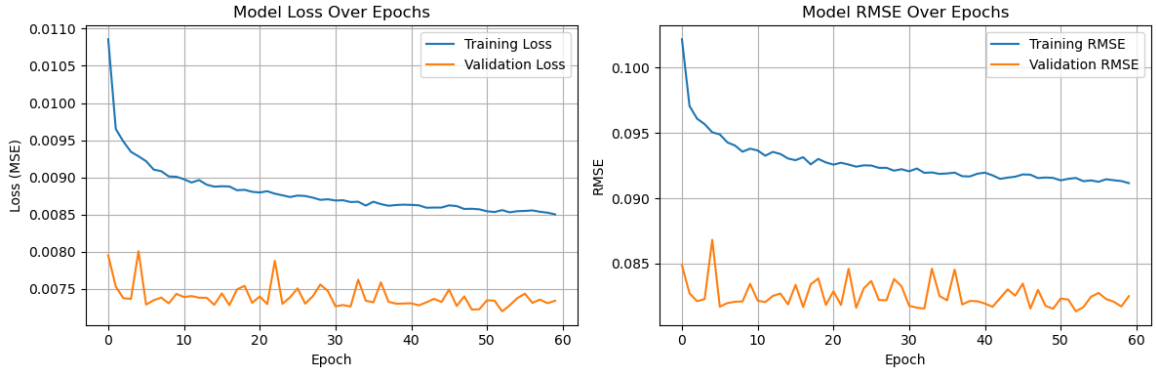

Figure 3: Training and testing errors for the wind speed model.

Figure 4 below shows the predicted wind speed values vs. the actual wind speed values for the testing data. The LSTM model did a very poor job of predicting the wind speed at two meters in meters per second. The model did not account enough for the extreme peaks and troughs in the wind speed values. As the number of samples increases, the changes in the prediction curve are not large enough in magnitude to match the true movement of the wind speed values. Future research and development should be performed on this model in order to find better predictions for the wind speed values.

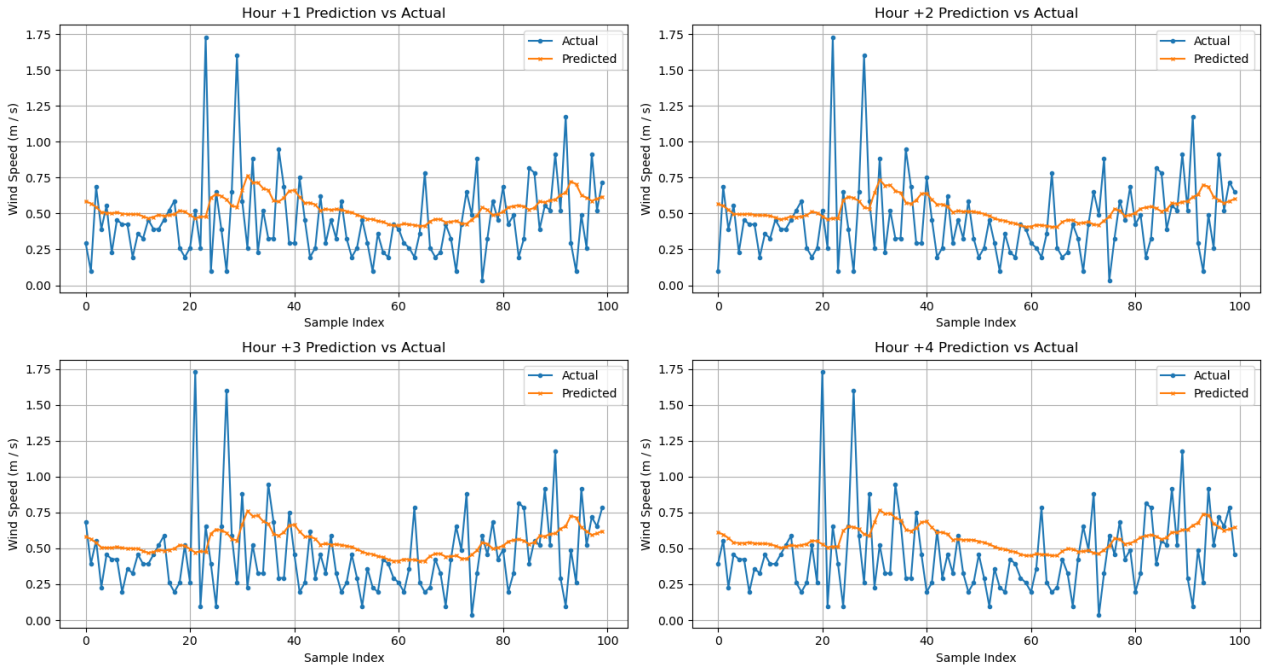

Figure 4: Predictions vs. actual values at each hour for wind speed at two meters.

Air Pressure

Similar to the temperature training process, 10,355 data points were used in training and 2,589 were used in testing for the air pressure model. This model received a few adjustments during the training process and the creation of predictions. The original model that was created struggled heavily to make accurate predictions, so changes were needed. First, the learning rate was lowered from 0.001 to 0.0001. Second, the number of epochs was enlarged from 30 to 45. As shown in Figure 5 below, these changes worked very well, as the training and testing losses both gradually decreased as the number of epochs increased. Oddly, the testing loss randomly spiked upward and then immediately fell back down to the value it was previously at.

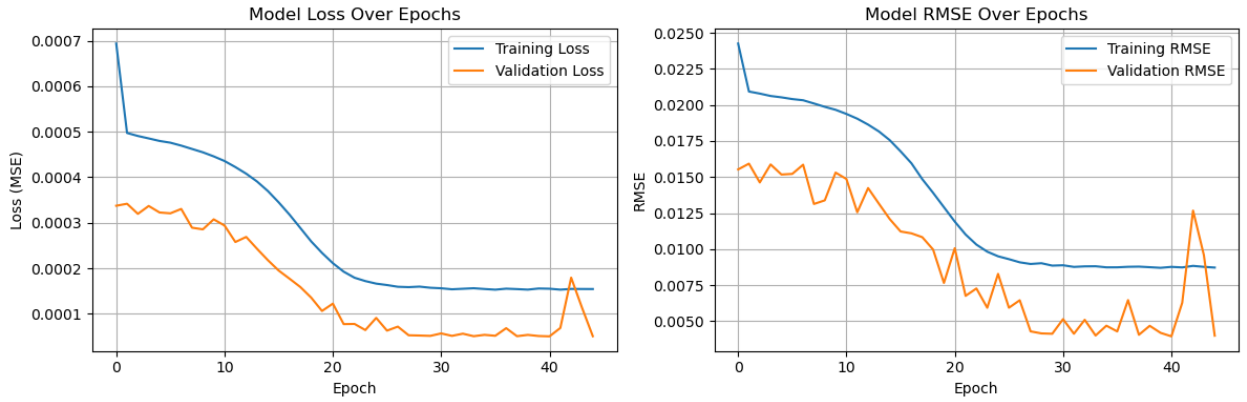

Figure 5: Training and testing errors for the air pressure model.

Figure 6 below shows the predicted air pressure values vs. the actual air pressure values for the testing data. The general trend of the actual air pressure values is matched by the pattern in the predicted air pressure values. However, the prediction curves are all located below the actual air pressure curves, which suggests that the model may have been weighed down by extremely low values in air pressure from the training data. It is important to note that these graphs can be somewhat deceiving from the naked eye. The prediction curve is only under the actual air pressure curve by around one to two mmHg. Therefore, the misses aren't as bad as it may seem. To fix this, it may be beneficial to investigate the summary statistics of the air pressure values in the data from the .csv file.

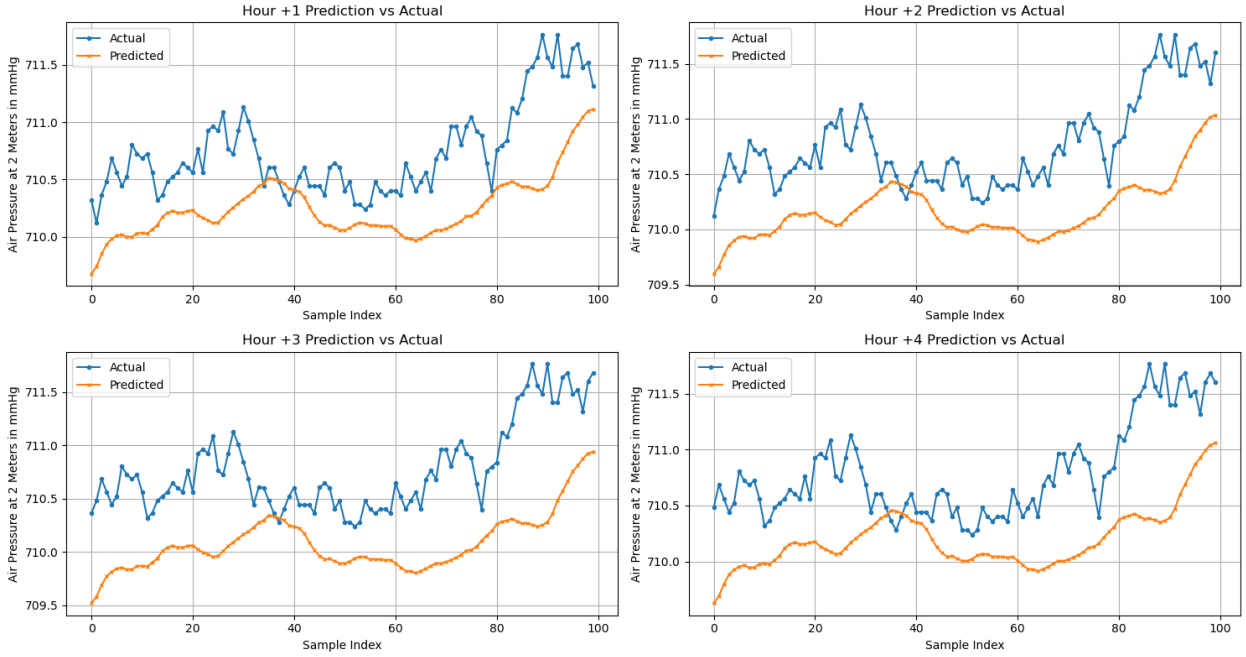

Figure 6: Predictions vs. actual values at each hour for air pressure at two meters.

Humidity

A third data issue arose when the humidity model was created. A suspicious number of data points read '100%' for relative humidity, so it was determined that the original data set assigned this value to any observations that had missing data. To account for this, all observations that read 100% were removed from the training and testing processes. As a result, only 7,246 data points were used for training and 1,812 data points were used for testing.

Like the model for air pressure, the humidity model received a few adjustments during the training process and the creation of predictions. The original model that was created struggled heavily to make accurate predictions, so changes were needed. First, the learning rate was enlarged from 0.001 to 0.005. Second, the number of epochs increased from 30 to 45. As shown in Figure 7 below, these changes

worked somewhat well, as the training and testing losses both gradually decreased as the number of epochs increased. Once again, the testing loss had a few small peaks, but they were minimal this time.

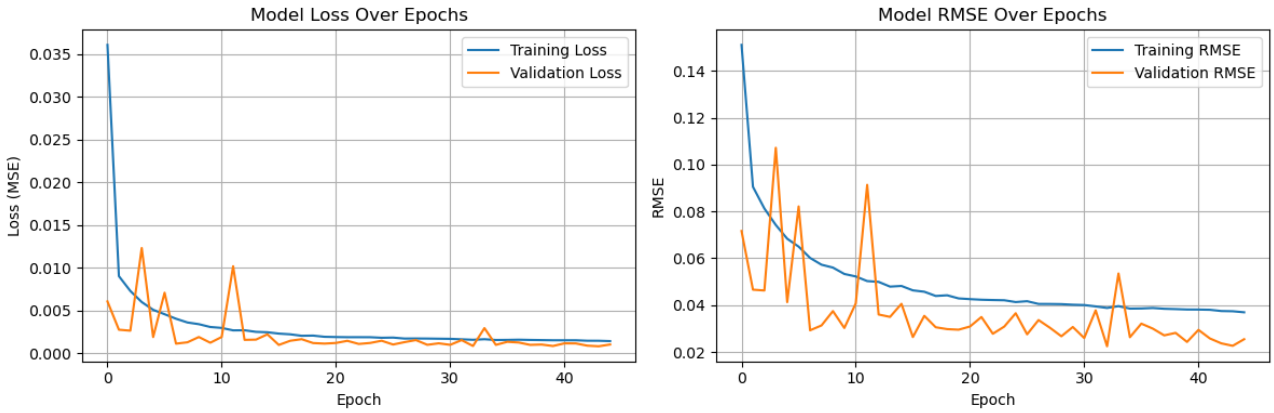

Figure 7: Training and testing errors for the humidity model.

Figure 8 below shows the predicted relative humidity values vs. the actual relative humidity values for the testing data. The general trend of the actual relative humidity values is matched by the pattern in the predicted relative humidity values. However, the prediction curves are all located above the actual relative humidity curves, which suggests that the model may have been hurt by the extreme high values in relative humidity from the training data. On the other hand, it is believed that the model did not account enough for the extreme low relative humidity values in the data. This was the opposite of what was expected: the 100% relative humidity values were removed, but the model's predictions still somehow overshot the actual values for relative humidity at two meters.

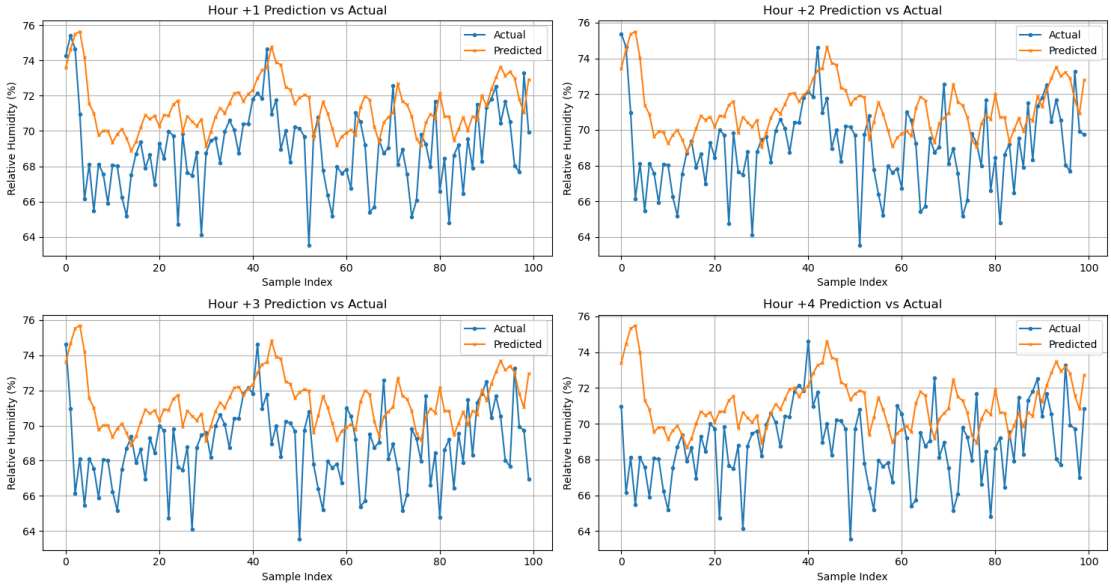

Figure 8: Predictions vs. actual values at each hour for relative humidity at two meters.

Comparing Models

Table 1 below shows the training mean square error (MSE), testing mean square error, training root mean square error (RMSE), and testing root mean square error for each of the four models that was created. This table allows for a comparison between the four models in regard to the training processes that were conducted. As one can see, the air temperature model achieved the best overall performance with the lowest MSE and RMSE values for both training and testing. This indicates that temperature follows more predictable temporal patterns compared to the other weather parameters. The wind speed model also demonstrated strong performance and notably showed minimal overfitting, as evidenced by the small gap between training and testing errors. In contrast, both the relative humidity and air pressure models exhibited signs of overfitting, with training MSE values (4.5592 and 4.7683 respectively) substantially higher than their testing MSE values (2.6247 and 1.6861 respectively). This unexpected pattern, where testing error is lower than training error, suggests that the testing data may have contained less variability or fewer extreme values than the training set, particularly given that 100% humidity values were filtered from the dataset. Despite these differences, all four models had some success in predicting their respective weather parameters with reasonable accuracy, with the temperature model showing the most reliable generalization to unseen data.

Model	**Training MSE**	**Testing MSE**	**Training RMSE**	**Testing RMSE**
Air Temperature	0.0338	0.0325	0.1839	0.1803
Relative Humidity	4.5592	2.6247	2.1352	1.6201
Air Pressure	4.7683	1.6861	2.1836	1.2985
Wind Speed	0.2164	0.1882	0.4652	0.4339

Table 1: Training and testing results for the four models.