Forecasting the Unseen: Predicting Relative Humidity With Classical and Deep Learning Approaches

IT1244 Team 2

Genie Tay Ting, Hui Yu Chao, Gan Zhi Yu Charlene, Yeo Jaye Lin

Introduction and Literature Review

A reliable, and more importantly, accurate weather forecast is paramount with far-reaching implications for decision making at a country and global scale. In particular, the prediction of relative humidity (RH) is crucial as various sectors across the economic landscape are impacted when relative humidity fluctuates. A gradual shift towards more humid conditions, even in temperate regions, can serve as early indicators towards rising temperatures and accelerated global warming (Gunawardhana et al., 2017). Many essential industries, such as agriculture, are highly sensitive to RH fluctuations. They depend on relative humidity conditions to be stable and predictable to maintain crop yield and avoid disruptions in food supply chains (Tibbitts, 1979). Current literature surrounding the prediction of relative humidity focuses on three main machine learning paradigms:

- **Classical Learning**: Zainkarmar and Mayorga (2021) employed traditional models like Random Forest and *k*-NN to facilitate outdoor relative humidity prediction.
- **Deep Learning**: Nagaraj and Kumar (2023) used several deep learning approaches, including Gated Recurrent Unit (GRU) and Feed-Forward Neural Network (FFNN), to capture daily relative humidity trends in Chennai, India.
- **Hybridised**: Evans at al. (2025) have adopted a hybrid approach that synergizes machine learning techniques (such as Support Vector Machine and Neural Networks) with statistical approaches (such as ARIMA) to bolster their predictive capabilities.

These works provide valuable in providing an understanding how relative humidity can be modeled but exhibit certain limitations in generalisability and temporal forecasting range. Some papers also lack the presence of hyperparameter tuning, opting instead to fix the hyperparameters to certain pre-defined numbers.

In this report, our approach towards predicting relative humidity accurately will also consist of both classical and deep learning models across seasons and time horizons.

Dataset Overview and Preprocessing

Our dataset comes from the file `weather.csv`. It consists of 7 columns: 6 predictor variables – temperature, wind speed, mean sea level pressure, surface solar radition, surface thermal radiation and total cloud cover – and one dependent variable – relative humidity.

Although the dataset lacks an explicit temporal parameter (e.g. date or time), a closer inspection of the `surface_solar_radiation` column reveals a recurring pattern: sequences of 24 rows where the transition between 0 to non-zero values align with a daily cycle. This strongly suggests that the dataset is recorded at hourly intervals, with values containing '0' indicating night-time. With this finding, we applied the following data preprocessing and feature engineering steps:

- **Outlier Detection**: The function `detect_out()` was used to find outliers that fail the 1.5 * IQR rule.
- **Normality Check**: QQ Plots were plotted to access the normality of each column of the dataset, revealing that most were non-Gaussian.
- **Skewness Correction**: Yeo-Johnson transformation was used to make the data columns more Gaussian. This also reduces the skewness of our dataset to mitigate the influence of outliers.
- **Normalisation**: Values were normalized after transformation to scale them to the 0 to 1 range, ensuring uniformity across all columns.
- **Outlier Removal**: A small number of remaining outliers, even after skewness correction and normalisation were dropped (12 rows).
- **Multicollinearity Check**: Multicollinearity was verified using Variance Inflation Factor (VIF). Most of our features had VIF values within the 1 to 3 range, suggesting slight correlation but no critical issues.
- **Correlation Analysis**: We also accessed feature dependencies using a heatmap (Figure 1). Again, we found no significant dependencies.

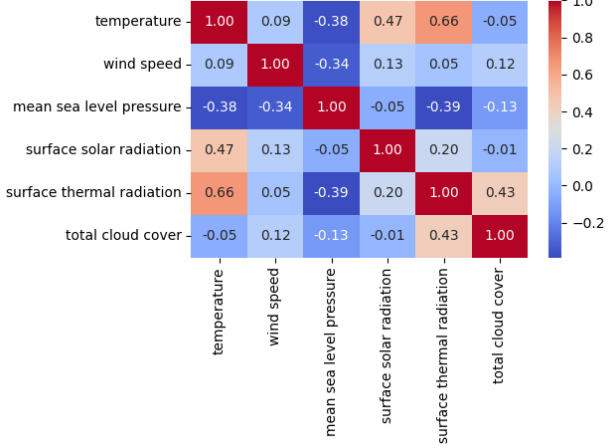

FIGURE 1: *Heatmap of Variables*

With preprocessing complete, we generated our final input and target `numpy` arrays our `final_X` and `final_y` which we will use in our ML algorithms.

Methods

As outlined in the introduction, our study employs a mix of classical and deep learning techniques, with two models from each category.

For classical learning, we selected supervised learning methods Multiple Linear Regression (MLR) and Random Forest (RF) as our main approaches. These were chosen for their simplicity and interpretability. MLR acts as our foundational base model, and RF can model non-linearities and is robust to overfitting. Although RF was not covered in this course, a brief overview is provided to ensure clarity.

For deep learning, we used Recurrent Neural Networks (RNN) and Long-Short Term Memory (LSTM) as our main models. These were chosen given its pervasiveness and widespread use sequence modelling and prediction tasks (Gunawan et al., 2023). LSTM was included as a counterpart to RNN due to its additional ability to capture long-term dependencies, making it suitable for forecasting over longer time horizons. This would allow for a meaningful comparison between the methods that rely on short-term (RNN) vs long-term (LSTM) temporal dependencies.

We opted not to use statistical time series models such as Autoregressive Integrated Moving Average (ARIMA) or its variants (e.g. SARIMAX, State-Space ARIMA) as they require the use of past values of the target variable for prediction, which is not feasible given our restrictions of not using humidity data from any time point for prediction.

To enhance model performance, we also decided to perform hyperparameter tuning for the following:

- N: number of historical datapoints used for predicted the target value
- η: base learning rate of algorithm (specific to deep learning approaches)
- e: number of epochs

Our models will also be adopting timeseries k-fold cross validation using TimeSeriesSplit() function from the sklearn library to make our models generalize well and reduce overfitting by modelling across multiple train-test splits instead of relying on a single test set. We selected $k = 4$ as cross validation can be computationally expensive, given how large our dataset is, and how time complexity may exponentiate for more computationally demanding deep-learning models. Our data was organized in a 70-10-20 split (70% train, 10% validate during training, 20% test). The test set will only be used when running our final model after training. This structure ensures fair evaluation and helps us draw accurate conclusions about model performance.

Results and Discussion

Two metrics were used to access the effectiveness our model: Root Mean Square Error (RMSE) and Mean Absolute Error (MAE). MAE is used due to its ease of interpretation as it represents the average magnitude of the prediction errors. RMSE, however, penalizes larger errors due to the squaring of differences. This makes it useful in determining if a model can accurately capture outliers or sharp variations in relative humidity (e.g. sharp spikes or dips). In general, the RMSE would be higher than the MAE due to RMSE's heavy penalisation of outliers.

In each model, the number of timepoints (N) used for prediction was finetuned

Multiple Linear Regression (MLR)

MLR is a linear model used to establish multiple input features and the target variable, estimating the relationship between the dependent and independent variables, by fitting a linear equation to the data points. The only parameter that required tuning for MLR was the value of N, and we used time series split and rolling window to cross validate our model on the training data and to tune N. We found that the optimal value of $N = 24$ which is equivalent to one day, suggesting that the data points in the span of one day had the most linear relationship. No other hyperparameters were tuned for MLR.

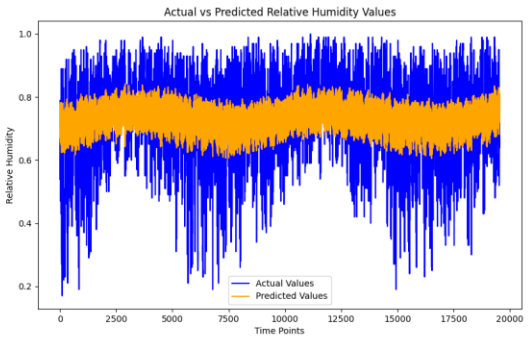

FIGURE 2: *Predicted vs Actual Values for MLR for 1 hour forecast. N = 24.*

We see from Figure 2 that MLR, while able to somewhat capture the trend, fails at properly capturing the actual value, with most of the predicted values falling within a narrow band. Furthermore, outliers were also not accurately capture by MLR. This is further exemplified by the high error values used for our metrics, with RMSE = 0.125 and MAE = 0.100.

Random Forest (RF)

Random Forest is an ensemble learning method. It constructs multiple decision tree regressors, and outputs the average of the trees. One of its strengths includes capturing non-linear relationships between variables such as temperature and wind speed, due to the combination of the decision trees, which can learn different aspects of the weather data.

To implement this regressor, we used 'RandomForestRegressor' from the module 'sklearn.ensemble', with its n_estimator, representing number of trees, set at 25, max_depth at 5, min_samples_split at 15, and random_state at 42. This is to balance between preventing overfitting and reducing computational cost.

TABLE 1: *RMSE and MAE comparison between different forecast horizons with best parameters for RF*

Delay	*N*	RMSE	MAE
1 hour	36	0.0872	0.0689
6 hours	24 and 36	0.1057	0.0813
24 hours	12 and 36	0.1134	0.0877

Between *N* values of 12, 24, and 36, representing half days, full days, and 1.5 days, the RMSE and MAE were lowest at *N* = 36 for delay = 1, representing predicting 1 hour in the future. It is expected that the prediction capabilities would be best for lesser time in the future. However, the RMSE and MAE remain relatively similar for the different *N*s when predicting 1 day in the future (delay = 24). Perhaps the model would need even larger *N*s, such as 168 representing a week, to detect weekly cycles that help the RF to predict the future humidity more accurately.

Recurrent Neural Network (RNN)

RNN is a form of sequential deep learning model that is suitable in capturing the relationship between data points in a sequence. In a time-series data, RNN would be able to perform forward prediction of humidity using past *N* time points.

Critically, the number of time points (*N*), epoch sizes and base learning rates (η) were finetuned for optimal model performance. Figure 2 demonstrates how the loss and RMSE vary with epoch number. This is to obtain a good epoch number which we can use in our final RNN model to determine the *N* and η values.

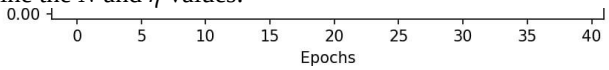

FIGURE 3: *Loss / RMSE vs Epoch Characteristic Curves for SimpleRNN. N = 24, η = 0.0005.*

We see from Figure 3 that the loss drops quickly within the first few epochs and stabilises early which indicates that the model learns efficiently from training data. More importantly, the validation RMSE follows closely with the Training RMSE, suggesting that the model does not overfit and thus good generalisation. Most of our other models with higher neuron size are overfitted, with the green curve further above the orange curve. We will thus keep our neuron size low and keep our number of epochs *e* to 35, which is the value before the spike of the green curve in Figure 3. The RNN error values obtained in our final model are: RMSE =2.14 and MAE = 2.09. These values are okay, but we can do better, through the use of LSTM.

Long Short-Term Memory (LSTM)

Like the RNN model, hyperparameter tuning was performed to determine the best *N* and η values that would yield the best performance in our LSTM model. We perform the same plot as RNN to determine the best epoch value, which is 15 (see Annex 1). The *N* and η values tested, were:

N = [24, 168, 720] # daily, weekly, monthly
η = [1e-6, 1e-5, 1e-4]

We used Adam's optimiser from the keras library, whose learning rate is adjusted according to the different η values above. Due to the complex internal gating mechanisms present in LSTM, large learning rates tend to cause the optimiser to overshoot the minima. Conversely, smaller learning rates may lead to extremely slow convergence without meaningful performance gains. We thus aim to find the optimal η value that strikes a balance between training efficency and model accuracy.

We performed hyperparameter tuning, in addition to 4-fold cross validation, across three forecast horizons: 1-hour, 6-hour and 24-hour ahead. Below displays the results of the three models, including each of their best parameters:

TABLE 2: *RMSE and MAE comparison between different forecast horizons with best parameters for LSTM*

Forecast Horizon	***N***	***η***	**RMSE**	**MAE**
1 hour	168	1e-6	0.0587	0.0457
6 hours	168	1e-6	0.0851	0.0638
24 hours	168	1e-6	0.2184	0.1792

We can conclude from Table 2 that our model performs best at 1-hour forecast when using past 24-hour data to predict. This aligns with expectations as it demonstrates that shorter-term predictions are generally more accurate, since they are more dependent on patterns closer to the target. We can see this in Figure 4 below, where the orange lines (predicted) nicely overlaps with the blue line (actual values). The predicted model of LSTM was also, more importantly, able to capture short-term outlier behaviour. This is shown by how the sudden spikes in the blue curve also are overlapped with spikes in the orange curve, with minor lapses here and there.

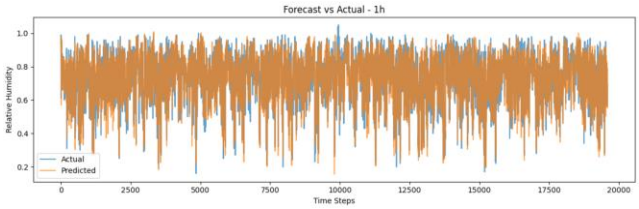

FIGURE 4: *Predicted vs Actual Values for LSTM for 1 hour forecast. N = 168, η = 0.0001.*

Nonetheless, the low RMSE and MAE scores for all forecast horizons stand as a testament to the fact that LSTM can still quite accurately handle both long-term and short-term predictions of relative humidity, compared to RNN.

Conclusions and Future Research

In conclusion, our study demonstrates the effectiveness of both classical and deep learning approaches in forecasting future relative humidity. Across all models, with their respective optimal parameters, we observe consistently low RMSE and MAE values, thus showing a high level of predictive accuracy.

Among the models evaluated, LSTM for the 1-hour forecast horizon achieved the lowest RMSE and MAE, highlighting the strength of deep learning in capturing short-term temporal dependencies. This suggests that, while classical models offer simplicity and interpretability, deep learning models (e.g. LSTM), can provide greater robustness and accuracy due to their ability to learn complex patterns from data.

This is not to downplay the effect of classical learning models, though. Both models are meritous in that they provide valuable insights into the area of time-series forecasting. Their combined use can provide a comprehensive strategy for more accurate and reliable weather forecasting statistics.

Our research can be extended further by looking into tuning other hyperparameters as well, such as the number of training epochs and the sequence length used in deep learning models. These factors may yield further improvements to model performance and generalization. A look into other types of ML algorithms, such as XGBoost, can also provide meaningful results and enhanced prediction accuracy as well.

Bibliography

Evans, L., Chen, I., Patel, R., Zhang, Z., Mitchell, A., & Walker, E. (2025). Developing a Hybrid Weather Forecasting Model Using Machine Learning and Traditional Approaches. https://doi.org/10.13140/RG.2.2.18496.03843

Gunawan, F. E., Budiman, A. S., Pardamean, B., Juana, E., Romeli, S., Cenggoro, T. W., Purwandari, K., Hidayat, A. A., Redi, A. A. N. P. & Asrol, M. (2023). Multivariate time-series deep learning for joint prediction of temperature and relative humidity in a closed space. *Procedia Computer Science*, *227*, 1046-1053. https://doi.org/10.1016/j.procs.2023.10.614

Gunawardhana, L. N., Al-Rawas, G. A., & Kazama, S. (2017). An alternative method for predicting relative humidity for climate change studies. *Meteorological Applications*, *24*(4), 551-559. https://doi.org/10.1002/met.1641

Nagaraj, R., & Kumar, L. S. (2023). Univariate Deep Learning models for prediction of daily average temperature and Relative Humidity: The case study of Chennai, India. *Journal of Earth System Science*, *132*(3), 100. https://doi.org/10.1007/s12040-023-02122-0

Tibbitts, T. W. (1979). Humidity and plants. *BioScience*, *29*(6), 358-363. https://doi.org/10.2307/1307692

Zarinkamar, R. T., & Mayorga, R. V. (2021). Outdoor Relative Humidity Prediction via Machine Learning

Techniques. *JOURNAL OF ENVIRONMENTAL INFORMATICS LETTERS*, *6*(2), 114-124. https://10.3808/jeil.202100074

Annexes

Annex I:
Determining the Best Epoch Value for LSTM (*e* = 15)

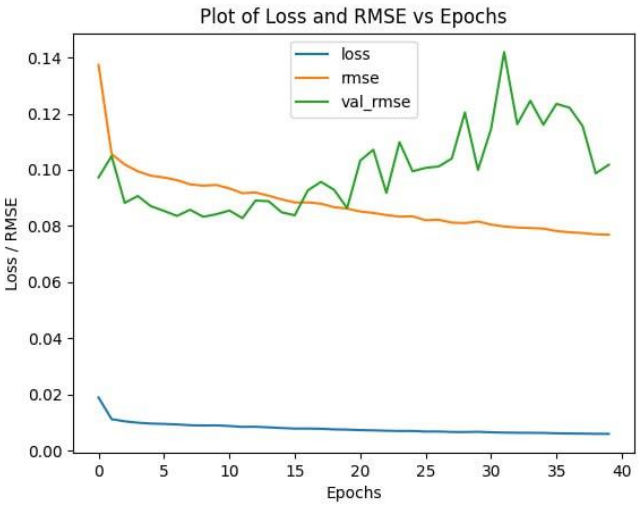

FIGURE 5: *Loss / RMSE vs Epoch Characteristic Curves for LSTM. N = 24, η = 0.0005.*

Similar to the explanation for RNN, we see that the model starts to overfit past *e* = 15, when the green line starts to diverge from the orange line. We thus chose *e* = 15 as our epoch number for LSTM.