ML Meteorologist: AI-Based Weather Prediction

1st Malak Al-Noubani
University of Texas at Arlington.
Department of Bioengineering
Arlington, TX
maa1421@mavs.uta.edu

2nd Bryan Nguyen
University of Texas at Arlington.
Department of Computer Science
Arlington, TX
bhn8449@mavs.uta.edu

3rd Rabib Husain
University of Texas at Arlington.
Department of Computer Science
Arlington, TX
rxh3770@mavs.uta.edu

***Abstract*—We propose ML Meterologist, a deep learning model that predicts weather conditions in Arlington, TX. We explore different approaches to optimizing this process by implementing a recurrent neural network involving long short term memory blocks. Models capable of predicting 3 hour increments within 24 hours are trained, classifying weather into 4 categories: clear, rainy, reduced visibility, and snowy. An accuracy greater than 85% accuracy was achieved on the test set. Lastly, we implement this model into a user-friendly graphical user interface for model deployment with Gradio.**

***Index Terms*—weather prediction, machine learning, neural networks**

I. Introduction

A basic function of human life is interacting with the outside world. As the fields of science and technology advanced, society has grown better prepared to deal with uncontrollable weather. Predicting the weather is important for transportation, agriculture, disaster management, safety, and deciding what to wear.

Humans have been trying to predict and understand the weather since Aristotle's time (citation). Modern meteorology started developing in the mid-17th century, and advancements in meteorological technology have been advancing since. Numerical-based models developed in the 1960s were used to predict weather, in which equations are solved for modeling atmospheric conditions [4]. Only one prediction is made based on the best probability for initial conditions. However, it struggled with long-term forecasting. Ensemble predictions were born around 2003, in which a set of predictions are made based on probability modelling of multiple initial conditions. This model shows promise for medium-range predictions, however, it still does not factor in observed frequency and is weak for long-range. With the introduction of machine learning, we propose to determine advanced methods of weather prediction beyond mathematical modelling.

In modern times, with advancements in technology, people now look to mobile weather apps along with their local weather stations to prepare for the day. These apps utilize weather models in order to make the most accurate prediction of the upcoming weather.

We propose ML Meterologist, a deep learning model that predicts weather conditions in the Dallas-Fort Worth (DFW) area. We will explore different approaches to optimizing this process by implementing neural networks. Lastly, we implement this model into a user-friendly graphical user interface for model deployment.

II. Literary Review

A. Weather Forecasting Using Machine Learning Techniques [1]

Data collection and preprocessing: This paper talks about using three different methods (Support Vector Machine, Artificial Neural Network, and a Recurrent neural Network) for weather predictions. This paper includes the implementation details of each and the results of their implementation. First of all, before performing any training, they preprocessed the data, which consisted of air temperature, atmospheric pressure at the weather station as well as sea level, humidity, wind direction, cloud cover, dew point temperature, and a few other things. This was all within a period of 12 years located at airport stations in India, ranging from 2006 to 2018. For the preprocessing itself, they filled in empty values utilizing linear interpolation for most values except for the dew point temperature, which was 0. An important thing that they kept in mind when choosing features is plotting relationships between the features to see if they are dependent on each other; if there shows a high dependence of one variable on the others then it doesn't really provide any meaningful info.

Training: Moving onto the first training technique, they utilized a linear SVC for temperature prediction using regression. As for the ANN, they utilized five layers with activation functions of relu for most of them except the output layer which ended off with a linear activation function. Since this is a temperature prediction, there is only one neuron at the end for the final prediction. For the others, they ranged anywhere from 5 to 32 neurons. A standard Adam optimizer with batch size of 32 was used and trained for 100 epochs. Lastly, for the RNN, the data was preprocessed further in order to make it into a time series, then normalized for utilization of the sigmoid function. Compared to the ANN, they changed the batch size from 32 to 256. 20 epochs were used this time; with 100 iterations instead of using loss change for termination. This RNN was used for humidity, temperature, and pressure predictions.

Testing: They realized that RNN is better than SVM and ANN for these predictions. The RNN had a much lower RMS error than the other two models, and ANN itself had a much lower error than SVM. Looking at the graphs themselves,

SVM was very erratic in its predictions, and ANN could not predict days with extreme lows/highs very well. RNN was clearly the best candidate.

Conclusion: As we are building a weather app that takes into consideration the data of previous years as well as the current conditions, it may be preferable to use an RNN. Even though weather predictions can be seemingly erratic and random at times, the RNN might be able to pick up on certain patterns within historical data and apply it to current data. Choosing a window of data is absolutely critical as well; we should not pick winter months to forecast summer temperatures, etc. However, it should be noted to keep other models in mind as they may have their own merits in terms of predictions.

B. Using recurrent neural networks for localized weather prediction with combined use of public airport data and on-site measurements [2]

Data collection and preprocessing: In this study, the researchers used two types of weather data. First, they took public weather data from Logan Airport in Boston, which is available through a database called the National Solar Radiation Database (NSRDB). This includes information like temperature, wind speed, humidity, and solar radiation. Second, they collected their own local data from weather sensors set up at Harvard Gund Hall and HouseZero, which are buildings located a few miles away from the airport.

The reason for using both sources is that public weather data may not always reflect what's really happening at a specific building site. Local buildings, trees, and urban conditions can change the weather slightly. So, the researchers compared the airport data to their local measurements and found big differences in things like temperature and humidity, even though the locations weren't far apart. To prepare the data, they cleaned it up, matched the time steps, and set up a system where they looked back at a few days of data to help the model make better predictions.

Training and testing: The researchers tested three types of machine learning models: a simple feed-forward neural network (FFNN), a long short-term memory network (LSTM), and a gated recurrent unit model (GRU). These are all tools that help computers find patterns in time-based data, like weather over several days. The goal was to predict local weather more accurately using mostly the airport data and just a small amount of local data.

They trained the models using data from January to May and then tested how well the models predicted weather from July to December. The GRU model worked the best—it made predictions that were much closer to the actual local weather data. For example, it had the lowest error score (called mean square error), much better than the other models and the airport data alone. This means the GRU model was best at capturing things like daily temperature changes and other patterns.

Conclusion: The study showed that it is possible to use a small amount of local data combined with public weather data to predict more accurate, location-specific weather. The GRU model performed the best and was able to create weather data that matched real conditions very closely. This is important for building design and energy planning, where using accurate weather data can lead to better decisions about things like heating and cooling systems. They also found that having at least four months of local data helped improve predictions a lot, and using data from a nearby weather station gave even better results. The method they used is helpful for architects, engineers, and city planners, especially when there isn't enough time or money to collect a full year of weather data. It can also support better climate planning and building efficiency in the future.

C. Probabilistic weather forecasting with machine learning [3]

Data collection and preprocessing: Google Deepmind's GenCast predicts the future weather based on the current (and 6 hours earlier), presenting a conditional probability problem. It was trained on 40 years of global weather data from 1979 to 2018 from the ERA5 database. More than 80 variables are used from the datasets for training. Instead of using raw weather data, reanalysis data is used. This means that the data was reconstructed with a data analysis method to try to construct what the weather conditions were over the entire globe.

Training: GenCast is based on a conditional diffusion model, commonly used for image generation. This method can generate a probability distribution of target data as well as form new predictions. Diffusion models have a noising and denoising process. The model adds noise to real weather data, then learns how to denoise it in order to reconstruct the actual data. The noise is representative of a Gaussian based on the spatial information of the spherical nature of the Earth. It produces 50 predictions for ensemble weather predicting to be similar to ENS methods.

The encoder and decoder for the noise uses a graph-transformer model to compute attention. The noisy data is what inputted into a multilayer perceptron model. The GenCast denoiser only has access to the last 12-hours of its prediction data so it does not rely on its predictions for a longer period during training, unlike other models. The denoiser also relies on minimizing mean squared error of increments of 12-hour forecasts. Furthermore, the model was trained in two stages with transfer learning. In the initial training the training data was downsampled, and they did finetuning on the second pass with the full resolution of data.

Testing: The ensemble technique (ENS) was used as the baseline for GenCast's results since they are aiming to outperform traditional weather methods using 50 different ensembles. 2019 was used as the test set. It was found that GenCast had better predictions than 96% of the targets in the test set, with a significant difference on 78%. It predicted humidity, geopotential, and sea conditions well. It only performed worse on some instances of temperature and wind.

Conclusion: GenCast was the first machine learning based model to outperform the ensemble prediction models. It can predict weather conditions for 15 days in the future while

being faster and less computationally expensive than running traditional weather models. It can be used to model the trajectory of a typhoon before it even happens.

Although this model is much more complex and computationally expensive than what we intend to implement, this successful model can provide insights to data preprocessing and testing methods. By using the ENS as a benchmark and considering using preanalyzed data as the inputs of the model, ML Meteorologist can improve.

Foxus on Local Modelling: ML Meterologist takes a unique approach from these documents by specifically training a model on one location, which will be useful to create an application for UTA students for day-to-day preparation. We will also explore how only specializing in weather patterns of one city may affect the results and applicability of the model.

III. Methodology

A. Dataset

To train ML Meterologist, weather data is needed from the Arlington area. Weather archive history from the Dallas / Fort Worth Airport (weather station 72259) was utilized from Raspisaniye Pogodi. The dataset comes with many features in which relevant data for weather prediction are chosen.

TABLE I
Training Features

Features	Units	Types
Air temperature at 2 meters high	℃	Continuous
Atmospheric pressure	mmHg	Continuous
Relative humidity at 2 meters high	%	Continuous
Mean wind direction	meters/second	Continuous
Mean wind direction	Compass points	Categorical, Nominal
Total cloud coverage	%	Continuous
Low air temperature	℃	Continuous
High air temperature	℃	Continuous
Horizontal visibility	kilometers	Continuous
Dewpoint temperature	℃	Continuous
Amount of precipitation	millimeters	Continuous
Duration of precipitation	hours	Continuous
Month	0-12	Categorical, Ordinal
Day of the month	1-31	Categorical, Ordinal
Hour of the day	0-24	Categorical, Ordinal

For RNN training in a sequential manner, training data was selected from April 21, 2020 - April 21, 2024 includes 11,690 weather observations and the testing data from April 21, 2024 - April 21, 2025 includes 2,929 data points.

Data Preprocessing / Cleaning / Labeling: Some data preprocessing steps were performed to prepare the Excel file data for model input. The data was treated as a Pandas DataFrame. The datetime for each weather observation was provided as a string in the format data.month.year hour:minute, and to include useful information to the neural network we split this into four features, namely hour of the day, day of the month, and the month.

Amount of precipitation was included as a numeric value measured in milliliters, except for observations listed as 'Trace of precipitation,' so we transformed this to be the smallest value of precipitation at 0.01 milliliters.

Some records of the minimum and maximum temperature were missing. This was handled with mean interpolation, paired with forward and backward fills for the borders.

Cloud coverage was provided as a percentage, such as 100%. Some observations were ranges listed as a string, such as 70-80%. For uniform understanding, we took the mean of the range to replace it. Some datapoints were missing, so we implemented a forward fill since cloud coverage is not expected to drastically change within hours without an observation in the data.

Mean wind direction was provided as a string of compass directions, such as 'Wind blowing from northeast' or 'Wind blowing from the south' so one-hot encoding was implemented to replace this categorical variable in a way that a machine learning algorithm can make sense of. Missing or zero data in this feature was assigned to be 'Calm, no wind' prior to encoding.

Horizontal visibility is a variable related to fogginess. It is given as a measure of distance. On foggy days the data is recorded as 'less than 0.1.' We transformed this string to be a numeric number of 0.09 to keep this feature as a continuous value.

Finally, after ensuring that all the continuous features were being read as numeric values, a standard scaler is applied for feature scaling. This will ensure that features with larger values naturally than others will not inherently overpower the model during gradient descent.

The target variable is the weather condition, in which 19 different categories exist. Missing data was replaced with clear skies.

TABLE II
Weather Observations from April 2020-April 2024

Target Value	Counts
Clear skies	10,944
Heavy Snow	1
Moderate Snow	6
Slight snow	16
Freezing drizzle	2
Drizzle	86
Dusty	4
Fog	16
Hazy	17
Ice pellets	6
Mist	188
Freezing rain	2
Heavy rain	23
Moderate Rain	43
Slight rain	266
Squalls	2
Severe Thunderstorm	12
Thunderstorm with hail	1
Thunderstorm	55
Total	**11,690**

We reduced these into four categories for ML Meteorologist

to classify:

Target Variables

- **Clear skies:** clear skies and squalls (10,946 counts)
- **Rainy:** drizzle, mist, moderate rain, heavy rain, slight rain, thunderstorm, severe thunderstorm, and thunderstorm with hail (674 counts)
- **Reduced visibility:** hazy, fog, dusty (37 counts)
- **Snowy**: heavy snow, moderate snow, slight snow, freezing drizzle, ice pellets, freezing rain (33 counts)

Missing data was interpolated depending on the temperature and amount of precipitation. If there was precipitation and normal temperatures, it was assigned to be rainy. If there was precipitation and there was freezing temperatures, it was assigned as snowy. Any remaining data was marked as clear. These categorical variables were label encoded before being input to the model: clear, 0; rainy, 1; reduced visibility, 2; snowy, 3.

B. Model Selection

For our model, we decided we wanted to take into consideration the conditions of the five most recent previous days of the date that we are predicting. As such, we would need to take into account a sequence of consecutive moments of time. The dataset we used measured eight points of time for each day, so each time series had a length of 40. Since we are predicting using a time series as input, it only makes sense to utilize a recurrent neural network, as mentioned earlier in the paper. At first, we attempted to use a SimpleRNN layer, but with longer time series, it tends to suffer from the vanishing gradient problem (or exploding gradient in the other direction). Instead, we decided to implement a Long Short-Term Memory (LSTM) layer with 64 units which deals better with long-term memory at the cost of being more complex by utilizing carries. For this layer, we used the default hyperbolic tangent (tanh) activation and sigmoid recurrent activation function. Next, we put in an additional three dense layers with a dropoff of 0.25 after the first layer. The hidden layers had units of 128 and 32, respectively. For the dense layers we used ReLU since it is the most widely used activation function for general use. Since we converted the weather results to integer class labels in our data preprocessing, we utilized a softmax activation function for the output layer in combination with a sparse categorical cross-entropy for our loss function for multiclass classification. Finally, we utilized an Adam optimizer to train for 20 epochs since that also tends to be most favorable for general model training. The base model summary is shown below.

TABLE III
MODEL SUMMARY

Layer Type	Shape	Parameters
LSTM	(None, 64)	24,576
Dense	(None, 128)	8,320
Dropout	(None, 128)	0
Dense	(None, 32)	4,128
Dense	(None, 4)	132

C. Model Training

Once the model was designed and compiled, we passed in all of our training data that we generated randomly from the list of times. We randomly extracted a time within the acceptable bounds, retrieved its corresponding weather output, and then obtained all 44 time points before it to form one time series. After randomly picking 12,000 samples, each a time series of length 40 with 32 features for each data point, it was passed into the model along with their respective output for each sample. This process was repeated eight times and saved as a model each time since we wanted to predict every third hour of the next day; this refers to 3:00, 6:00, 9:00, etc. Each model had their own respective target values based on each inputs' time series. All eight models are shown further into the paper. Since we based our training data on the Arlington area, it is clear that our data is skewed, which is affecting the training. Most days are just plain sunny; an overwhelming amount, in fact (around 95 percent!). After performing some tests, it was more difficult for the model to predict weather outside of a normal sunny day.

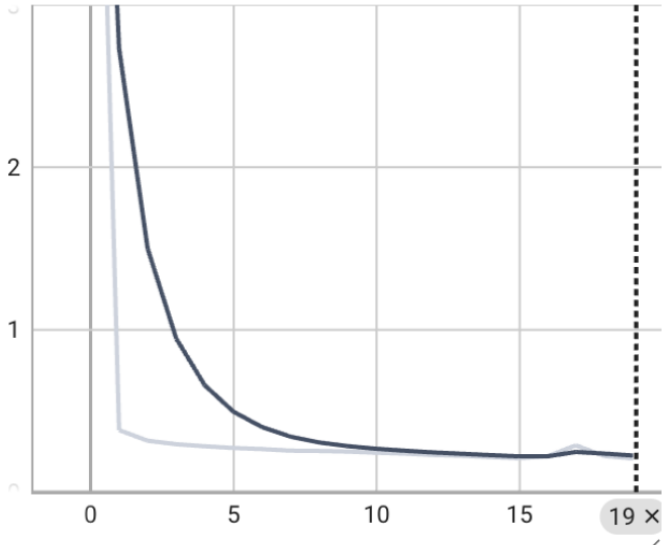

Fig. 1. Loss during training over 20 epochs.

Some initial attempts were made to modify the model to help mitigate this effect to form the architecture described above; we did not want to change the data itself too much since, in general, it is primarily only sunny in these regions. In the end, we grouped up the classes into four main classes for easier classification: clear, rainy, snowy, and reduced visibility. For reference, there were about 10,944 for the clear skies class label, and 674, 33, and 37 for the other respective categories. Randomly generating time series from these data points will generate a heavy skew in our data. Perhaps if we chose a different area to train with, there would be less of a problem; however, we wanted to focus on our local region.

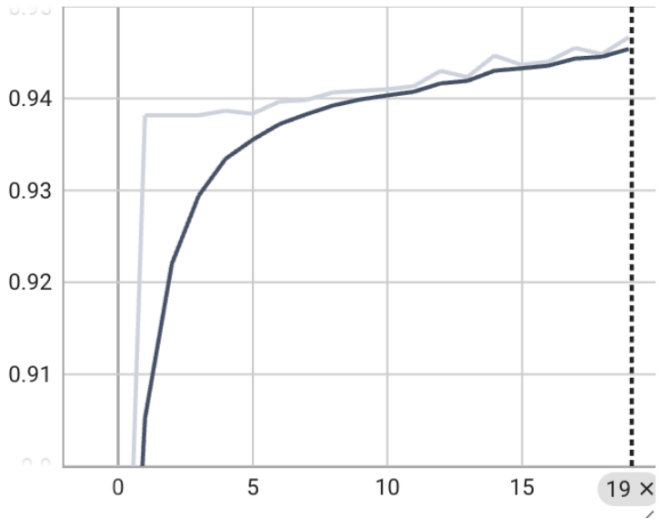

Fig. 2. Accuracy during training over 20 epochs.

D. Testing Procedure

Data Preprocessing: The testing phase was designed to evaluate the trained LSTM weather prediction model on unseen data to assess its real-world applications. We used records from April 2024 - April 2025 which is completely separate from the data used for training. We prepared this test data exactly like we did with the training data:

- Missing value handling.
- Categorical Encoding: Wind direction (DD) was one-hot encoded using the same encoder fitted on the training data to avoid introducing new categories.
- String-to-Number conversion:
  - "Trace of precipitation" string in precipitation amount was replaced with 0.01 milliliters.
  - Cloud cover entries were converted to averages.
- Normalization: All numerical features were standardized using the training dataset's StandardScalar.
- Temporal feature engineering.

Sequence generation: To match the inputs structure, 4,000 random sequences of length 45 timesteps were extracted from the test dataset:

- Random Sampling: Sequences were generated by randomly selecting start/end points across the test period (April 2024–April 2025) to ensure diversity in weather conditions.
- Label Alignment: Each sequence target was the weather label at the next timestep.

IV. Results

Model Inference: The trained LSTM models were evaluated on the test sequences using:

- Forward pass: The model processed each sequence to output a probability distribution over weather classes.
- Performance Metrics: Test Accuracy shown below.

TABLE IV
Test Results

Model	Training Loss	Test Accuracy
3 hours ahead	0.1842	0.867
6 hours ahead	0.1936	0.874
9 hours ahead	0.1786	0.856
12 hours ahead	0.1945	0.864
15 hours ahead	0.1947	0.874
18 hours ahead	0.1976	0.875
21 hours ahead	0.2024	0.867
24 hours ahead	0.2228	0.862

To ensure the model's reliability with real-world data, we validated its robustness through several key measures. By testing exclusively on post-training data from 2024-2025, we verified the model could genuinely predict new weather patterns rather than simply recalling training examples. We implemented random sampling of weather sequences to prevent any location-specific biases from skewing results, while maintaining strict data hygiene by applying the exact same preprocessing steps used during training, including standardized scaling and encoding, without any adjustments to the test data. For future reproducibility, we established protocols to replicate the random sequence selection process consistently. This comprehensive validation approach guarantees that our performance metrics accurately represent the model's predictive power when faced with completely new, real-world weather scenarios, demonstrating its practical utility beyond just theoretical exercises.

V. Visualization

For model implementation, we created a user interface that displays the predictions for the upcoming 24 hours. Our weather model can make predictions for 3-hour intervals from the accessed time.

Fig. 3. ML Meteorologist User Interface

Back-end

ML Meterologist is designed to make predictions based on the previous five days. That means to make predictions in real-time, the application needs to access the current weather variables in addition to data from each multiple of 3 hours for the past 120 hours to fit the sequence length of the input

shape. The current weather conditions in Arlington can be accessed from a free API from OpenWeather, including the temperature, air pressure, humidity, wind speed and direction, cloud coverage, horizontal visibility, low temperature, high temperature, and precipitation amount. Precipitation duration is not explicitly provided by this API. However, the precipitation amount is given per hour, so if precipitation exceeded 0 millimeters, it was filled with a duration of 1 hour. Wind direction was given by degrees, so we used mapping with the 16 cardinal directions with an additional 'Calm, no wind' category to match the expected wind direction encoding.

One challenge that we faced was attempting to access the current dew point temperature, since it was only available in the upgraded version of the OpenWeather API. We overcame this by using a different library called Meteostat, which provides historical weather data. For the dew point temperature, the most recent value is accessed and utilized. The current date and time can be accessed through the Datetime Python module, as well as calculating the times for each of the needed previous days' data that will be accessed. This all accounts for 1 data row consisting of the current conditions. Data validation was conducted by checking the units from the documentation and converting values if necessary.

The other 39 data rows will be previous data, highlighting 3-hour intervals of the past 5 days (3, 6, 9, 12, . . . , 111, 114, 117, and 120 hours ago). The Meteostat library was utilized to obtain the necessary information from each time point. The Hourly and Daily classes from Meteostat provide different information. Since the minimum and maximum temperatures are only calculated each day, it only needs to be accessed once per day of the month from the Hourly class. Other features such as wind speed and direction, precipitation, air pressure, dew point temperature, humidity, horizontal visibility, and precipitation change on an hourly basis, and are accessed through the Hourly class. Reconstruction of the obtained data to match the input order and shape of the RNN model was conducted, in addition to data validation to ensure the units are correct.

The numerical categories in this implementation dataset are standardized, following the data preprocessing procedure of the training and testing datasets. Finally, the processed data is fed into the 8 different models for the 3-hourly predictions for the upcoming 24 hours. Once again, since the models' output is a softmax, the predicted classification with the highest score is considered: 0: clear skies 1: rainy 2: Reduced visibility 3: Snowy

Front-end

Rather than attempting to predict precise numerical values for various weather parameters, the models focus on classifying weather conditions into four primary categories: Clear, Rain, Fog, and Snow. This classification approach simplifies the prediction task while still providing actionable information to end users. The models are loaded dynamically at runtime, with the system gracefully handling any loading errors that might occur. The system leverages the Meteostat library to access real-time and historical weather data, creating a robust foundation for predictions. For current conditions, the application queries location-specific weather parameters for Arlington, Texas, including temperature, pressure, humidity, wind speed and direction, cloud cover, visibility, and precipitation. The Gradio-based user interface transforms complex predictions into an accessible, user-friendly format. The interface presents forecasts in a timeline view, with each time period displaying the predicted conditions, relevant icons, and helpful suggestions for users. The visualization layer incorporates contextual awareness, dynamically adapting the display based on multiple factors. Most notably, the system determines whether each forecast period falls during daylight or nighttime hours and selects appropriate icons accordingly. For rain and snow predictions, the system displays precipitation probability percentages, offering users additional information for planning purposes. Each weather condition is also accompanied by practical suggestions, such as reminders to carry an umbrella during predicted rain or to drive carefully in foggy conditions. Each forecast period is presented in a clearly defined row with four distinct information columns:

- Temporal context - Displaying both the day of the week and specific time (e.g., "Mon 3 pm").
- Weather condition - Featuring the contextual icon and descriptive label.
- Practical suggestion - Offering actionable advice tailored to the specific weather condition.
- Precipitation information (Only if it's raining or snowing) - Showing probability percentages for rain and snow conditions.

VI. Conclusion

ML Meteorologist demonstrates the successful applications of deep learning techniques to weather prediction for the Arlington, Texas area. By implementing an LSTM-based RNN architecture, our model effectively classifies weather conditions in four categories (clear skies, rainy, snowy, reduced visibility, and snow) with impressive accuracy rates, achieving more than 85% accuracy in our test set.. Our data preprocessing steps were crucial to the model's success, particularly our handling of the significant imbalance in Arlington's weather patterns, where clear skies dominate approximately 95 percent of the observations. By thoughtfully categorizing the 19 original weather conditions into four meaningful classes and implementing appropriate feature engineering, we created a model that can effectively identify less common weather events despite their relative rarity in the training data. The development of a user-friendly interface that displays predictions at 3-hour intervals provides practical utility for UTA students and Arlington residents. By incorporating real-time data from OpenWeather API and historical data from Meteostat, the application bridges the gap between sophisticated machine learning models and everyday users who need reliable weather forecasts for daily planning. Additionally, our model was likely not optimal or the most accurate; we could always adjust hyperparameters such as the number of hidden layers, the type of those layers, as well as the number of units within each

layer. We could have also preprocessed the data differently so our training could perform differently. Our method of preprocessing was purely based on our initial knowledge on the correlation of certain input variables with the weather itself. As we are not experts in forecasting, preprocessing may have gone wrong in certain areas. We can explore different avenues for our model. For one, our model was restricted purely to Arlington; we could have tried out a different city or perhaps expanded to generalize to a statewide region. Another idea we had in mind was to predict not the weather as a classification problem, but rather to predict something like the minimum/maximum temperature instead as a regression problem. Because of the widespread availability of datasets and ease of use from Python libraries,, we can definitely pivot our model to suit a different purpose based on our needs. With the expansion of machine learning approaches to common analytical processes, new patterns and relationships can be discovered. Exploring weather prediction algorithms, traditional and modern, can help society become better equipped to face the days ahead.

ACKNOWLEDGMENTS

We acknowledge Mai Anh Do for providing extensive weather knowledge.

REFERENCES

[1] Singh, Siddharth and Kaushik, Mayank and Gupta, Ambuj and Malviya, Anil Kumar, Weather Forecasting Using Machine Learning Techniques (March 11, 2019). Proceedings of 2nd International Conference on Advanced Computing and Software Engineering (ICACSE) 2019, Available at SSRN: https://ssrn.com/abstract=3350281 or http://dx.doi.org/10.2139/ssrn.3350281.

[2] Han, J. M., Ang, Y. Q., Malkawi, A., & Samuelson, H. W. (2021). Using recurrent neural networks for localized weather prediction with combined use of public airport data and on-site measurements. Building and Environment, 192, 107601. https://doi.org/10.1016/j.buildenv.2021.107601.

[3] Price, I., Sanchez-Gonzalez, A., Alet, F. et al. Probabilistic weather forecasting with machine learning. Nature 637, 84–90 (2025). https://doi.org/10.1038/s41586-024-08252-9.

[4] Remi Lam et al. ,Learning skillful medium-range global weather forecasting.Science382,1416-1421(2023).DOI:10.1126/science.adi2336.

AUTHOR CONTRIBUTIONS

All authors collaborated on the introduction. Each author completed the following sections:

- Bryan Nguyen: Weather Forecasting Using Machine Learning Techniques, Model Selection, Model Training, Testing, Conclusion
- Rabib Husain: Using recurrent neural networks for localized weather prediction with combined use of public airport data and on-site measurements, Testing Procedure, Model Inference, Front-end, Conclusion
- Malak Al-Noubani: Probabilistic weather forecasting with machine learning, Dataset, Back-end