Stormcast LSTM

Author Names — *Eric Gerner, Alvaro Gonzalez, Ben Johnson-Gomez*
Institution — *Texas A&M University–San Antonio*
Course — *Machine Learning – Term Project*
Date: December 5, 2025

---

1. Introduction

1.1 Motivation

- Why this sub is important
  ML is the future, so it's best to get a head start in learning it.

- Real-world problem that this addresses
  Floods and other natural disasters

1.2 Problem Statement

- What we are trying to predict/classify
  Rainfall

- Significance of our prediction?
  Rain is a common natural phenomenon that can lead to more severe weather.

1.3 Objectives

- Goals
  We want to use LSTM to predict rainfall and how much of it at least a day before it happens.

- success criteria
  Regressor captures MSE and MAE, and classifier captures accuracy and F1 score.

---

2. Methodology

2.1 Dataset Description

- **Source of the dataset (e.g., Kaggle Weather Dataset)**
  A dataset that has all of the major cities in the USA, and rainfall from 2024 to 2025
  Link

- **Number of samples, features**
  Total Samples: 3,655 (Individual timestamps/days after filtering).

  Number of Features: 7 (Temperature, Humidity, Wind Speed, Precipitation, Cloud Cover, Pressure, Rain Tomorrow).

- **Preprocessing (cleaning, handling missing values)**
  Missing Values: used ffill() followed by bfill().

  "Forward Fill" (propagate the last known good weather to the next hour) followed by "Backward Fill" (fix any gaps at the very start of the file).

  Column Selection: Automated numeric filtering using

- **Location filtering**
  Selected Location: "New York"
  Since it was looking for the city with the most amount of data, and this was it. As well as it being changed if coded in and future implementations could grab as many cities.

- **Scaling (e.g., MinMaxScaler)**
  Method: MinMaxScaler(feature_range=(0, 1))
  Why: LSTMs are sensitive to large numbers. This squashed all values (like Pressure 1013 and Rain 0.5) into the same 0.0 to 1.0 scale so the model wouldn't get confused by the size differences.

- Train/valid/test split
  Sequence Length (Lookback): 14 steps (The model looks at the past 2 weeks).
  Split Ratio: Training: 80% and Testing: 20%
  Validation: Used the Test Set during training (validation_data (X_test, y_test)) to monitor performance live.

2.2 Training Procedure

- **Hyperparameters (epochs, batch size, lookback window)**
  Lookback Window: 14 Days (The model sees the past 2 weeks to predict the next day).

Batch Size: 32 (The model processes 32 days of data at a time before updating weights).
Optimizer: Adam (Adaptive Moment Estimation).
Learning Rate: 0.001

- **Software/libraries**
  Deep Learning Framework: TensorFlow (Keras API)
  Data Manipulation: Pandas, NumPy
  Data Preprocessing: Scikit-Learn (MinMaxScaler for normalisation)
  Visualisation: Matplotlib, Seaborn

- **Adam Optimizer**
  We used the Adam optimizer because it handles sparse gradients well and converges faster than standard SGD (Stochastic Gradient Descent) for time-series data.
  Purpose: Since rain is rare (e.g., 10% of days), the model tends to become "lazy" and predict "No Rain" constantly. Class weights force the model to treat a "Rain" error as significantly more important (e.g., 9x penalty) than a "No Rain" error.

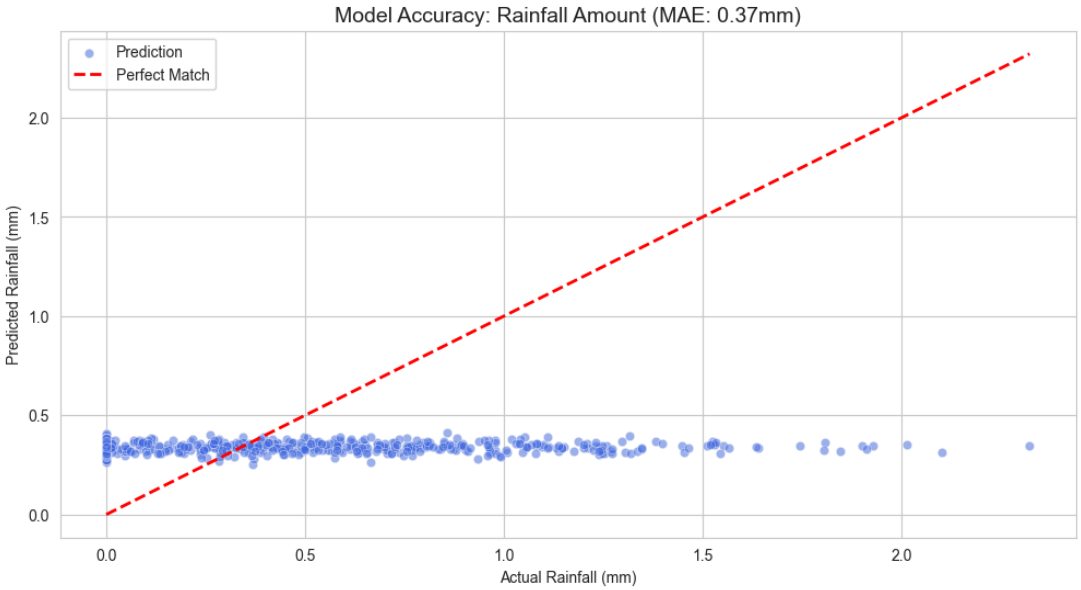
- **Early stopping**
  Used In: Regression Model (train_regressor.py)
  Monitor: val_loss (Validation Loss)
  Patience: 10 Epochs
  Behavior: If the model stops improving for 10 consecutive epochs, training stops automatically to save time and prevent overfitting. It then restores the best weights from history, not the last ones.

---

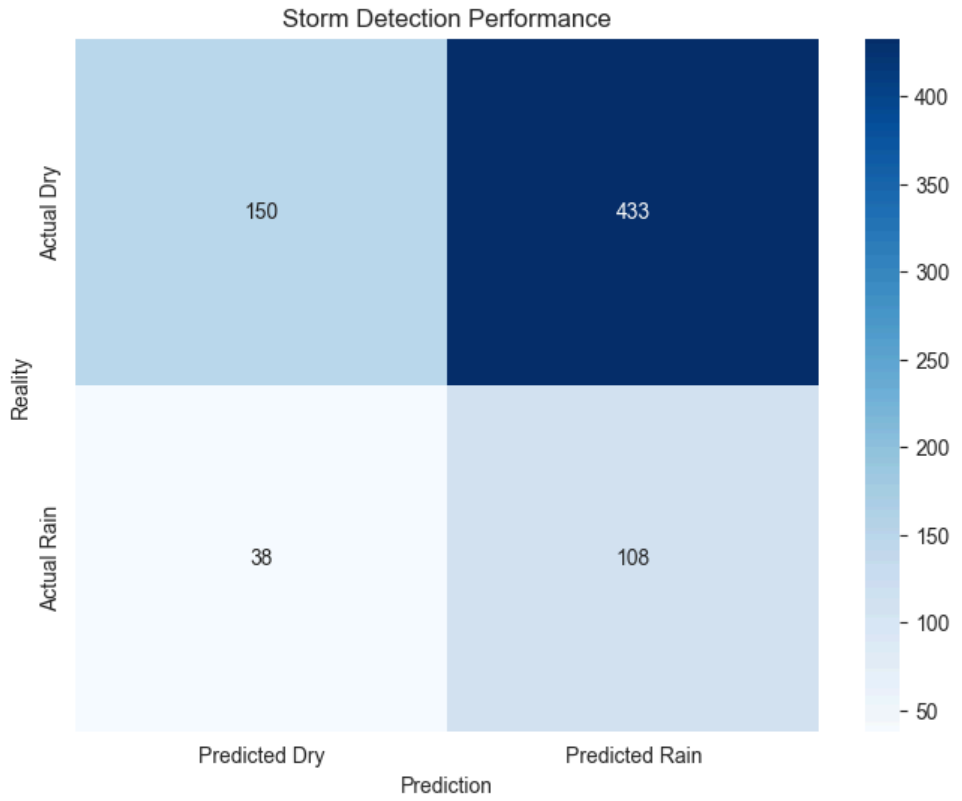
3. Experiments and Results

3.1 Evaluation Metrics

Regression:



Classification:



3.2 Quantitative Results

Table 1 — Regressor Performance

Metric	Value-
Test MSE	0.37
Test MAE	0.2091

Table 2 — Classifier Performance

Metric	Value
F1 Score	0.35

3.3 Analysis

- Explain which model performed better and why
  The classifier, due to its task, which is wondering whether it'll rain tomorrow, is much simpler than estimating the exact amount of precipitation.

- Interpret errors and pattern
  - Underestimation of high-rainfall days
  - Difficulty capturing sharp transitions
  - The classifier misclassifies marginal rain events
  - Strong performance on stable weather patterns
- Discuss data limitations impacting performance
  - Data consists of one location
  - Low temporal resolution
  - Missing important atmospheric indicators
  - Imbalanced target distribution

---

4. Discussion

4.1 Limitations

- Dataset imbalance
- Few extreme weather events = weak generalisation
- Lack of spatial diversity
- Missing features (wind direction, pressure, etc.)

4.2 Ethical Considerations

- **Bias**: Dataset location bias = inaccurate forecasting elsewhere
- **Societal impact**: Incorrect disaster predictions may affect public safety
- **Model transparency**: LSTM is not easily interpretable
- **Fairness**: If used in real-world emergency systems, unequal error rates matter

4.3 Future Work

- Add more locations
- Use attention-based models (Transformers)
- Compare classical ML models (XGBoost, ARIMA, etc.)
- Deploy as a forecasting tool for real-time weather stations.

---

5. Conclusion

In this project, we aimed to forecast using LSTM models for both regression and classification tasks. The classifier performed well in predicting whether it would rain, while the regressor struggled with accurately estimating rainfall amounts due to data limitations and the complexity of precipitation patterns. Overall, the results show that LSTMs can capture broad weather trends, but more detailed and diverse data are needed for precise rainfall prediction.