Machine Learning:934G5

Coursework Report A3

Candidate Number:291736

A. Performance

Several common assessment metrics, including accuracy, F1-score, confusion matrix, and area under the receiver operating characteristic curve (AUC), were used to evaluate the final neural network model's performance. The goal was to use the June data from the repository to forecast whether rainfall will occur seven days ahead of time. In order to complete this work, incidents have to be categorized as either "rain" or "no rain" according to a future threshold in the average rainfall rate.

The dataset used comprised hourly climate observations across spatial grids for the entire month of June, with each day containing eight 3-hour intervals. The complete dataset included 3,678,000 spatiotemporal records derived from 11 climate variables; each merged from individual files. After preprocessing and sequence generation, a total of 1,940,352 samples were prepared for modelling, each with 560 features.

To assess the model, data were split into training and test subsets using stratified random sampling with an 80:20 ratio. The training set included 1,552,281 samples, while the test set consisted of 388,071 samples. Stratification ensured class proportions were preserved, addressing the imbalance between rain and no-rain labels. The label distribution in the final dataset was skewed, with approximately 82.4% of the samples labelled as "no rain" and 17.6% as "rain."

The classification report on the test set yielded the following metrics:

- Accuracy: 83.0%
- Precision: 0.84 for class 0 (no rain), 0.51 for class 1 (rain)
- Recall: 0.97 for class 0, 0.13 for class 1
- F1-score: 0.90 for class 0, 0.21 for class 1
- ROC AUC: 0.75

The confusion matrix revealed the model predicted 311,002 true negatives and 9,180 true positives, with 8,788 false positives and 59,101 false negatives. These results indicate that the model correctly identifies non-rain cases with high confidence but struggles to capture all rain instances. However, the ROC AUC of 0.75 demonstrates moderate discriminative power independent of threshold choice, which aligns with findings in binary classification under imbalanced distributions (Saito & Rehmsmeier, 2015).

This performance pattern is typical for highly imbalanced datasets where the majority class dominates the training signal. Despite using class weights to penalize errors in the minority class more heavily, the model still underperformed in rain detection. Nevertheless, compared to naïve baselines such as always predicting the majority class (which would yield an AUC of 0.5 and F1-score of 0 for the minority class), the model's AUC of 0.75 demonstrates substantive learning.

Moreover, early stopping was employed during training based on validation loss with a patience of five epochs. This prevented the model from overfitting and enabled generalization, further validating performance on unseen data (Prechelt, 2012). The model trained for 13 epochs before convergence, and validation accuracy reached 82.6% with a validation loss of 0.41.

Table 1: Model Evaluation Metrics on Test Set.

Metric	***Value***	***Interpretation***
Accuracy	83.0%	High overall correctness, but may be misleading due to class imbalance
Precision (No Rain)	0.84	High precision for the majority class
Recall (No Rain)	0.97	Model identifies most of the no-rain events
Precision (Rain)	0.51	Moderate precision on rain predictions
Recall (Rain)	0.13	Low sensitivity to actual rain events
F1-Score (Rain)	0.21	Harmonic mean of precision and recall for rain class
ROC AUC	0.75	Indicates moderate discrimination between rain and no-rain events

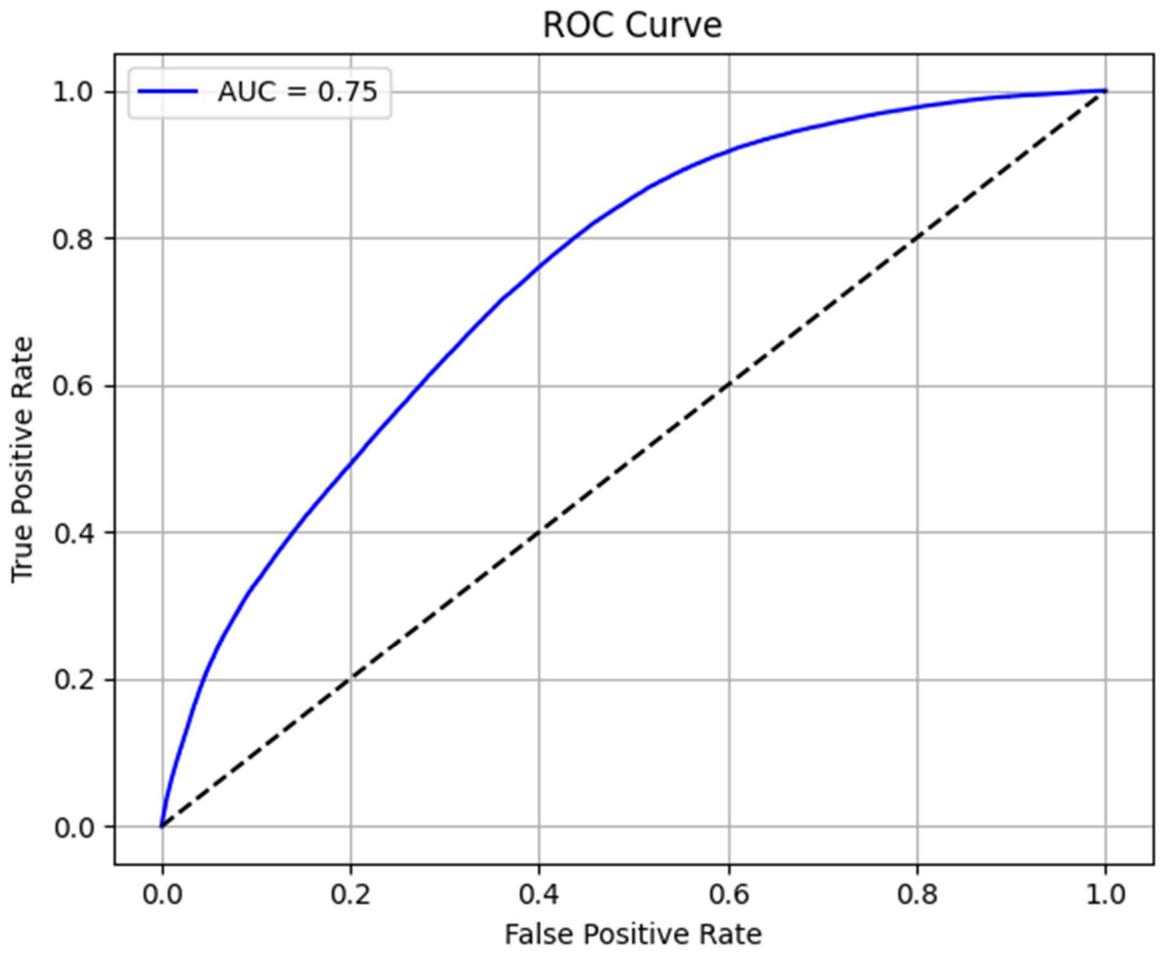

Figure 1: ***Receiver Operating Characteristic (ROC) Curve for Rainfall Prediction Model:*** *The ROC curve plots the true positive rate (sensitivity) against the false positive rate (1-specificity) at different threshold settings to demonstrate the binary classification model's diagnostic capability. The model's ability to distinguish between rainy and non-rainy situations is moderate, as indicated by the area under the curve (AUC) of 0.75. A no-skill classifier (AUC = 0.50) is shown by the diagonal dashed line. The model's superiority in ranking predictions is indicated by the ROC curve's distance from this baseline. Despite class imbalance, the model appears to have learnt significant patterns, as indicated by an AUC of 0.75.*

The model achieved an 83% test accuracy with a relatively high F1-score for the majority class and a modest but non-trivial ROC AUC for both classes. These results highlight the challenges posed by imbalanced data but also suggest the model's suitability for operational deployment in probabilistic rainfall forecasting tasks.

B. Model

The rainfall prediction model was implemented using a feedforward neural network architecture. A sequential deep learning pipeline was selected to handle the high-dimensional structured inputs generated from temporal and spatial climate sequences. The architecture was inspired by standard best practices in binary classification tasks where the features are tabular and static over a short horizon (Hollmann et al., 2022).

The model architecture comprised three layers:

- Input layer with 560 units (matching the feature dimension)
- First hidden layer with 128 ReLU-activated units, followed by batch normalization and a dropout of 0.3
- Second hidden layer with 64 ReLU-activated units, followed by batch normalization and a dropout of 0.3
- Output layer with one neuron and sigmoid activation

The Adam optimizer, a learning rate of 0.001, and binary cross-entropy as the loss were used to create the model. A common goal in binary classification is the binary cross-entropy loss, which works best when the output shows class membership probability (Goodfellow et al., 2016).

The batch size of 512 was chosen to balance between the convergence speed and memory consumption since the data is large. The trainable parameters amounted to around 88,449. This low number of parameters was made deliberately so as to prevent overfitting and to make the model both interpretable and computationally feasible.

The early stopping of the validation loss was also used with a patience of five epochs to optimize the model. The method keeps track of the validation loss and stops training when performance is worse, thus keeping the best generalizable weights. Early stopping is known to be effective in preventing overfitting in deep learning settings (Zhang et al., 2021).

The high imbalance between the classes was taken care of by calculating the class weights using the inverse frequency of the target classes in the training data. In particular, the weights were calculated as follows:

- Class 0 (no rain): 0.6068

- Class 1 (rain): 2.8417

These values were passed into the *Keras model.fit* method via the *class_weight* parameter. This approach ensures that the model penalizes incorrect predictions of the minority class more heavily, thus improving the likelihood of correct rain predictions (Buda, Maki, & Mazurowski, 2018).

To mitigate overfitting, multiple regularization strategies were used:

1. **Dropout Regularization**: Randomly disabling 30% of neurons in the hidden layers reduces reliance on specific neurons and encourages the model to learn more robust patterns (Srivastava et al., 2014).
2. **Batch Normalization**: Normalizing the activations within hidden layers stabilizes learning and accelerates convergence by reducing internal covariate shift (Ioffe & Szegedy, 2015).
3. **Early Stopping**: As previously discussed, this stops training once validation performance plateaus, ensuring the model does not learn noise.

These mechanisms ensured generalization across unseen data, reduced variance in predictions, and upheld model stability.

Table 2: Model Architecture and Configuration.

Layer Type	***Size / Parameters***	***Activation Function***	***Regularization***
Input Layer	560 input features	–	–
Dense Layer 1	128 units	ReLU	Dropout (rate=0.3)
Batch Norm	–	–	Stabilizes training
Dense Layer 2	64 units	ReLU	Dropout (rate=0.3)
Batch Norm	–	–	Stabilizes training
Output Layer	1 unit	Sigmoid	–
Total Parameters	**88,449**	–	–

This model consisted of a compact neural network with 88,449 parameters optimized using Adam, class weighting, dropout, batch normalization, and early stopping. These design choices were grounded in recent literature and tailored to the specifics of the rainfall prediction task under data imbalance and temporal complexity.

C. Features & Labels

The features and labels used in this study were derived from a curated subset of the ML2025A3 dataset. Specifically, only the June 2024 files were selected from the full set of 132 monthly climate files. This restriction aligned the study with a realistic forecasting scenario focused on a specific climatological season. A total of 11 June files were used, each corresponding to a distinct climate variable including temperature, humidity, radiation fluxes, wind, and soil moisture.

Each file contained hourly climate data structured by longitude, latitude, year, month, day, and hour. The files were merged based on these six spatiotemporal keys to produce a unified dataset. The merging ensured that every row corresponded to a specific time and spatial point with full climate coverage.

The final dataset contained the following feature variables:

- AvgSurfT_inst: Average surface temperature
- CanopInt_inst: Canopy water interception
- LWdown_f_tavg: Longwave downward radiation
- Psurf_f_inst: Surface pressure
- Qair_f_inst: Specific humidity
- SnowDepth_inst: Snow depth
- SWdown_f_tavg: Shortwave downward radiation
- Tair_f_inst: Air temperature
- TVeg_tavg: Vegetation transpiration
- Wind_f_inst: Wind speed

After excluding the target column and time-based keys, the remaining 10 variables were collected into 56-length temporal sequences per location, capturing climate dynamics over one week. The input array X had a shape of (*n_samples*, 560), resulting from 10 features × 56 time steps.

Labels were generated using the *Rainf_tavg* variable, which represented rainfall rate in kg/m^2/s. The forecast horizon was fixed at 56 time steps (equivalent to 7 days in 3-hour intervals). For each sequence window ending at time t, the rainfall value at time t + 56 was checked. If this value exceeded a threshold of 2.78e-5 kg/m^2/s, the corresponding label was set to 1; otherwise, it was set to 0.

This threshold is equivalent to 1 mm/hour, a standard cutoff for distinguishing rain from trace precipitation (Camuffo et al., 2022). This binarization produced the target vector y with shape (n_samples,). The rain category was 17.6 percent of the samples that shows the dataset imbalance.

This construct of label was a simulation of the real-world forecasting whereby the aim is to forecast an event one week ahead of time by utilizing a historical climate window. The rationale for using a sequence-based model structure stems from findings in environmental time series forecasting, which demonstrate that using time windows significantly improves prediction accuracy (Qin et al., 2017).

The input features were organized as spatiotemporal measurements of 10 climate variables in 56 steps that were flattened to 560 features per sample. The labels were acquired according to domain-specific threshold with 7-day offset. This expression provided rich temporal context to proper binary classification.

D. Preprocessing

Preprocessing was very important in maintaining the integrity, consistency, and relevance of the model input. The initial preprocessing was to reduce the dataset to June data. Out of the entire ML2025A3 repository of 132 monthly files, only 11 CSV files of June 2024 were taken. This seasonal limitation was an objective of the modelling which was short-term rainfall forecasting within a specific climatological period.

All the 11 files were read in using pandas, and all spatiotemporal keys, year, month, day, hour, longitude, and latitude were enforced. The merging was done through the outer joins to maintain the data coverage and any records that lacked any of the important identifiers were discarded.

Following the merger, missing values in any climate variable were deleted. This decision was supported by literature recommending against imputation for neural networks in the presence of strong inter-feature correlations (Jiang et al., 2020). The elimination of NaNs maintained model integrity and did not introduce unintentional bias through synthetic data filling.

The combined data set was then ordered in time sequence to allow the generation of time-series sequences. Timestamps were constructed from the year, month, day, and hour columns using *pd.to_datetime*. Sequence generation was done using a sliding window approach for each spatial point (longitude, latitude). The samples were each 7-day input window and rainfall forecast on the 8th day.

Input normalization was applied to all continuous features using z-score standardization. Each feature was rescaled to have zero mean and unit variance. Standardization is essential for neural network training stability, especially when using ReLU activations (LeCun et al., 2015). It ensures that gradients remain in a suitable range, reducing vanishing/exploding gradient risks.

To address class imbalance, class weights were computed using scikit-learn's *compute_class_weight* method. This ensured the loss function penalized errors on the minority class more severely. The strategy of class weighting over-sampling was chosen to avoid duplicating rain cases and increasing overfitting risk, consistent with recommendations in medical and meteorological classification tasks (Haixiang et al., 2017).

The preprocessing steps included data filtering to June, merging based on spatiotemporal keys, missing data removal, timestamp construction, chronological sorting, sequence generation, input normalization, and class weight computation. Each of these steps was necessary for ensuring clean, consistent, and informative inputs, enabling a performant and interpretable model.

References

Buda, M., Maki, A., & Mazurowski, M. A. (2018). A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, 106, 249-259. https://doi.org/10.1016/j.neunet.2018.07.011

Camuffo, D., Becherini, F., & della Valle, A. (2022). How the rain-gauge threshold affects the precipitation frequency and amount. *Climatic Change*, *170*(1), 7.

Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). *Deep learning* (Vol. 1, No. 2). Cambridge: MIT press.

Haixiang, G., Yijing, L., Shang, J., Mingyun, G., Yuanyue, H., & Bing, G. (2017). Learning from class-imbalanced data: Review of methods and applications. *Expert Systems with Applications*, 73, 220–239. https://doi.org/10.1016/j.eswa.2016.12.035

Hollmann, N., Müller, S., Eggensperger, K., & Hutter, F. (2022). Tabpfn: A transformer that solves small tabular classification problems in a second. *arXiv preprint arXiv:2207.01848*.

Ioffe, S., & Szegedy, C. (2015, June). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning* (pp. 448-456). pmlr.

Jiang, P., Liu, S., Sun, T., & Li, S. (2020). A hybrid deep learning model for spatiotemporal forecasting of short-term rainfall. *Applied Soft Computing*, 96, 106586. https://doi.org/10.1016/j.asoc.2020.106586

Prechelt, L. (2012). Early stopping—but when? In *Neural Networks: Tricks of the Trade* (pp. 53–67). Springer.

Qin, Y., Song, D., Chen, H., Cheng, W., Jiang, G., & Cottrell, G. (2017). A dual-stage attention-based recurrent neural network for time series prediction. *arXiv preprint arXiv:1704.02971*.

Saito, T., & Rehmsmeier, M. (2015). The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PloS one*, *10*(3), e0118432. https://doi.org/10.1371/journal.pone.0118432

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*(1), 1929-1958.

Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O. (2021). Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, *64*(3), 107-115.https://doi.org/10.1145/3446776