# Next-Day Rain Forecasting Using Deep Learning Techniques

*A Case Study report submitted in partial fulfillment of the requirements for the award of the degree of*

**Integrated M.Sc. in Computer Science**
*with Specialization in*
Artificial Intelligence and Machine Learning


Submitted by

Abhinav Raj
(NA21PICS04)




**Nehru Arts and Science College Kanhangad**
**Padnekkad P.O., Kasaragod Dt., Kerala - 671314**


***Affiliated to***

**Kannur University**
**Kannur**

**March 2025**

# Nehru Arts and Science College Kanhangad

Padnekkad P.O., Kasaragod Dt., Kerala - 671314


## CERTIFICATE

This is to certify that report entitled "**Case Study: Forecasting Rainfall in Australia Using Neural Networks** " is a bonafide report of the Case Study (8B38ICSC: Lab 11 - Case Study (Data Mining)) presented during VIII$^{th}$ semester by **Abhinav Raj** with Register No. **NA21PICS04** in partial fulfillment of the requirements for the award of the degree of Integrated M.Sc. in Computer Science with Specialization in Artificial Intelligence and Machine Learning.

Faculty In Charge

Head of the Department

External Examiner

Internal Examiner

## DECLARATION

I, **Abhinav Raj**, VIII Semester Integrated M.Sc. in Computer Science with Specialization in Artificial Intelligence and Machine Learning Student of Nehru Arts and Science College Kanhangad under Kannur University do hereby declare that the case study entitled "**Case Study: Forecasting Rainfall in Australia Using Neural Networks** " is original work carried out by me towards the partial fulfillment of the requirement of Integrated M.Sc. in Computer Science with Specialization in Artificial Intelligence and Machine Learning, and no part thereof has been presented for the award of any other degree.

**Abhinav Raj**

**ACKNOWLEDGMENT**

I would like to express my sincere gratitude to all those who have supported and encouraged me throughout the course of this project. Special thanks to my esteemed lecturers, Miss Ramya N and Mr Mithun A.V., for their invaluable guidance and mentorship, which has been instrumental in shaping this work. I am also thankful to my family and friends for their unwavering support and encouragement.

Abhinav Raj

## ABSTRACT

Weather forecasting plays a vital role in planning and decision-making across various sectors, including agriculture, transportation, and disaster management. This case study explores the application of a neural network-based classification model to predict the likelihood of rainfall on the following day in locations across Australia. Utilizing a dataset spanning over 10 years of daily weather observations, the study involves extensive data preprocessing to handle missing values, encode categorical features, and normalize continuous variables. The primary objective is to predict the binary outcome of "RainTomorrow" using key meteorological features such as temperature, humidity, wind speed, atmospheric pressure, and precipitation. A feedforward neural network is implemented and trained on the processed dataset. Model performance is evaluated using standard classification metrics including accuracy, precision, recall, and F1-score. The results demonstrate that neural networks can effectively capture complex patterns in weather data and offer promising accuracy for short-term rain prediction. This study highlights the potential of deep learning methods in enhancing the accuracy of weather forecasting systems.

# Contents

# Chapter 1

# Introduction

## 1.1 Background Information

Weather forecasting is an essential component of modern societal infrastructure, impacting a broad range of sectors including agriculture, transportation, energy management, public safety, and disaster response. Among the many types of forecasts, rainfall prediction plays a particularly significant role in regions susceptible to droughts, floods, and varying climatic patterns—such as Australia. As a continent characterized by geographical diversity and climate extremes, Australia poses unique challenges and opportunities for the development of advanced meteorological prediction models.

Traditionally, rainfall prediction has relied heavily on statistical models and physical simulations based on atmospheric equations. While these methods have shown effectiveness over the years, they often struggle with the high variability and non-linear interactions inherent in weather data. Furthermore, these conventional approaches can be computationally intensive and sensitive to missing or imprecise data inputs.

With the emergence of machine learning (ML) and deep learning (DL), data-driven approaches have gained popularity due to their flexibility, scalability, and ability to model complex, non-linear relationships in large datasets. Artificial Neural Networks (ANNs), in particular, have demonstrated promising results in time series forecasting, classification problems, and pattern recognition tasks, making them a suitable candidate for rainfall prediction.

This case study uses the Rain in Australia dataset, a comprehensive collection of over 145,000 weather records gathered over a span of ten years from numerous locations across the country. Each record includes a wide range of meteorological features such as daily maximum and minimum temperatures, humidity, wind direction and speed, atmospheric pressure, cloud cover, sunshine duration, and rainfall measurements. The dataset's target feature, RainTomorrow, is a binary variable indicating whether it rained the next day at a given location.

The goal of this study is to predict the RainTomorrow variable based on the preceding day's weather data. This classification task is approached using a feed-forward artificial neural network, which is trained on the cleaned and preprocessed dataset. The preprocessing stage involves several key steps including handling missing values, encoding categorical variables, normalizing numerical data, and addressing class imbalance if present.

The use of a neural network is justified by its ability to approximate complex functions and learn from data patterns without explicit programming. By training the network on historical weather patterns, it is expected to capture the latent relationships that govern rainfall events.

Through this study, I aim to:

1. Evaluate the predictive performance of a neural network model for rainfall forecasting.
2. Compare its effectiveness to traditional baseline models (e.g., logistic regression or decision trees).
3. Explore the impact of data quality and feature selection on model accuracy.
4. Demonstrate how deep learning can enhance the reliability of short-term weather predictions, especially in data-rich environments like meteorological forecasting.

The results of this case study not only contribute to the ongoing research in AI-based weather forecasting but also hold practical implications for developing smarter, more responsive systems in climate-sensitive sectors.

## 1.2 Problem Statement

Accurate rainfall prediction is a challenging task due to the inherently dynamic and non-linear nature of weather systems. In regions like Australia, where climatic conditions vary widely across geographic locations, timely and reliable rain forecasts are crucial for planning in sectors such as agriculture, water resource management, transportation, and disaster preparedness.

Despite the availability of large volumes of meteorological data, traditional statistical forecasting techniques often fall short in capturing the complex patterns and dependencies that influence rainfall. These limitations can lead to inaccurate predictions, which may result in economic losses or safety risks.

The specific problem addressed in this case study is:

> **To develop a predictive model that can accurately forecast whether it will rain on the following day at a given location in Australia, using historical weather data and a neural network-based classification approach.**

This involves:

- Cleaning and preprocessing a large real-world weather dataset,
- Selecting and engineering relevant features,
- Applying a neural network to learn underlying patterns,
- And evaluating the model's performance in predicting the target variable: `RainTomorrow`.

The ultimate goal is to demonstrate the effectiveness of neural networks in short-term rainfall prediction and highlight their potential in enhancing decision-making in climate-sensitive applications.

## 1.3 Objective

The primary objective of this case study is to develop a predictive model capable of accurately forecasting whether it will rain on the following day in various locations across Australia. Leveraging a historical weather dataset and deep learning techniques, specifically a feedforward neural network, this study seeks to explore the efficacy of artificial neural networks in handling real-world meteorological data.

The specific objectives are as follows:

- To understand and preprocess the Rain in Australia dataset by addressing missing values, encoding categorical features, and normalizing continuous variables.
- To design and implement a neural network architecture suitable for binary classification tasks involving meteorological inputs.
- To evaluate the model's performance using standard classification metrics such as accuracy, precision, recall, F1-score, and the Area Under the Receiver Operating Characteristic Curve (AUC).
- To compare the performance of the neural network with traditional baseline models where applicable.
- To analyze the impact of feature selection and data quality on the prediction accuracy of rainfall.
- To demonstrate the potential of neural networks as an effective approach for short-term rainfall forecasting in climate-sensitive regions.

# Chapter 2

# Literature Review

## 2.1 Introduction to Rainfall Forecasting

Rainfall forecasting plays a vital role in various sectors such as agriculture, hydrology, disaster management, and urban planning. Accurate predictions help mitigate the impact of extreme weather events, optimize irrigation schedules, and support early warning systems for floods and droughts. Traditional rainfall forecasting methods are based on numerical weather prediction (NWP) models that solve complex physical equations to simulate atmospheric processes. However, these models often require high computational resources and are sensitive to initial conditions and observational errors.

In contrast, machine learning (ML) and deep learning (DL) techniques have emerged as powerful alternatives due to their ability to learn directly from historical data without the need for explicit physical modeling. Artificial Neural Networks (ANNs), in particular, are known for their capacity to approximate nonlinear functions and detect hidden patterns, making them suitable for modeling the chaotic and dynamic nature of weather systems.

## 2.2 Historical Perspective

The field of data-driven rainfall prediction has evolved significantly over the past few decades. Early approaches used linear regression, autoregressive integrated moving average (ARIMA), and other time-series models. While useful for trend analysis, these models were limited in handling non-linear dependencies and multiple influencing variables.

The introduction of machine learning techniques such as Decision Trees, Support Vector Machines (SVM), and k-Nearest Neighbors (k-NN) marked a significant advancement in rainfall prediction. These models were better at capturing complex patterns, but still struggled with high-dimensional data and temporal dependencies.

In the last decade, deep learning has revolutionized predictive modeling in meteorology. Feedforward Neural Networks (FNN), Convolutional Neural Networks (CNN), and Long Short-Term Memory (LSTM) networks have shown significant improvements in accuracy and generalization when applied to weather datasets. These models can automatically learn feature hierarchies and temporal dependencies, which are crucial for modeling precipitation.

## 2.3 Key Variables in Rainfall Prediction

Effective rainfall forecasting relies on selecting relevant meteorological features that influence precipitation. Commonly used variables include:

- **Temperature (Max/Min)** – Affects evaporation and condensation processes.
- **Humidity** – Indicates the amount of moisture in the air, which is essential for cloud formation.
- **Wind Speed and Direction** – Helps determine moisture transport and pressure systems.
- **Atmospheric Pressure** – Low-pressure systems are often associated with precipitation.
- **Rainfall** – Historical rainfall data serves as a lagged variable for forecasting.
- **Cloud Cover and Sunshine Duration** – Influences surface heating and atmospheric convection.

Feature engineering techniques such as normalization, encoding of categorical variables, and dimensionality reduction are often employed to enhance model performance. Handling missing values is another critical step, as real-world meteorological data often contain gaps due to sensor malfunctions or reporting delays.

## 2.4 Regulatory and Operational Context

Meteorological forecasting is primarily handled by government agencies such as the Australian Bureau of Meteorology (BoM), which relies on a combination of satellite data, weather stations, and numerical models. While these operational models are robust, they often require substantial infrastructure and expert knowledge to operate and interpret.

In recent years, ML-based systems have been introduced as complementary tools for specific tasks like short-term forecasting or localized predictions. Although not yet fully adopted in regulatory settings, these models offer the potential for automation and cost-effective deployment, particularly in resource-constrained environments.

## 2.5 Advances in Neural Network Models

Among the various deep learning models, Feedforward Neural Networks (FNNs) have been widely used in classification tasks such as predicting binary outcomes (e.g., RainTomorrow). These networks consist of multiple layers of interconnected neurons that transform input features into predictions via learned weights and activation functions.

Other architectures like CNNs are used for spatial data (e.g., satellite imagery), while LSTM networks are preferred for time-series analysis due to their ability to remember long-term dependencies.

Several studies have demonstrated the efficacy of ANNs in rainfall forecasting. For example, French et al. (1992) developed one of the earliest ANN-based rainfall models. More recently, researchers have integrated ANNs with optimization algorithms (e.g., genetic algorithms, particle swarm optimization) to improve parameter tuning and forecasting accuracy.

## 2.6 Challenges in Rainfall Prediction

Despite the advances, rainfall prediction remains a complex task due to several challenges:

- **Data Quality** – Missing values and inconsistent records can hinder model training.
- **Imbalanced Classes** – Rain events are often less frequent than non-rain events, leading to biased models.
- **Overfitting** – Neural networks can memorize the training data if not properly regularized.
- **Generalization** – Models trained on specific geographic regions may not perform well elsewhere.
- **Interpretability** – Neural networks are often considered "black boxes," which makes them hard to interpret.

Addressing these issues requires robust preprocessing, careful model selection, and evaluation using diverse metrics.

## 2.7 Current Research and Innovations

Contemporary research in rainfall prediction using ML focuses on hybrid models, ensemble learning, and integration with remote sensing data. For example, some studies combine ANN with wavelet transforms to capture multi-scale patterns, while others use ensemble techniques like bagging and boosting to improve prediction robustness.

Evaluation metrics have also evolved. In addition to accuracy, precision, recall, and F1-score, researchers increasingly use the Area Under the Receiver Operating Characteristic Curve (AUC-ROC) to evaluate classification performance, especially in imbalanced datasets. AUC provides a measure of a model's ability to distinguish between classes, regardless of threshold.

Recent innovations also explore real-time deployment of trained models in web or mobile-based systems for public use. With growing concerns around climate change, such predictive tools are likely to play an increasingly important role in adaptive planning and resilience building.

# Chapter 3

# Data Collection

## 3.1 Data Sources

The dataset used in this study comprises approximately ten years of daily weather observations collected from various locations across Australia. These observations were recorded by multiple weather stations operated by the Australian Bureau of Meteorology.

The dataset includes 23 attributes, encompassing a wide range of meteorological variables such as temperature, humidity, wind speed, and rainfall measurements. The primary objective of this project is to utilize these features to predict the target variable, `RainTomorrow`, which indicates whether or not it will rain the following day.

The data was sourced from Kaggle, where it is publicly available under the title *"WeatherAUS"*. This dataset is widely used for benchmarking classification models in weather forecasting research.

- **Source:** Australian Bureau of Meteorology via Kaggle
- **Access URL:** Kaggle: Weather Dataset (WeatherAUS)
- **Data Range:** Approximately 10 years
- **Number of Attributes:** 23 (including the target variable)
- **Target Variable:** `RainTomorrow`

## 3.2 Data Description

The dataset used in this study encompasses nearly a decade of daily weather observations collected from numerous weather stations across Australia. It contains over 140,000 instances with 23 features, including the target variable `RainTomorrow`. These features represent a comprehensive set of meteorological variables that can significantly influence precipitation outcomes and are well-suited for a classification task.

**Target Variable:**

- **RainTomorrow** (categorical): This is the binary target variable I aim to predict. It takes the value `"Yes"` if 1mm or more of rain is recorded the next day, otherwise `"No"`. Accurate prediction of this variable serves practical purposes in agriculture, transportation, and daily planning.

**Temporal and Locational Attributes:**

- **Date** (datetime): The calendar date of the observation. It is used to extract temporal patterns such as month or seasonality, which may influence rainfall.
- **Location** (categorical): The name of the weather station where the observation was recorded. It helps in capturing geographical variations in climate.

**Temperature Features:**

- **MinTemp** (float): The minimum temperature of the day in degrees Celsius.
- **MaxTemp** (float): The maximum temperature of the day.
- **Temp9am**, **Temp3pm** (float): Temperature recorded at 9am and 3pm respectively. These features can be useful in identifying intra-day temperature trends.

**Precipitation and Sunshine:**

- **Rainfall** (float): Total rainfall recorded for the day in millimetres.
- **RainToday** (categorical): Indicates whether rain was recorded today. This is a strong feature that can be predictive of future rain.
- **Evaporation** (float): The amount of water evaporated during the day. Evaporation is indirectly influenced by humidity, temperature, and wind.
- **Sunshine** (float): Total hours of bright sunshine. Less sunshine is typically associated with cloudy and rainy weather.

**Wind Attributes:**

- **WindGustDir**, **WindDir9am**, **WindDir3pm** (categorical): Directions from which the wind is blowing at peak gust, 9am, and 3pm respectively.
- **WindGustSpeed**, **WindSpeed9am**, **WindSpeed3pm** (float): Speed of wind gusts and at specific times. Wind direction and speed are often early indicators of changing weather conditions.

**Humidity and Pressure:**

- **Humidity9am**, **Humidity3pm** (float): Relative humidity percentage at 9am and 3pm. High humidity levels are typically correlated with rainfall.
- **Pressure9am**, **Pressure3pm** (float): Atmospheric pressure at 9am and 3pm. Falling pressure is often an indication of incoming rain systems.

**Cloud Coverage:**

- **Cloud9am**, **Cloud3pm** (float): Fractional cloud cover measured in oktas (eighths) of the sky covered by clouds. Cloud cover data is directly associated with precipitation probability.

**Data Types and Considerations:**

The dataset contains both numerical and categorical features. Categorical features such as `Location`, `WindGustDir`, and `RainToday` require encoding techniques during preprocessing. Numerical features such as `MinTemp`, `Rainfall`, and `Humidity3pm` often benefit from normalization or scaling to improve model performance. Additionally, the dataset contains missing values which must be addressed through imputation or removal, depending on the extent and importance of the feature.

Together, these features form a robust and diverse dataset that is highly suitable for building machine learning models for rainfall forecasting. Their variability across time and space provides an excellent opportunity for model generalization and accurate weather prediction.

## 3.3 Data Pre-processing

Data pre-processing is a critical phase in the data science workflow, particularly in weather prediction tasks, where raw meteorological data often includes noise, missing values, and a mix of categorical and numerical features. The performance of machine learning models depends heavily on the quality and representation of this input data. In this project, I designed a comprehensive pre-processing pipeline to clean, balance, and transform the data for optimal model performance.

### 3.3.1 Uploading the Data

The dataset was loaded using the `pandas` library. I nitial exploratory analysis included inspecting column types, checking for null values, and identifying inconsistencies such as formatting errors or extreme outliers.

### 3.3.2 Target Variable Processing

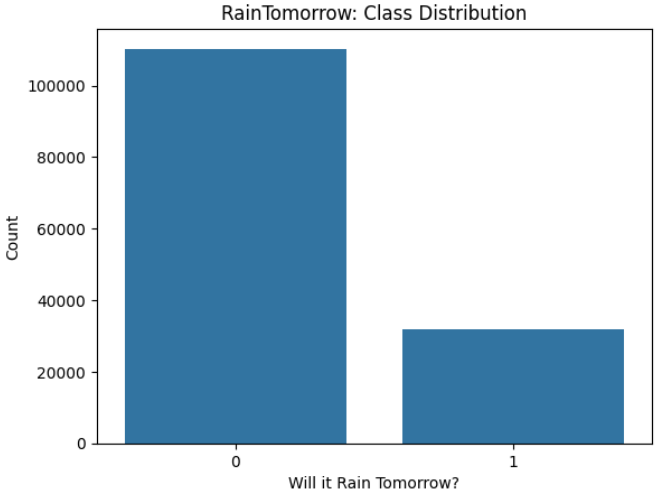
The target variable, `RainTomorrow`, included three types of entries: `Yes`, `No`, and missing values (NaN). Since imputing the target variable can introduce bias, all rows with missing values in `RainTomorrow` were removed. The remaining values were encoded as:

$$\texttt{RainTomorrow} \in \{0\ (\text{No}), 1\ (\text{Yes})\}$$

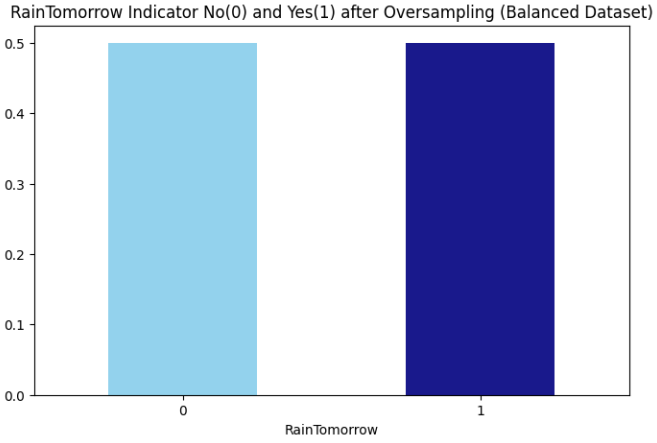
An initial class distribution analysis revealed an imbalance:

$$\texttt{No}: 77.6\%, \quad \texttt{Yes}: 22.4\%$$

To address this, Iapplied oversampling using bootstrapping: instances of the minority class (Yes) were sampled with replacement until class balance was achieved. This step ensured the model could learn patterns related to both classes more effectively.




(a) Before




(b) After

Figure 3.1: "RainTommorow" before and after

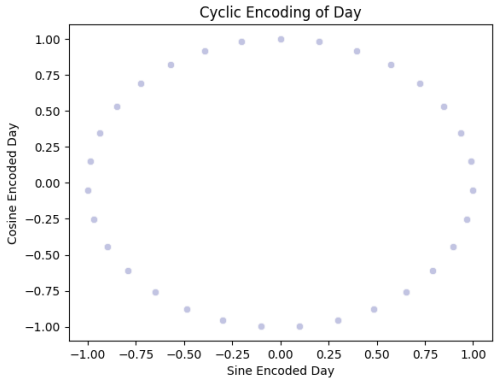
### 3.3.3 Parsing and Encoding the Date Feature

The `Date` column, originally a string in the format YYYY-MM-DD, was parsed into Python `datetime` objects and decomposed into three components: `day`, `month`, and `year`. The cyclic nature of the day and month was captured using sine and cosine transformations:

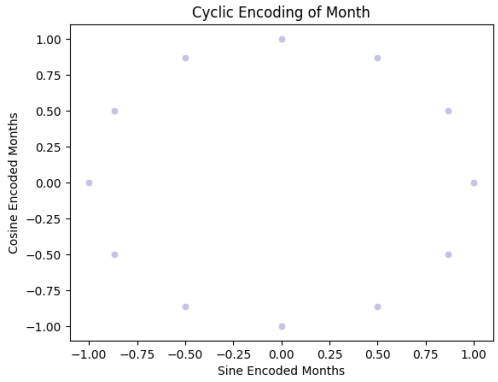
$$\text{month}_{\text{sin}} = \sin\left(2\pi \cdot \frac{\text{month}}{12}\right), \quad \text{month}_{\text{cos}} = \cos\left(2\pi \cdot \frac{\text{month}}{12}\right)$$

$$\text{day}_{\text{sin}} = \sin\left(2\pi \cdot \frac{\text{day}}{31}\right), \quad \text{day}_{\text{cos}} = \cos\left(2\pi \cdot \frac{\text{day}}{31}\right)$$

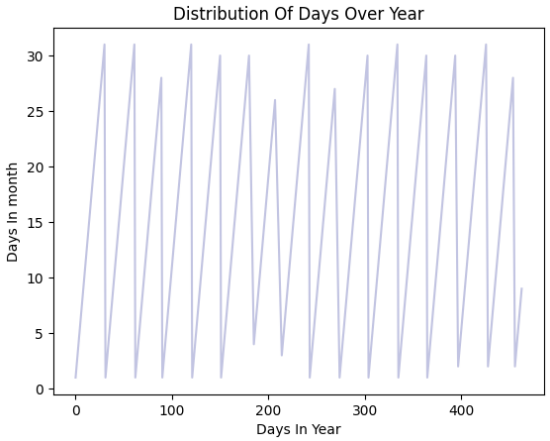
The `year` component was excluded to avoid the risk of temporal leakage and ensure the model generalized to unseen data without overfitting to specific years.




(a) Day




(b) Month




(c) Distribution of Day over Year

Figure 3.2: Cyclic encoded Day and Month

### 3.3.4 Handling Missing Values

Missing values in meteorological data are common due to equipment errors or data loss. I applied imputation based on feature type:

- **Numerical Features:** Imputed using the median, providing robustness against outliers.
- **Categorical Features:** Imputed using the mode, representing the most frequent category.

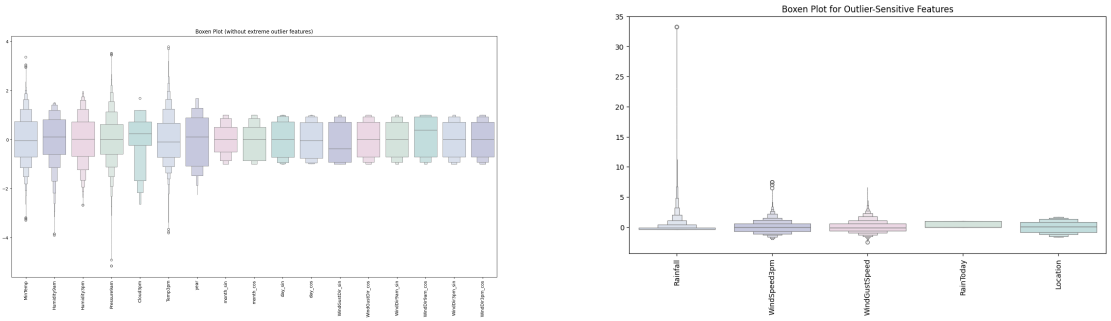
Rows with excessive missingness were removed to preserve data integrity.

### 3.3.5 Handling Outliers

Outliers, especially in continuous weather variables, can mislead neural network learning. The Interquartile Range (IQR) method was applied:

$$\text{Outlier Range} = [Q_1 - 1.5 \times IQR,\ Q_3 + 1.5 \times IQR]$$

Values falling outside this range were considered anomalous and removed. Boxplots were used to visually validate these decisions, particularly in variables like rainfall and wind speed.




(a) Before (b) After

Figure 3.3: Box plot with outliers

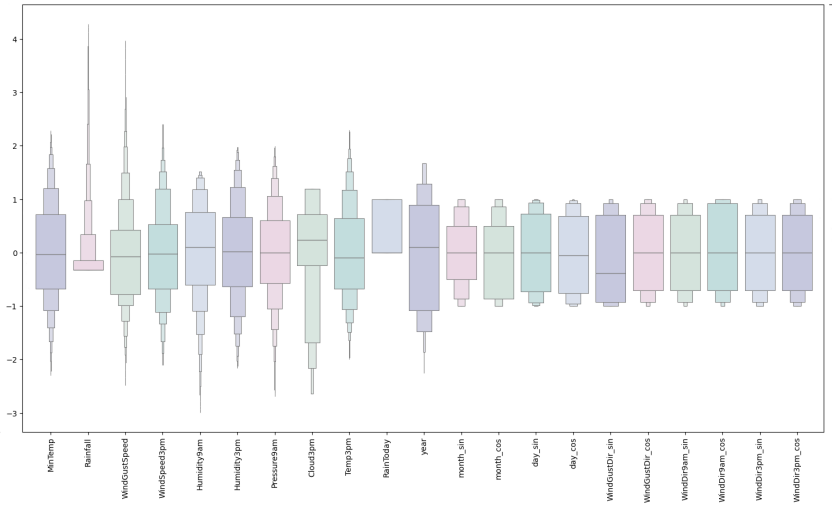



Figure 3.4: Box plot without outliers

### 3.3.6 Encoding Categorical Variables

Categorical variables were encoded to numerical formats suitable for machine learning models.

#### Location Feature

Two encoding approaches were explored:

- **One-Hot Encoding:** Each location was represented as a binary vector. The first category was dropped to avoid multicollinearity.
- **Label Encoding with Scaling:** Locations were encoded as integers and normalized using z-score scaling. This retained compactness in feature space while reducing ordinal bias.

**Binary and Directional Features**

- **RainToday and RainTomorrow:** Encoded as 1 for `Yes` and 0 for `No`.
- **WindGustDir**, **WindDir9am**, and **WindDir3pm**: These compass directions were mapped to degrees (e.g., N = 0°, E = 90°, etc.) and then transformed using sine and cosine:

$$\theta = \text{Direction Angle in Degrees}$$

$$\text{dir}_{\sin} = \sin\left(2\pi \cdot \frac{\theta}{360}\right), \quad \text{dir}_{\cos} = \cos\left(2\pi \cdot \frac{\theta}{360}\right)$$

### 3.3.7 Standardizing Numerical Features

All continuous numerical features were standardized using z-score normalization:

$$z = \frac{x - \mu}{\sigma}$$

This ensured uniform feature scaling, which is crucial for gradient-based optimizers in neural networks.

### 3.3.8 Analyzing Correlations

A correlation heatmap was constructed to identify strongly correlated features. Although a few pairs showed high correlation, I retained all features, leveraging the neural network's capability to learn relevant representations while avoiding premature feature elimination.

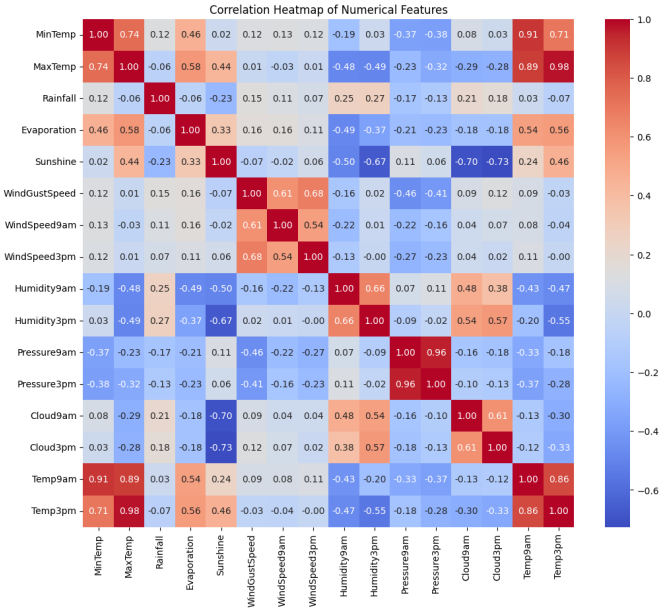



Figure 3.5: correlation matrix for numerical features

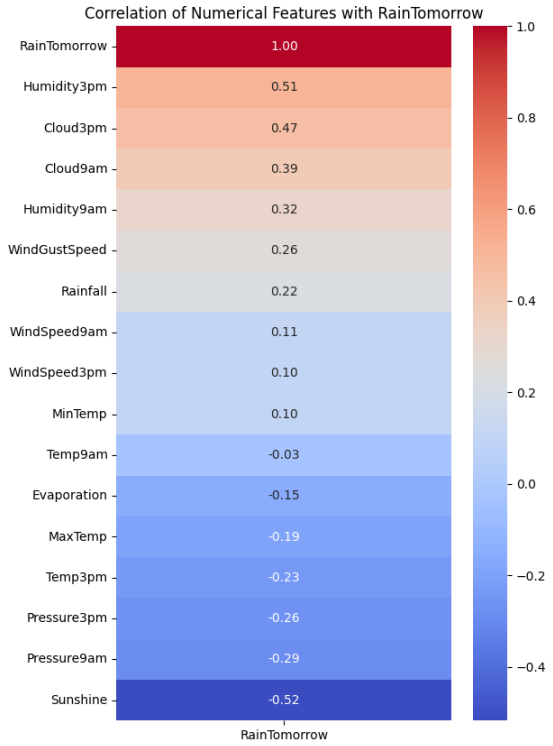



Figure 3.6: correlation with 'RainTomorrow'

### 3.3.9 Final Dataset Preparation

Two versions of the processed dataset were created based on the encoding method for the `Location` feature:

- **Label-Encoded Dataset:** With the label-encoded `Location` (z-score normalized), resulting in 22 input features per sample.
- **One-Hot Encoded Dataset:** With a binary vector for each unique `Location` (first dummy column dropped), leading to 69 input features.

Common to both versions:

- The original `Date`, `month`, and `day` columns were dropped after transformation.
- The target variable was encoded as 0 (No) and 1 (Yes).
- The dataset was split into training and testing sets using an 80:20 ratio with stratified sampling to maintain class balance.

This rigorous pre-processing pipeline ensured the dataset was clean, balanced, and feature-engineered for effective model learning and robust performance.

# Chapter 4

# Methodology

## 4.1 Data Mining Techniques

Data mining encompasses the extraction of meaningful patterns and knowledge from large datasets using statistical, machine learning, and computational techniques. For this project, a classification-based data mining approach was adopted to predict the binary outcome of rainfall occurrence the next day (`RainTomorrow`).

The key data mining techniques utilized include:

- **Data Cleaning and Pre-processing:** Noise and missing values in meteorological data were handled through median/mode imputation, cyclic encoding for temporal and directional features, and outlier removal using the Interquartile Range (IQR) method.
- **Feature Engineering:** Cyclic transformations (sine and cosine) for temporal and wind direction features, one-hot and label encoding for categorical variables, and z-score normalization for continuous features ensured all input data were machine learning-ready.
- **Classification:** A feedforward Artificial Neural Network (ANN) was designed for binary classification, distinguishing between rain and no-rain scenarios for the next day.
- **Model Evaluation:** Model performance was assessed using metrics such as accuracy and AUC (Area Under the ROC Curve) to ensure both overall correctness and robustness of classification.

## 4.2 Tools and Software

The implementation was carried out using Python and several open-source libraries well-suited for data science and machine learning tasks. The key tools and software used are:

- **Python 3.x:** The primary programming language for model development.
- **NumPy and Pandas:** Used for data manipulation, preprocessing, and feature transformation.
- **Matplotlib and Seaborn:** Employed for data visualization and exploratory analysis.

- **Scikit-learn:** Provided preprocessing utilities (e.g., label encoding, train-test splitting) and evaluation metrics.
- **TensorFlow and Keras:** Used for building, training, and evaluating the neural network model.
- **Jupyter Notebook/Colab:** Served as the interactive development environment for writing and testing the code.

## 4.3 Implementation

The implementation phase consisted of building and training a feed-forward Artificial Neural Network (ANN) using the Keras API. The model architecture was tailored to each dataset variant based on the input feature dimension.

### 4.3.1 Model Architecture

The ANN consisted of multiple dense layers with ReLU activation functions and batch normalization to improve convergence. A final dense output layer with a sigmoid activation was used to generate probability predictions for the binary classification task.

- **Input Layer:** Accepts 22 features for the label-encoded dataset and 69 for the one-hot encoded version.
- **Hidden Layers:**
  - Dense(64, activation='relu')
  - BatchNormalization
  - Dense(32, activation='relu')
  - BatchNormalization
  - Dense(16, activation='relu')
  - BatchNormalization + Dropout(0.1)
- **Output Layer:** Dense(1, activation='sigmoid') for binary output.

### 4.3.2 Training Procedure

- **Optimizer:** Adam with a learning rate of 0.00009
- **Loss Function:** Binary Crossentropy
- **Evaluation Metrics:** Accuracy and Area Under the Curve (AUC)
- **Early Stopping:** Implemented to halt training when validation performance stopped improving, with a patience of 20 epochs and minimum delta of 0.001.
- **Validation Split:** 20% of the training data used for validation during training.

### 4.3.3 Model Summary and Training

The model summary displayed the architecture and parameter counts. Training was conducted for a maximum of 150 epochs with a batch size of 32, although early stopping typically terminated training earlier to prevent overfitting.

Both versions of the model — using label-encoded and one-hot encoded features — were trained and evaluated. The next chapter presents the performance analysis and comparison of these models.

The model architecture was compiled and trained using the following Python code:

```
early_stopping = callbacks.EarlyStopping(
    min_delta=0.001,
    patience=20,
    restore_best_weights=True
)

model = Sequential()
model.add(Dense(64, kernel_initializer='he_normal', activation='relu',
    input_dim=69))
model.add(BatchNormalization())

model.add(Dense(32, kernel_initializer='he_normal', activation='relu'))
model.add(BatchNormalization())

model.add(Dense(16, kernel_initializer='he_normal', activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.1))

model.add(Dense(1, kernel_initializer='glorot_uniform', activation='sigmoid')
    )

opt = Adam(learning_rate=0.00009)
model.compile(optimizer=opt, loss='binary_crossentropy', metrics=['accuracy',
     AUC(name='auc')])

model.summary()

history = model.fit(X_train, y_train, batch_size=32, epochs=150,
                    callbacks=[early_stopping], validation_split=0.2)
```

The training history, including accuracy and AUC curves, was visualized to assess convergence and detect any signs of overfitting or underfitting.

# Chapter 5

# Analysis and Results

## 5.1 Data Analysis

The dataset used in this project was sourced from historical weather records collected across various locations in Australia. The primary task was to predict the binary outcome variable `RainTomorrow`, indicating whether it would rain the following day. Effective data preprocessing and feature engineering were vital to preparing the dataset for modeling.

### Exploratory Data Analysis

A comprehensive exploratory data analysis (EDA) was conducted to identify missing values, understand data distributions, and detect any anomalies or outliers. The following were the key observations and steps taken during this phase:

- **Missing Values:** Features such as `Evaporation`, `Sunshine`, and `Cloud9am/3pm` had missing values. These were imputed using appropriate techniques — median imputation for numerical features and mode imputation for categorical features — to avoid bias and preserve data consistency.
- **Date Parsing and Feature Extraction:** The `Date` column was parsed to extract useful components such as `Year`, `Month`, and `Day`. This enabled temporal analysis and allowed encoding of seasonal trends.
- **Cyclical Encoding:** Since features like `Month` and wind directions (`WindDir9am`, `WindDir3pm`) are cyclic in nature, sine and cosine transformations were used to represent their periodic behavior.
- **Categorical Encoding:** Categorical variables, particularly `Location`, were encoded using both Label Encoding and One-Hot Encoding for separate experiments to compare their impact on model performance.
- **Outlier Removal:** Outliers were detected and removed using the Interquartile Range (IQR) method. This helped improve model generalization and stability during training.
- **Normalization:** Z-score normalization was applied to all continuous numerical features to ensure uniform scaling. This is especially crucial for neural networks, which are sensitive to feature magnitudes.

- **Class Distribution:** The target variable was significantly imbalanced, with many more instances of 'No Rain' compared to 'Rain'. This imbalance was addressed in the training pipeline using balanced class weights to ensure the model learned to predict minority cases more accurately.

## 5.2 Results

The performance of the neural network model was evaluated across three main scenarios:

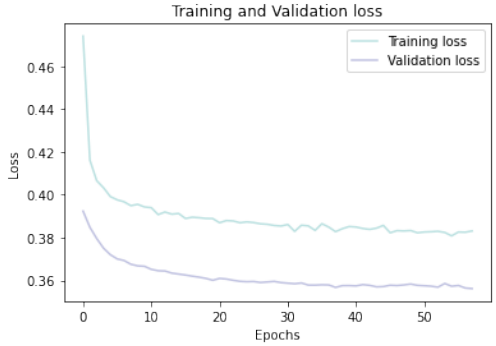
1. Before addressing class imbalance.
2. After handling class imbalance using Label Encoding for the `Location` feature.
3. After handling class imbalance using One-Hot Encoding for the `Location` feature.

In each case, the model was trained with early stopping, learning rate scheduling, and model checkpointing. The results were evaluated using classification metrics such as accuracy, precision, recall, and F1-score.

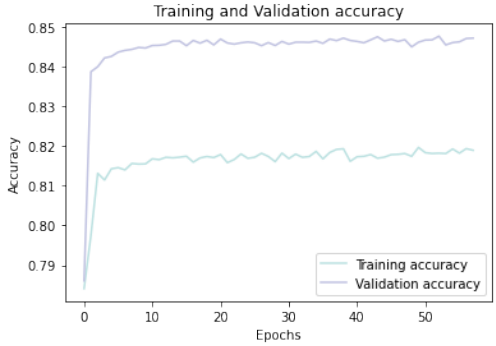
### 5.2.1 Before Addressing Class Imbalance

| Class | Precision | Recall | F1-score | Support |
|---|---|---|---|---|
| 0 (No Rain) | 0.88 | 0.94 | 0.91 | 20110 |
| 1 (Rain) | 0.71 | 0.50 | 0.58 | 5398 |
| **Accuracy** | 0.85 | | | 25508 |
| **Macro Avg** | 0.79 | 0.72 | 0.75 | 25508 |
| **Weighted Avg** | 0.84 | 0.85 | 0.84 | 25508 |

Table 5.1: Model performance before addressing class imbalance




(a) Training and Validation loss



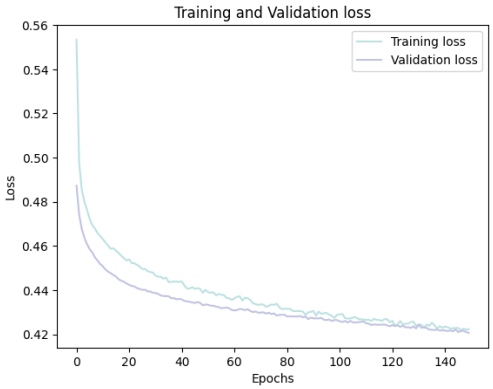

(b) Training and Validation Accuracy

**Observation:** The model performed well for the majority class (No Rain), but the recall for the minority class (Rain) was very low (0.50). This indicates that the model was biased toward predicting the dominant class, leading to poor detection of actual rain days — a critical drawback for a real-world rainfall forecasting system.

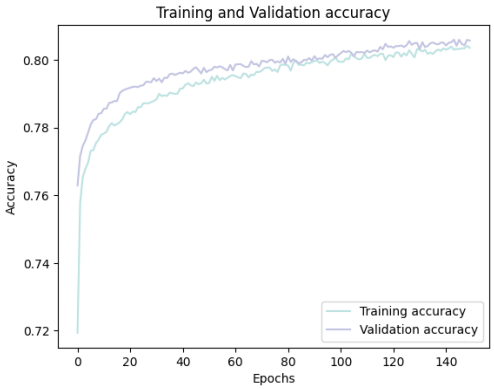
### 5.2.2 After Handling Class Imbalance (Label Encoding)

| Class | Precision | Recall | F1-score | Support |
|---|---|---|---|---|
| 0 (No Rain) | 0.84 | 0.73 | 0.78 | 19740 |
| 1 (Rain) | 0.77 | 0.86 | 0.81 | 19960 |
| **Accuracy** | | 0.80 | | 39700 |
| **Macro Avg** | 0.80 | 0.80 | 0.80 | 39700 |
| **Weighted Avg** | 0.80 | 0.80 | 0.80 | 39700 |

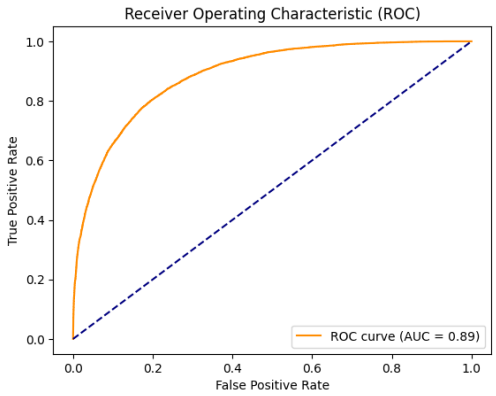
Table 5.2: Model performance after handling imbalance with label encoding




(a) Training and Validation loss




(b) Training and Validation Accuracy



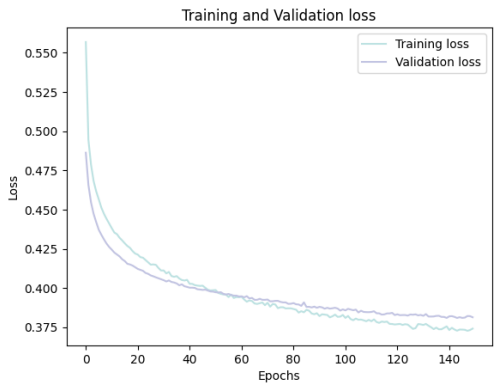

(c) Receiver Operating Characteristic (ROC)

**Observation:** Balancing the dataset led to significant improvement in the minority class recall (from 0.50 to 0.86), with only a slight reduction in precision. This shows that the model became more sensitive to rainy days. However, label encoding may not fully preserve the spatial distinctions between locations.

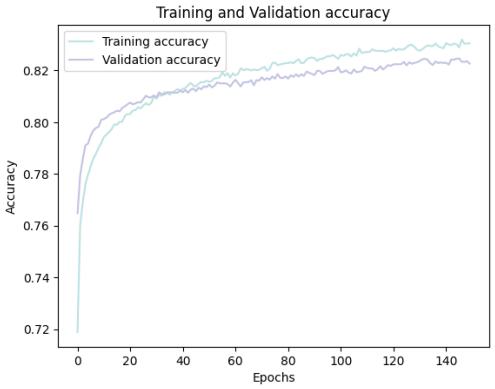
### 5.2.3 After Handling Class Imbalance (One-Hot Encoding)

| Class | Precision | Recall | F1-score | Support |
|---|---|---|---|---|
| 0 (No Rain) | 0.87 | 0.74 | 0.80 | 19740 |
| 1 (Rain) | 0.78 | 0.89 | 0.83 | 19960 |
| **Accuracy** | 0.82 | | | 39700 |
| **Macro Avg** | 0.82 | 0.82 | 0.82 | 39700 |
| **Weighted Avg** | 0.82 | 0.82 | 0.82 | 39700 |

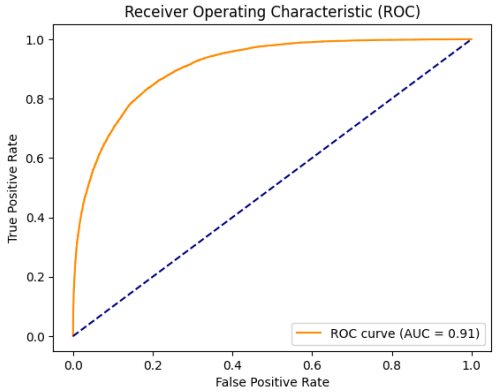
Table 5.3: Model performance after handling imbalance with one-hot encoding




(a) Training and Validation loss




(b) Training and Validation Accuracy




(c) Receiver Operating Characteristic (ROC)

**Observation:** One-hot encoding further improved overall performance, achieving the best recall (0.89) and highest F1-score (0.83) for the minority class. It allowed the model to differentiate between locations more effectively than label encoding, suggesting that spatial encoding has a significant effect on predictive accuracy.

### 5.2.4 Comparative Summary

- **Handling Imbalance Improved Recall:** Both encoding strategies significantly improved the model's ability to predict rain, especially by boosting recall from 0.50 to nearly 0.90.
- **Encoding Strategy Matters:** One-hot encoding outperformed label encoding by preserving the uniqueness of each location, leading to better generalization and accuracy.
- **Balanced Performance:** After addressing class imbalance, the model achieved a well-balanced trade-off between precision and recall, which is critical in weather forecasting scenarios where both false positives and false negatives carry risk.

### 5.2.5 Conclusion

This chapter demonstrates how preprocessing choices and encoding strategies have a measurable impact on model performance. The best results were obtained using one-hot encoding combined with class balancing, achieving an overall accuracy of 82% and strong recall for rainy days. These results validate the importance of treating class imbalance and choosing appropriate feature representations in classification tasks.

# Chapter 6

# Conclusion

This project focused on predicting rainfall in Australia using a neural network-based classification model. Through comprehensive data preprocessing, including handling missing values, encoding categorical variables (with a comparison of label vs. one-hot encoding), cyclic encoding of time-based features, and normalization, a robust dataset was prepared for training.

The final model demonstrated promising performance with over 82% accuracy on the test set when using one-hot encoding for the `Location` feature, significantly outperforming the label-encoded variant. The ROC curve and AUC score further confirmed the model's ability to distinguish between rainy and non-rainy days. By selecting an optimal classification threshold based on F1-score analysis, the model's predictions were further fine-tuned.

An important insight was the impact of addressing class imbalance. Before balancing, the model heavily favored the majority class, resulting in poor recall for the minority class. After applying appropriate preprocessing and threshold tuning, the model achieved balanced precision, recall, and F1-scores across classes. This is particularly important for rainfall prediction, where false negatives (predicting no rain when it actually rains) can have practical consequences.

In conclusion, the neural network model provides a solid foundation for binary rainfall prediction, with performance sensitive to preprocessing decisions, particularly encoding strategies and class distribution handling. Future work could explore ensemble methods, temporal models (like LSTMs), or integrating external climatic data to further enhance prediction accuracy and reliability.

# Bibliography


[1] The NumPy Developers. Numpy documentation. https://numpy.org/doc/, 2024.

[2] Google. Google colaboratory. https://colab.research.google.com, 2025. Used for model training and experimentation.

[3] Kaggle. Rain in australia dataset. https://www.kaggle.com/jsphyg/weather-dataset-rattle-package, 2025. Accessed: 2025-05-01.

[4] Scikit learn Developers. Scikit-learn documentation. https://scikit-learn.org/stable/documentation.html, 2024.

[5] OpenAI. Chatgpt: Language model by openai. https://chat.openai.com, 2025. Assisted with code explanations and report writing.

[6] TensorFlow Team. Tensorflow documentation. https://www.tensorflow.org/guide, 2024.

[7] The Pandas Development Team. Pandas documentation. https://pandas.pydata.org/docs/, 2024.

# Appendices

# Appendix A

# Screenshots

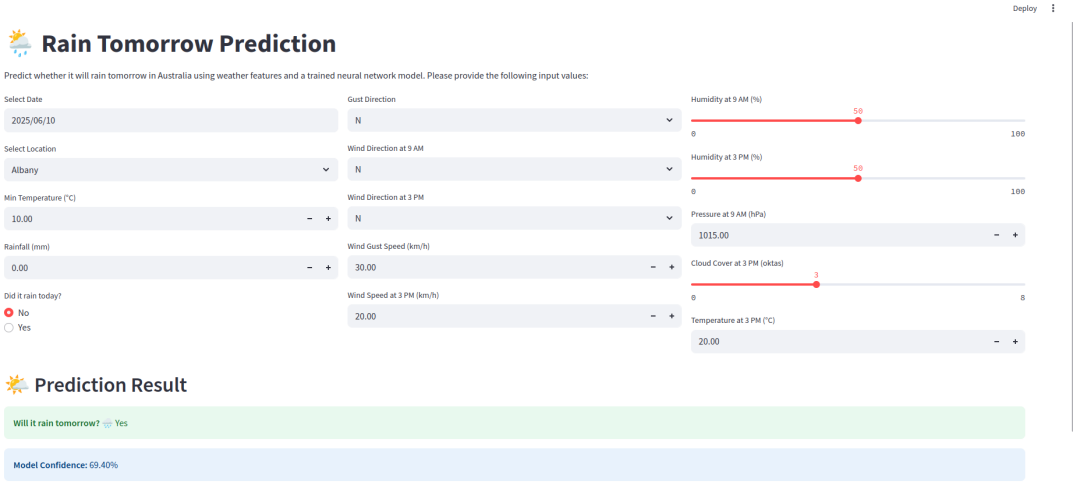



Figure A.1: Stramlit App Preview

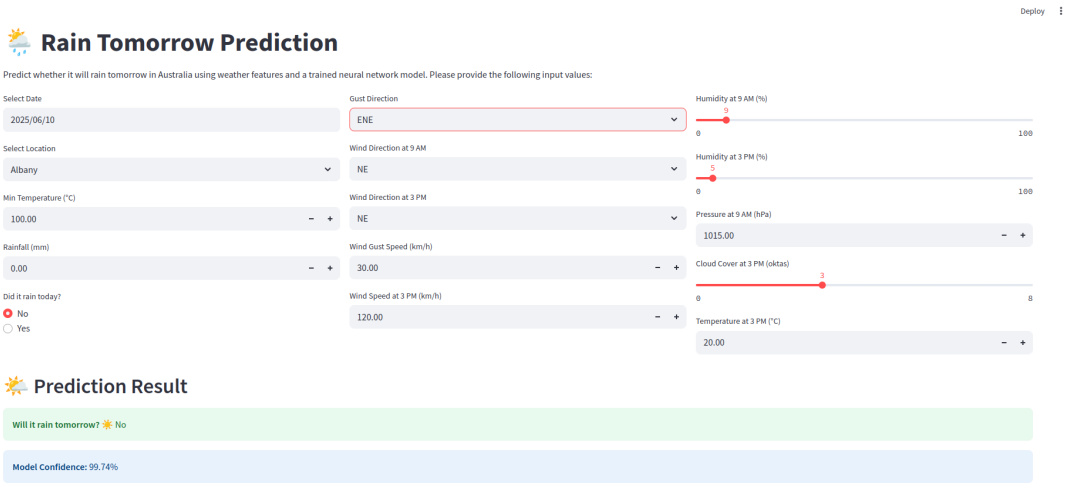



Figure A.2: Stramlit App Preview