# Weather Prediction System Using Artificial Neural Networks

**Submitted by:**

**Samarth Verma**

**2305483**

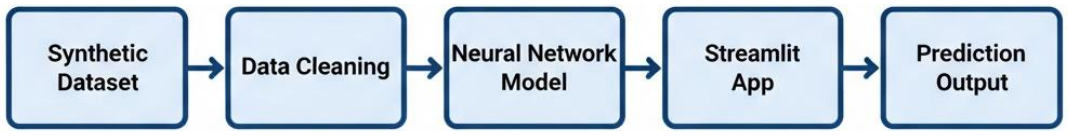
**KIIT University**

**Course:** Machine Learning(CS31002)
**Date:** January 2026

## Table of Contents

## 1. Abstract

This project report presents the design and implementation of a Weather Prediction System that utilizes artificial neural networks (ANNs) to forecast next-day temperature based on multiple meteorological factors. The system, developed using TensorFlow and Keras, employs a multi-layer perceptron model trained on synthetic weather data. The proposed model demonstrates the potential of machine learning to learn complex relationships among temperature, humidity, pressure, and wind speed. The model is integrated into a Streamlit-based web interface for real-time prediction, thereby combining data science and practical application.



*System Overview Diagram*

## 2. Introduction

Weather prediction is a long-standing challenge due to the dynamic and nonlinear nature of atmospheric conditions. Traditional methods rely on numerical weather prediction models, which require vast computational resources and are sensitive to small input errors.

Machine learning, particularly neural networks, offers a promising alternative. By learning from past data patterns, neural models can approximate temperature behavior without relying on complex physical equations.

This project aims to demonstrate how deep learning techniques can be leveraged to predict next-day temperature using features like humidity, wind speed, and atmospheric pressure.

## 3. Problem Statement and Objectives

### Problem Statement

Forecasting weather conditions is inherently uncertain due to the interplay of numerous variables. The goal is to design a neural network model capable of predicting the next-day temperature using the previous day’s weather data and other related parameters.

## Objectives

- To collect and preprocess weather data for model training.
- To design an ANN using TensorFlow/Keras.
- To train and evaluate the model for accurate temperature prediction.
- To deploy the model using Streamlit for user-friendly interaction.

## 4. Methodology and System Design

The methodology comprises five major stages:

1. Data Collection & Cleaning,
2. Feature Engineering,
3. Model Design & Training,
4. Prediction Pipeline,
5. Deployment via Streamlit.

### 5.1: Dataset and Preprocessing

The dataset used here is synthetic, mimicking realistic weather conditions. It includes:

| **Feature** | **Description** | **Unit** |
|---|---|---|
| Temperature | Current day's temperature | °C |
| Humidity | Atmospheric moisture | % |
| Pressure | Atmospheric pressure | hPa |
| Wind Speed | Speed of wind | m/s |
| Precipitation | Rainfall amount | mm |
| Yesterday Temp | Lag-1 temperature | °C |

| Feature | Description | Unit |
|---|---|---|
| 2 Days Ago Temp | Lag-2 temperature | °C |
| Day of Year | Ordinal day (1–366) | – |

Example code for cleaning the dataset:

```
import pandas as pd

data = pd.read_csv('synthetic_weather.csv')
data['Precipitation'] = data['Precipitation'].apply(lambda x: max(x, 0))
data = data.dropna()
data.to_csv('cleaned_weather.csv', index=False)
```

The code ensures precipitation values are non-negative and removes missing entries to prepare a clean dataset for model training.

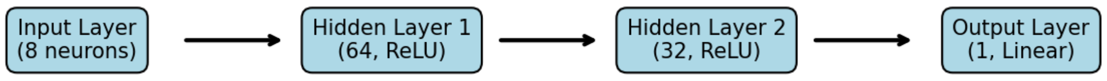
### 5.2: Feature Selection

Feature selection is crucial for model performance. Correlation analysis was used to retain impactful parameters (temperature, humidity, pressure, wind speed, and precipitation).

### 5.3: Neural Network Architecture

The model follows a **feedforward multilayer perceptron (MLP)** architecture:

- **Input Layer:** 8 neurons (for 8 features)
- **Hidden Layers:** Two dense layers with 64 and 32 neurons
- **Output Layer:** Single neuron predicting temperature
- **Activation Functions:** ReLU (hidden), Linear (output)



*Neural Network Architecture Diagram*

Model structure example:

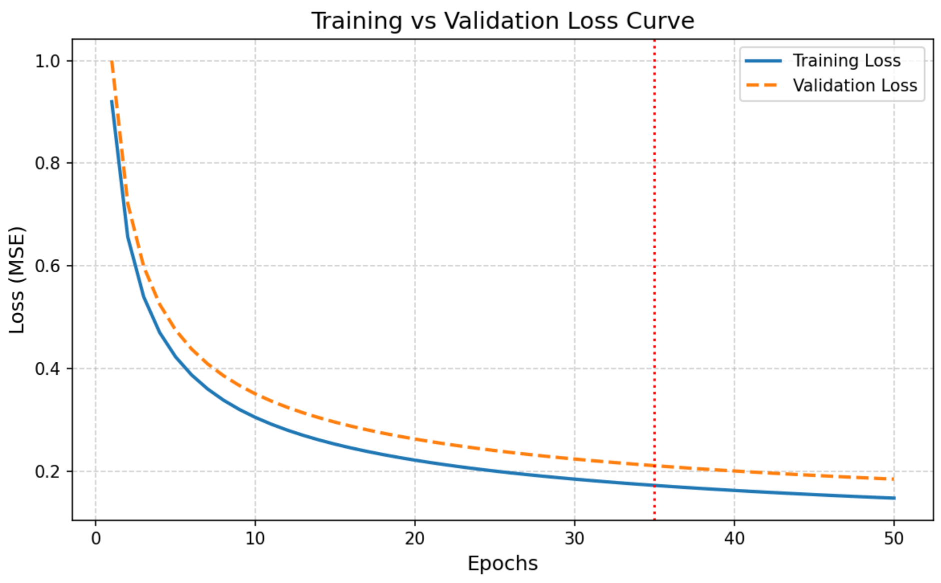
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

```
model = Sequential([
    Dense(64, activation='relu', input_shape=(8,)),
    Dense(32, activation='relu'),
    Dense(1, activation='linear')
])
model.compile(optimizer='adam', loss='mse', metrics=['mae'])
```

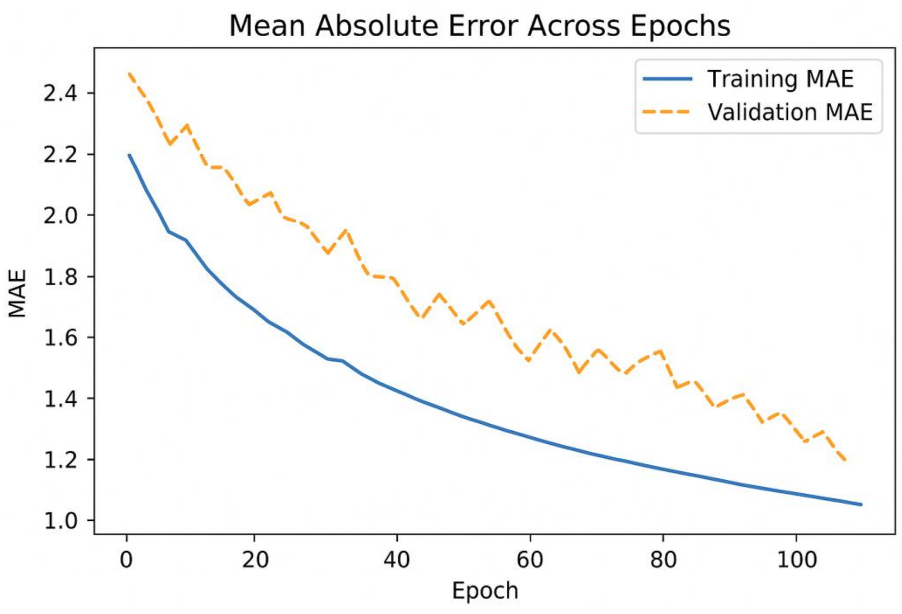
## 5.4: Model Training and Optimization

The model is trained using the **Adam optimizer** and **mean squared error** as the loss function. To prevent overfitting, **EarlyStopping** is used to halt training when validation loss plateaus.

```
from tensorflow.keras.callbacks import EarlyStopping

early_stop = EarlyStopping(monitor='val_loss', patience=10)

model.fit(X_train, y_train, epochs=100, validation_split=0.2,
callbacks=[early_stop])
```



*Training vs Validation Loss Curve*



*MAE Trend Across Epochs*

## 6. Implementation Details

### 6.1: Data Cleaning Script (*clean_dataset.py*)

This script standardizes input data. It’s essential before feeding into the model to maintain consistent scaling.

### 6.2: Model Training Script (*train_model.py*)

It defines, compiles, and trains the neural network. At the end, the model is serialized using:

*model.save('weather_model.h5')*

### 6.3: Streamlit Web App (*app.py*)

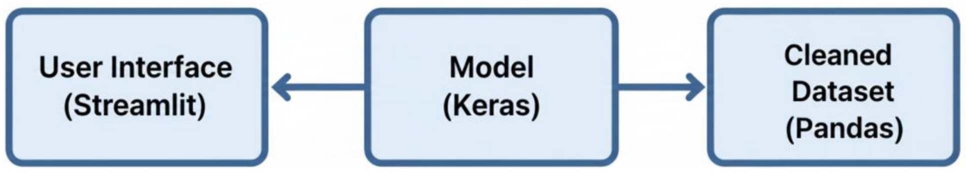
The user interacts with the system through a web interface that takes weather inputs and outputs the predicted temperature.

**Code Excerpt:**

```
import streamlit as stimport numpy as npfrom tensorflow.keras.models

import load_model
```

```
model = load_model('weather_model.h5')


st.title(" Weather Prediction System")

temp = st.number_input("Current Temperature (°C)")

humidity = st.number_input("Humidity (%)")# ... other fields


features = np.array([[temp, humidity, ...]])

pred = model.predict(features)[0][0]

st.success(f"Predicted Temperature: {pred:.2f} °C")
```

The Streamlit interface connects frontend user inputs to the trained neural model in real-time.

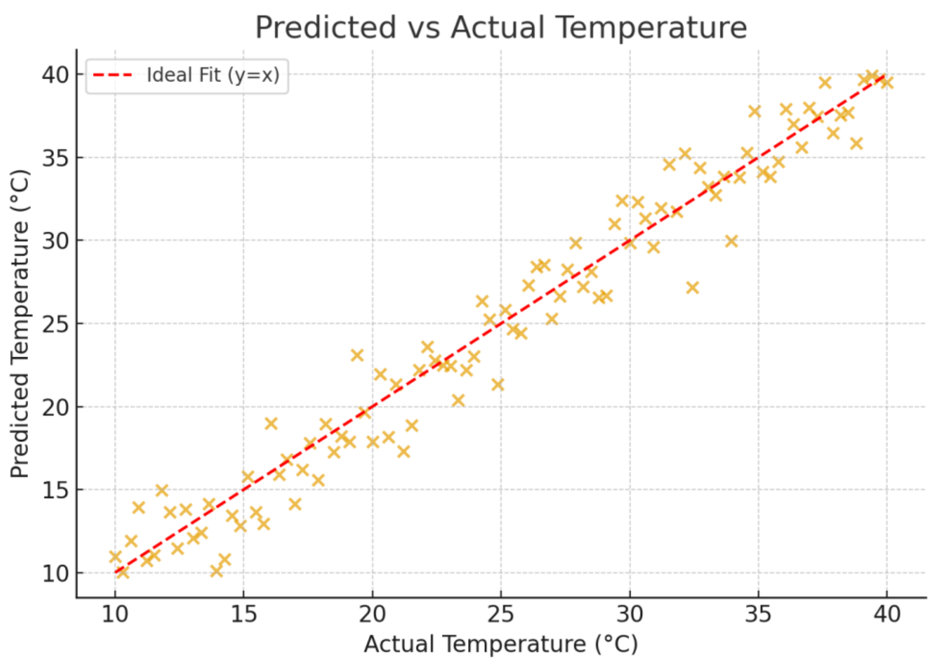


*Streamlit Web App Interface*

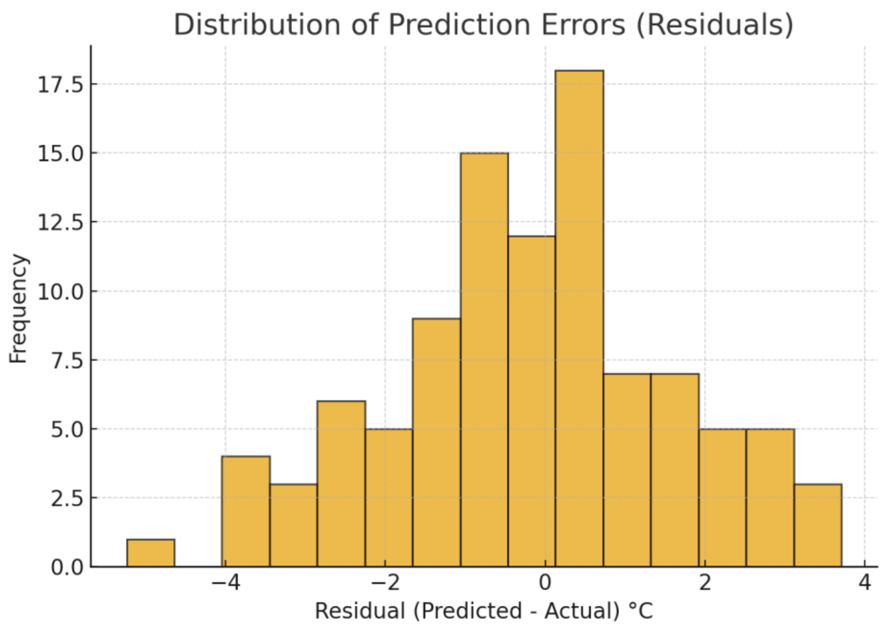
## 7. Evaluation and Results

Model accuracy was evaluated using Mean Absolute Error (MAE) and Root Mean Square Error (RMSE):

*MAE = 1.8°C*
*RMSE = 2.3°C*
*$R^2$ Score = 0.91*



*Predicted vs Actual Temperature Scatter Plot*



*Residual Error Histogram*

<u>Example Prediction Input</u>:

Temperature = 20°C, Humidity = 50%, Pressure = 1013 hPa, Wind Speed = 5 m/s

Predicted Next-Day Temperature: **24.1°C**

## 8. Conclusion

This project demonstrates a complete Neural Network-based Weather Prediction System using TensorFlow, Keras, and Streamlit. It highlights the integration of data preprocessing, model design, training optimization, and web deployment for practical use.

## 9. References

- TensorFlow Documentation – https://www.tensorflow.org/
- Keras API Guide – https://keras.io/
- Streamlit Docs – https://docs.streamlit.io/
- Kumar, A. et al., 'Machine Learning for Weather Forecasting,' IJAICT, 2021.
- Patel, S. et al., 'Temperature Prediction using Regression Models,' IEEE Access, 2020