AI for Weather Prediction in Hills

Contributor: Dr. Chandni Agarwal
National ICT Awardee 2016 | CBSE Awardee 2021 | IT Educator & AI Mentor

Learning Outcomes (at the end)

After this tutorial, learners can:

- Prepare a small weather dataset for a hill location.
- Build a basic forecast using **Colab** (low-code).
- Evaluate predictions with **MAE** and visualize results clearly.
- Communicate findings with a **chart**.

Objective:

To Predict **tomorrow's temperature** (or rainfall) for a hill location using past weather data.

What You Need

- A Google account **Colab**
- 15–60 days of daily data (Date, Temperature °C, Rainfall mm, Humidity %)
- Optional: Canva (to make one summary chart/infographic)

Step by Step Demo

1: Prepare CSV

In Google Sheets: File → Download → **.CSV** (includes `Date, Temp_C, Rain_mm, Humidity_pct`).

2. Open Colab

- Go to **colab.research.google.com** → New Notebook
- Upload the CSV (left sidebar → Files → Upload).

3: Code to predict

Use **yesterday's values** to predict **tomorrow's temperature** (a strong, simple baseline for hills). weatherprediction.ipynb - colab link
https://drive.google.com/file/d/1mCl8MQ8zBT8ZPIDkaPsI7fvPpjb5MZWB/view?usp=sharing - csv file

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error
from sklearn.linear_model import LinearRegression

# 1) Load data
df = pd.read_csv('/content/your_weather.csv', parse_dates=['Date'])
df = df.sort_values('Date').reset_index(drop=True)

# 2) Create lag features (yesterday’s values)
df['Temp_yday'] = df['Temp_C'].shift(1)
df['Rain_yday'] = df['Rain_mm'].shift(1)
df['Humid_yday'] = df['Humidity_pct'].shift(1)

# Drop first row with NaNs from shift
df = df.dropna()

# 3) Train/Test split (last 20% days for testing)
split = int(len(df)*0.8)
train, test = df.iloc[:split], df.iloc[split:]

X_train = train[['Temp_yday','Rain_yday','Humid_yday']]
y_train = train['Temp_C']
X_test  = test[['Temp_yday','Rain_yday','Humid_yday']]
y_test  = test['Temp_C']
# 4) Train model
model = LinearRegression()
model.fit(X_train, y_train)
# 5) Evaluate
preds = model.predict(X_test)
mae = mean_absolute_error(y_test, preds)
print('MAE (°C):', round(mae, 2))
# 6) Predict tomorrow from today’s last row
last = df.iloc[-1]
X_next = [[last['Temp_yday'], last['Rain_yday'], last['Humid_yday']]]
print('Tomorrow Temp (°C):', round(model.predict(X_next)[0], 2))
# 7) Plotting line chart
import matplotlib.pyplot as plt
plt.figure(figsize=(12, 6))
plt.plot(test['Date'], y_test, label='Actual Temperature (°C)')
plt.plot(test['Date'], preds, label='Predicted Temperature (°C)')
plt.xlabel('Date')
plt.ylabel('Temperature (°C)')
plt.title('Actual vs Predicted Temperature Over Time')
```

```
plt.legend()
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

- **MAE (°C)** ≈ average error. Under **2–3°C** is decent for tiny datasets.
- You can also predict **Rain_mm** by setting `y_train = train['Rain_mm']` and changing target names similarly.

Output :

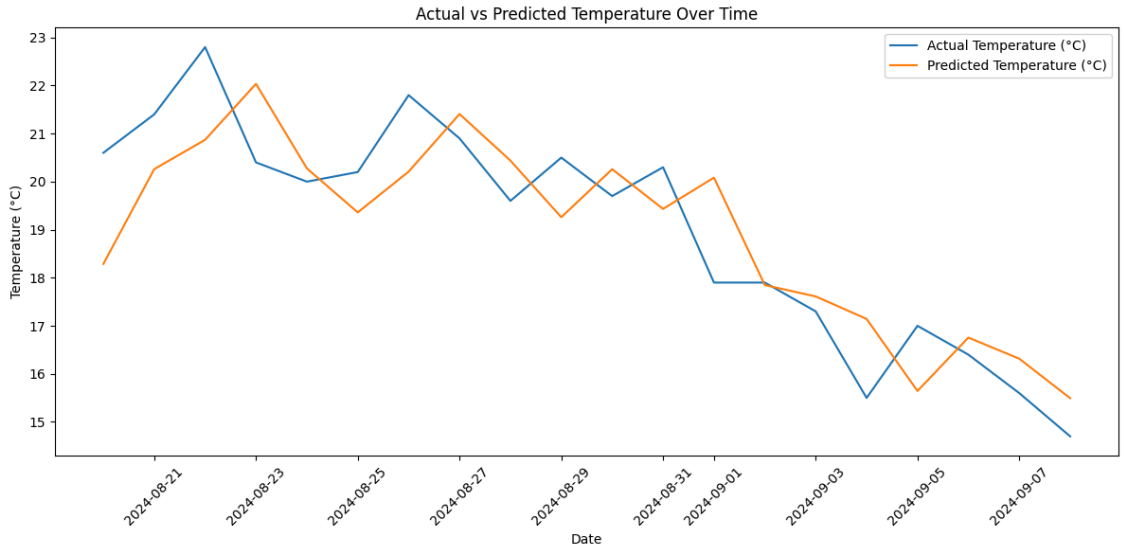
```
MAE (°C): 1.06

Tomorrow Temp (°C): 15.49
```

Colab Link - *Untitled0.ipynb* *Weather prediction*

CSV file -
https://drive.google.com/file/d/1mCl8MQ8zBT8ZPlDkaPsI7fvPpjb5MZWB/view?usp=sharing

4. Plot Actual vs Predicted (Optional)

Use Colab to draw a quick line plot and screenshot it for your report.



Conclusion:

The linear regression model effectively analyzed past weather trends of the hilly region (Uttarakhand) using temperature, rainfall, and humidity data. With a low mean absolute error, the model provides a reliable estimate of the next day's temperature, demonstrating how simple

machine learning techniques can support weather forecasting and environmental analysis in regional contexts.

video link - https://youtu.be/pwvCvWlLjs4