**REPORT**

**EDA-CSE3040**

**Team – M**

Bheemireddy ShabareshwarReddy – 23MIA1091

Peta Siva Nandhan Reddy – 23MIA1111

Akash Annam – 23MIA1112

**Slot: E2+TE2**

**WEATHER PREDICTION**

# Rainfall Prediction Using Machine Learning

# 2. Acknowledgement

We would like to express our sincere gratitude to **Prof. Asnath Victy Phamilay**, our faculty mentor for Exploratory Data Science, for her invaluable guidance, support, and constructive feedback throughout this project. Her expertise and encouragement were instrumental in shaping our understanding and application of machine learning in the context of climate science.

We also extend our thanks to our peers and the department for fostering a collaborative and intellectually stimulating environment, which enabled us to explore innovative solutions aligned with the Sustainable Development Goals.

# 3. Abstract

Climate change is one of the most pressing global challenges of our time, directly affecting ecosystems, economies, and communities. Accurate weather forecasting and rainfall prediction are critical for sustainable agriculture, disaster preparedness, and environmental management. This project is aligned with **United Nations Sustainable Development Goal 13 (STG-13): Climate Action**, aiming to leverage machine learning for climate-related predictions.

We designed and implemented two machine learning models using two distinct datasets:

* In **Review 2**, we used historical daily weather data (weather.csv) to develop a **Ridge Regression model** that forecasts the **next day’s maximum temperature**. Through backtesting and feature engineering (rolling averages and expanding means), we achieved a mean absolute error (MAE) of approximately **4.79**.
* In **Review 3**, we improved upon our approach by addressing faculty feedback and focused on **predicting rainfall occurrence** using a separate dataset (Rainfall.csv). We implemented a **Random Forest Classifier**, handled class imbalance through downsampling, and tuned hyperparameters using GridSearchCV. The model showed promising accuracy and reliability in rainfall classification.

Our work demonstrates how data science and machine learning can be practically applied to support climate action initiatives and contribute to environmental sustainability.

# 4. Table of Contents

|  |  |  |
| --- | --- | --- |
| S.No | Content | Page Number |
|  | Cover Page | 0-1 |
|  | Acknowledgement | 2-2 |
|  | Abstract | 2-2 |
|  | Table of Contents | 3-3 |
|  | Introduction | 3-4 |
|  | Problem Statement | 4-5 |
|  | Dataset Description | 5-8 |
|  | Data Preprocessing | 8-11 |
|  | Feature Engineering | 11-14 |
|  | Model Building | 14-17 |
|  | Evaluation Metrics | 18-19 |
|  | Visualizations | 19-20 |
|  | Review-wise Improvements | 20-21 |
|  | Results & Interpretation | 21-21 |
|  | Conclusion | 21-22 |
|  | Future Scope | 22-23 |
|  | References | 23-24 |
|  | Appendix | 24-24 |

# Communications materials - United Nations Sustainable ...5. Introduction

5.1 What is Climate Change?

Climate change refers to long-term shifts and alterations in temperature, precipitation patterns, wind patterns, and other elements of the Earth's climate system. While climate variability has occurred naturally over millennia, the current phase of climate change is significantly driven by human activities, notably the burning of fossil fuels, deforestation, and large-scale industrialization. These actions have led to increased greenhouse gas emissions, contributing to global warming and frequent extreme weather events.

5.2 Why is Weather and Rainfall Prediction Important?

Weather and rainfall predictions play a crucial role in the day-to-day functioning of society. Accurate forecasts can:

* Help **farmers** plan irrigation and harvesting schedules
* Enable **governments** and **disaster management agencies** to issue timely alerts for floods, droughts, or heatwaves
* Support **climate scientists** in monitoring trends and anomalies
* Aid **transportation and aviation sectors** in route planning and safety

Moreover, in regions prone to erratic rainfall or water scarcity, predictive models can inform long-term planning for water resource management.

5.3 About STG-13: Climate Action

**Sustainable Development Goal 13 (STG-13)** — *“Take urgent action to combat climate change and its impacts”* — emphasizes the need for predictive capabilities, data-driven decision-making, and climate-resilient infrastructure. Our project aligns with this goal by leveraging machine learning to provide actionable insights from climate data.

5.4 Real-life Impact of Predictive Modeling

The application of machine learning in climate science has immense potential. By training algorithms on historical weather data, we can make:

* **Temperature forecasts** to alert citizens during extreme cold or heat
* **Rainfall predictions** that help farmers reduce crop losses
* **Flood risk predictions** that support evacuation planning
* **Energy grid optimizations** for renewable resource management

Such data-driven interventions support both immediate action and long-term policy-making to combat climate change.

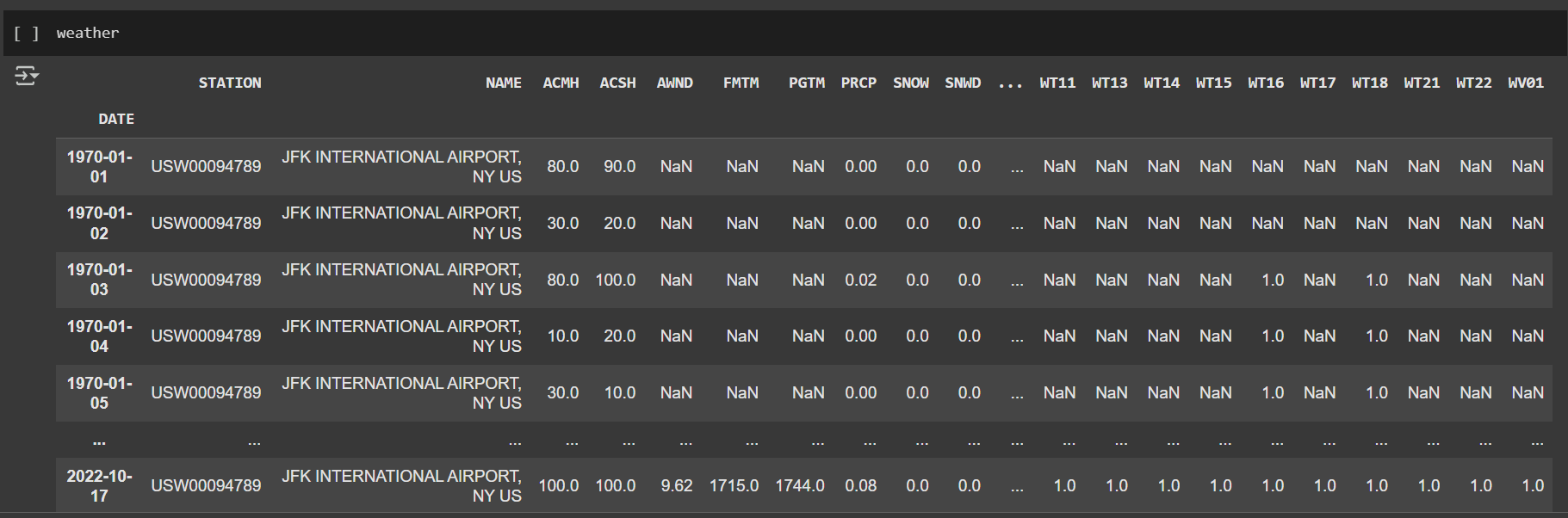
# 6. Problem Statement

Our project involves the development of machine learning models on two different climate-related problems, each using a unique dataset and approach.

6.1 Review 2: Predicting Next-Day Maximum Temperature (Regression)

In the first phase, we used a dataset comprising daily weather observations (weather.csv) to predict the **next day's maximum temperature (TMAX)**. The goal was to explore regression techniques to forecast numeric climate indicators based on historical patterns. This model helps in short-term weather planning and early warnings.

* **Approach Used**: Ridge Regression
* **Challenge Addressed**: Time-series forecasting using engineered statistical features



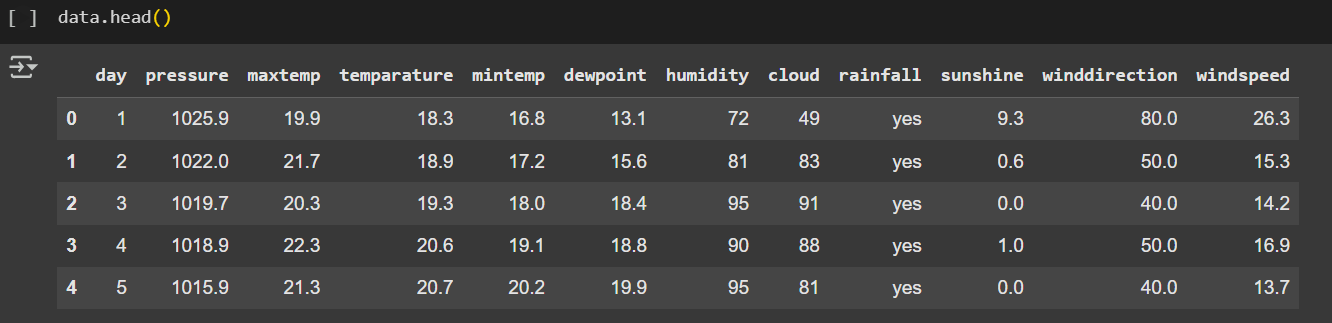
* **Output**: Continuous numerical value (temperature in °F)

6.2 Review 3: Predicting Rainfall Occurrence (Binary Classification)

Based on faculty feedback and further enhancement, we shifted focus to a classification task in the second phase. Using a refined dataset (Rainfall.csv), we built a model to predict whether **it will rain on a given day** or not. This binary prediction is more aligned with practical decision-making in agriculture and disaster preparedness.

* **Approach Used**: Random Forest Classifier with hyperparameter tuning
* **Challenge Addressed**: Handling class imbalance, optimizing accuracy
* **Output**: Binary label (1 = Rainfall, 0 = No Rainfall)

Both problem statements contribute to **climate-aware decision-making** and showcase the application of machine learning in addressing global climate challenges under STG-13.



# 7️. Dataset Description

A. Review 2 – weather.csv

For our initial model (Review 2), we used a publicly available weather dataset from **NOAA** (National Oceanic and Atmospheric Administration), downloaded via Kaggle. It contains over 19,000 records from **January 1, 1970 to October 21, 2022**, collected at JFK International Airport, NY, USA.

**Key Features:**

* **DATE:** Timestamp (set as index)
* **prcp:** Daily precipitation (inches)
* **snow:** Snowfall (inches)
* **snwd:** Snow depth (inches)
* **tmax:** Maximum temperature (°F)
* **tmin:** Minimum temperature (°F)

**Data Challenges:**

* Several columns like acmh, wsfm, wt01, etc., had over **50% missing values**, so we **dropped them**.
* Some placeholder values like 9999 were not present in major columns, but we still validated and cleaned them.
* Columns were converted to lowercase and filtered to retain only those with **<5% missing data**.

**Code Snippet Used:**

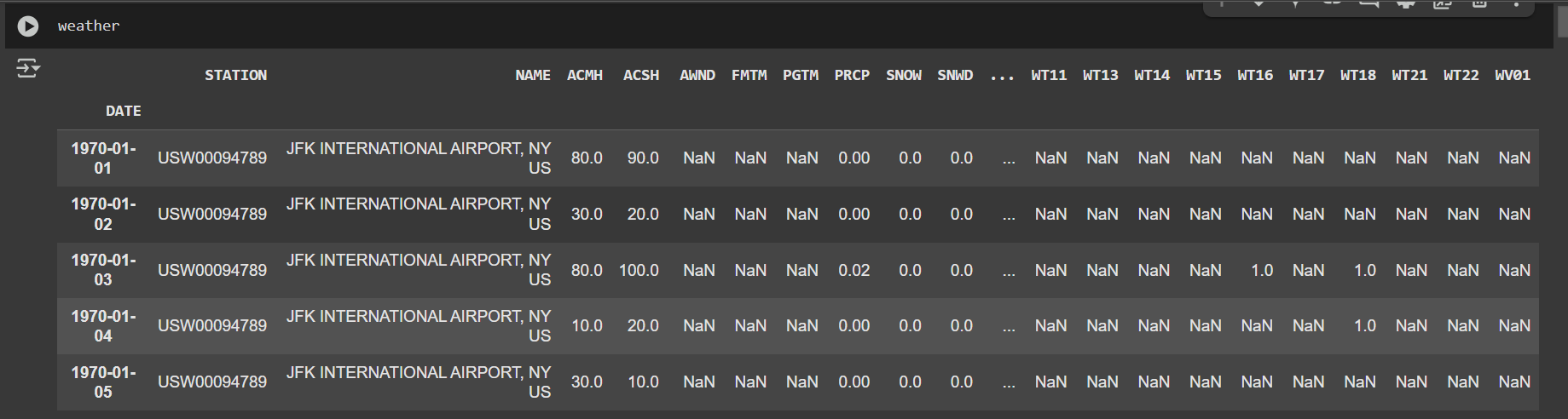
weather = pd.read\_csv("weather.csv", index\_col="DATE")

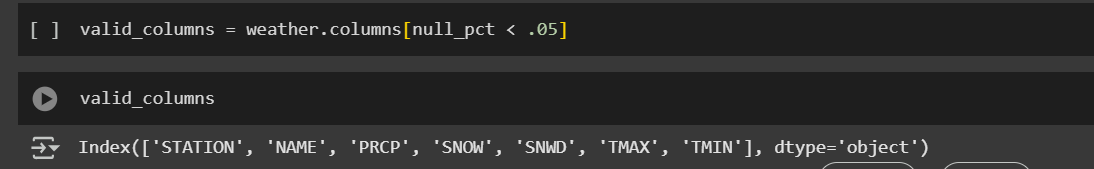
null\_pct = weather.apply(pd.isnull).sum()/weather.shape[0]

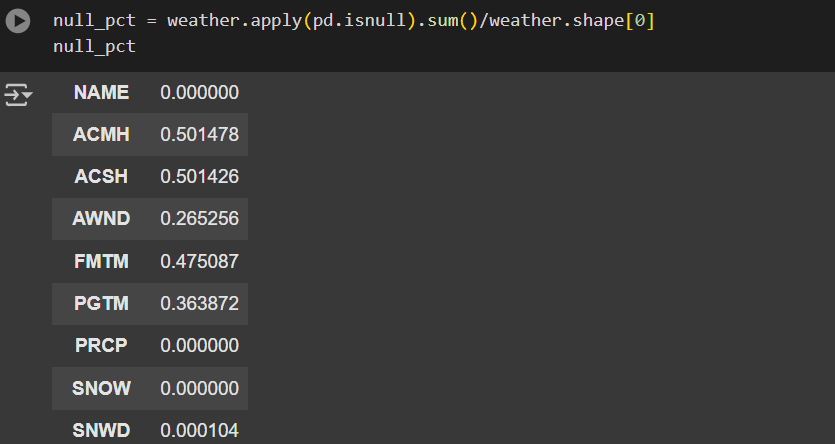
valid\_columns = weather.columns[null\_pct < .05]

weather = weather[valid\_columns].copy()

weather.columns = weather.columns.str.lower()







B. Review 3 – Rainfall.csv

For our second and improved version (Review 3), we worked with a different dataset focused on **rainfall prediction** using **daily atmospheric parameters**.

**Key Features:**

* **pressure:** Atmospheric pressure (hPa)
* **dewpoint:** Dew point temperature (°C)
* **humidity:** Relative humidity (%)
* **cloud:** Cloud cover (%)
* **sunshine:** Duration of sunshine (hours)
* **winddirection:** Wind direction (degrees)
* **windspeed:** Wind speed (km/h)
* **rainfall:** **Target variable** – indicates rainfall occurrence (yes / no)

We converted the rainfall column into binary form:

* "yes" → 1
* "no" → 0

**Class Imbalance:**

* There were significantly more "no rainfall" entries than "rainfall" ones.
* To address this, we **downsampled** the majority class to create a balanced dataset.

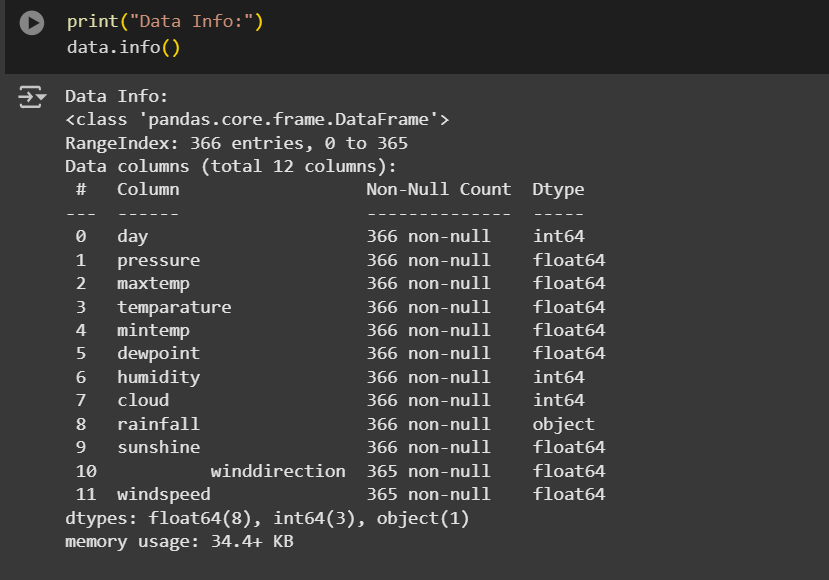
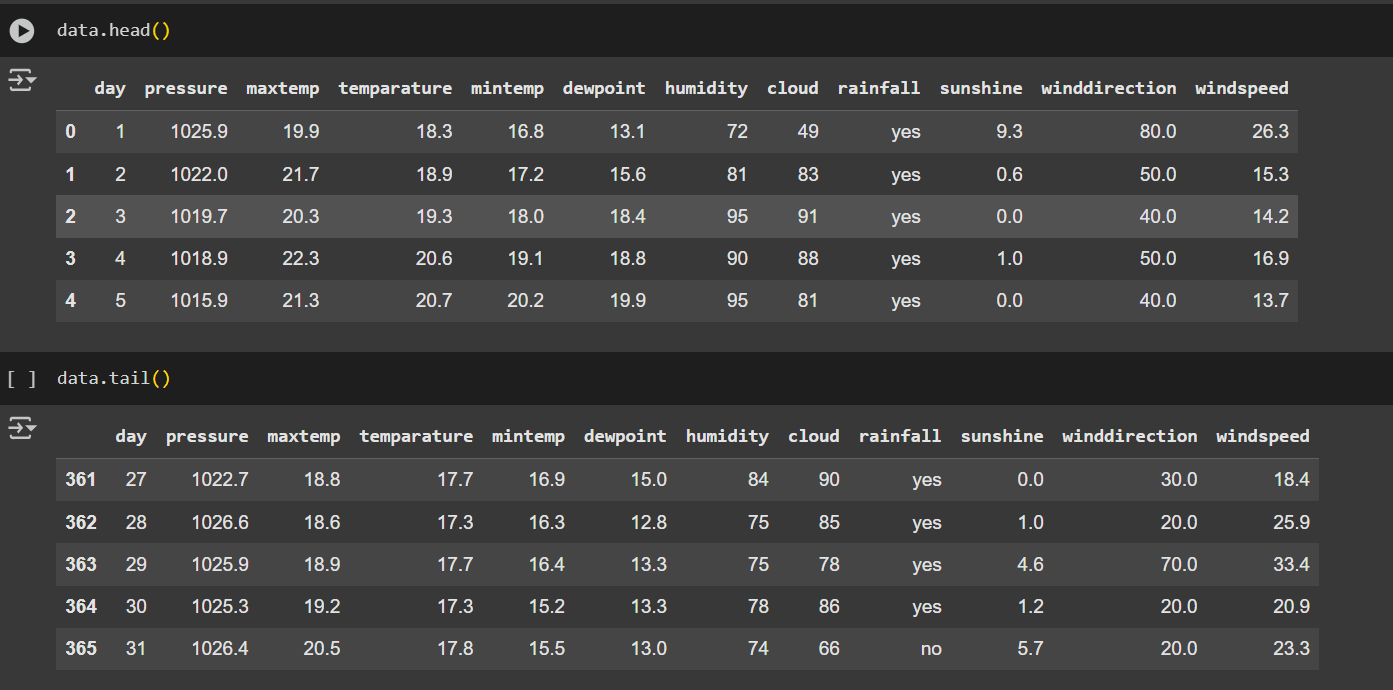
**Source:** https://drive.google.com/file/d/1zXfbIE6USi6UidxgpB2XcyCQmKKd8iGD/view?usp=sharing

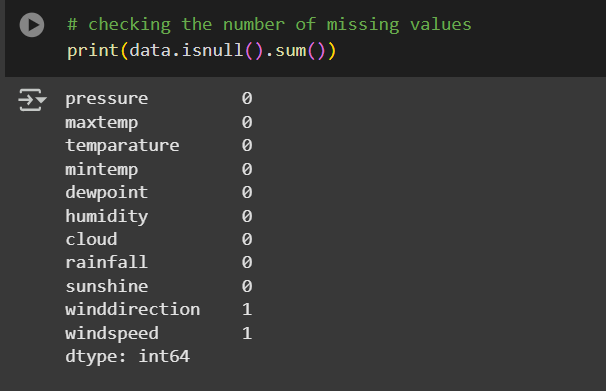
**Code Snippet Used:**

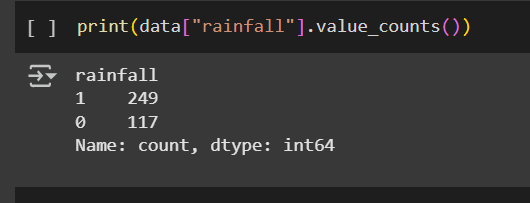
data = pd.read\_csv("Rainfall.csv")

data.columns = data.columns.str.strip()

data["rainfall"] = data["rainfall"].map({"yes": 1, "no": 0})







# 8️. Data Preprocessing

Common Cleaning Steps

**Null Value Handling:**

* **Review 2:** We removed columns with more than **5% missing data**, as instructed.
* **Review 3:** We retained all columns but **imputed missing values** using:
  + mode() for categorical column winddirection
  + median() for numeric column windspeed

**🔤 Column Cleaning:**

* Removed unwanted spaces using data.columns.str.strip()
* Converted all column names to lowercase

**🧮 Type Conversion:**

* For time series: Converted DATE index to datetime (pd.to\_datetime())

**📌 Review-Specific Steps**

**🔹 Review 2:**

* Created a **target column** for next-day prediction using:

Code:

weather["target"] = weather.shift(-1)["tmax"]

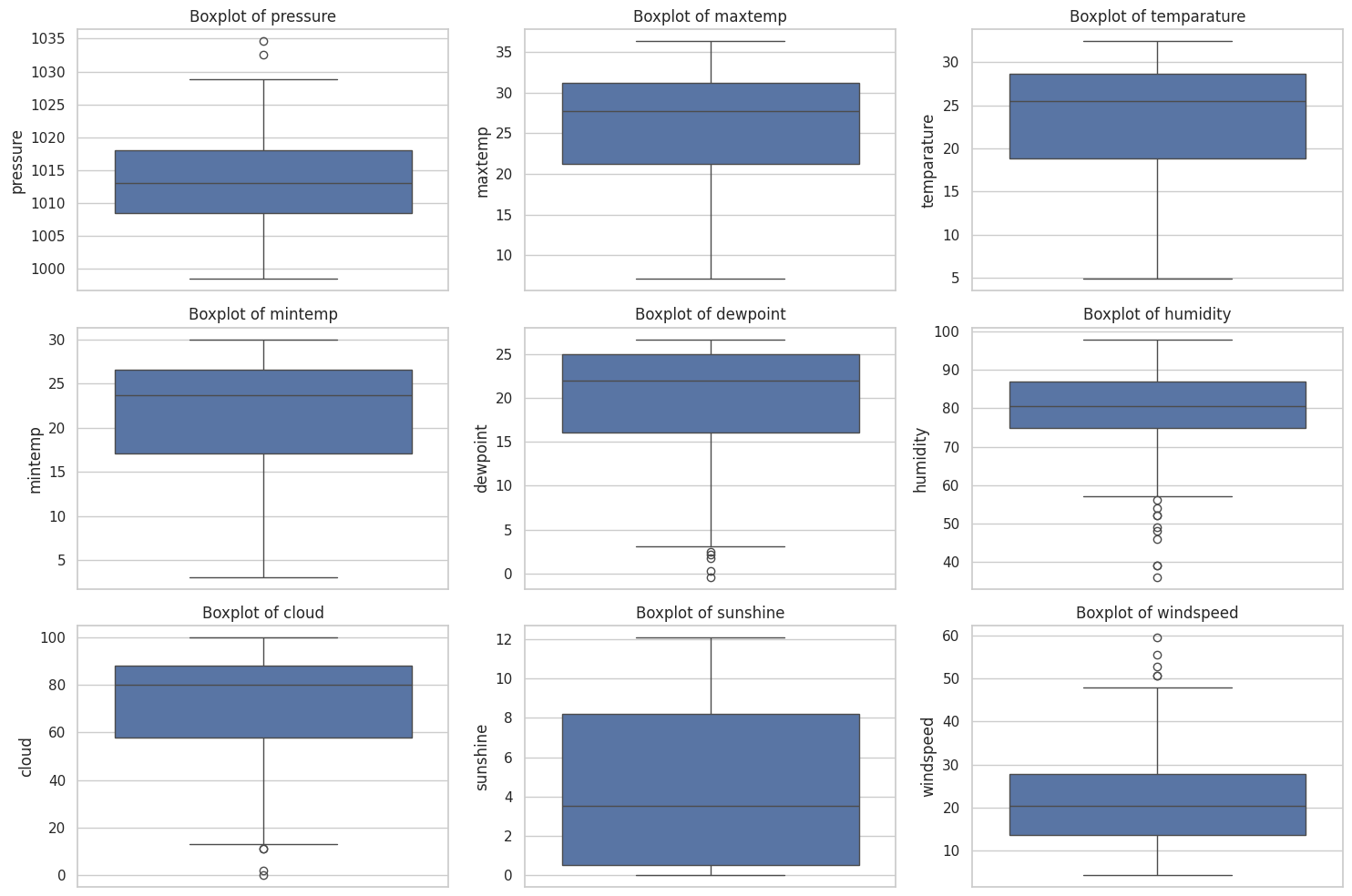
**Review 3:**

* Converted target column rainfall to binary values
* Dropped highly **correlated columns** like maxtemp, mintemp, temparature using heatmap insights

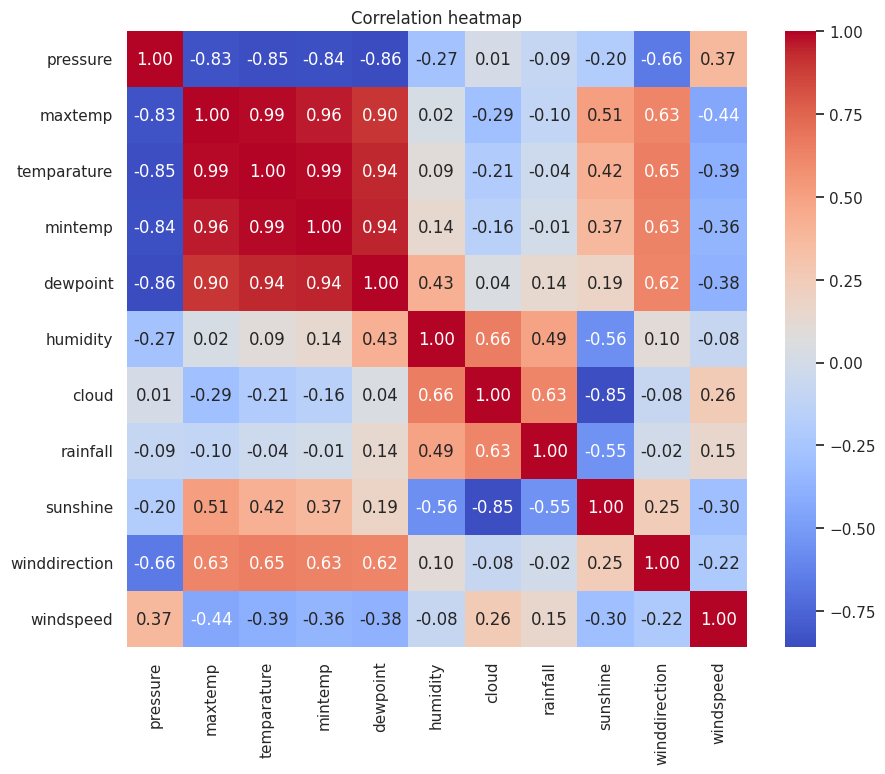
**Outlier Handling**

* Used **boxplots** to visualize skewness and detect outliers in features like pressure, humidity, etc.
* This helped in understanding variability and avoiding misleading models

**Visuals   
Boxplots**: Use code block to show 9 feature-wise plots



* **Heatmap**: For feature correlation (use sns.heatmap())



**Code Snippet Used:**

plt.figure(figsize=(15, 10))

for i, column in enumerate(['pressure', 'humidity', 'dewpoint', 'cloud', 'sunshine', 'windspeed'], 1):

plt.subplot(3, 3, i)

sns.boxplot(data[column])

plt.title(f"Boxplot of {column}")

plt.tight\_layout()

plt.show()

sns.heatmap(data.corr(), annot=True, cmap="coolwarm", fmt=".2f")

# 9️.Feature Engineering

**Applied in Review 2 – weather.csv**

To improve our model’s ability to predict the **next day’s maximum temperature**, we performed **feature engineering** using time-based weather patterns. Since temperature and precipitation often show short- and long-term seasonal behavior, we extracted trends using **rolling averages**, **percentage differences**, and **expanding means**.

**A. Rolling Averages**

We created rolling averages for **3-day** and **14-day windows** for key weather features like tmax, tmin, and prcp.

**Why?**

* Smoothens noisy data
* Helps model understand **short-term and mid-term trends**

**Code Snippet:**

def compute\_rolling(weather, horizon, col):

label = f"rolling\_{horizon}\_{col}"

weather[label] = weather[col].rolling(horizon).mean()

weather[f"{label}\_pct"] = (weather[label] - weather[col]) / weather[col]

return weather

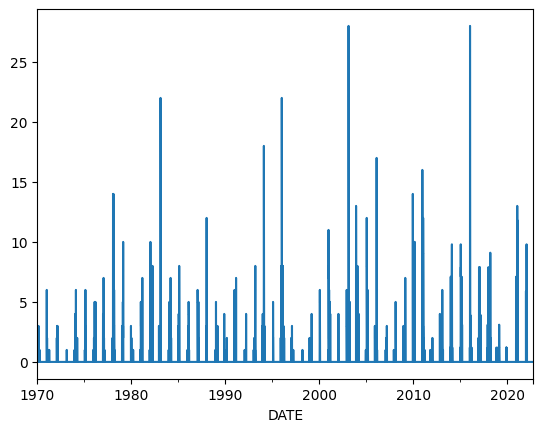
rolling\_horizons = [3, 14]

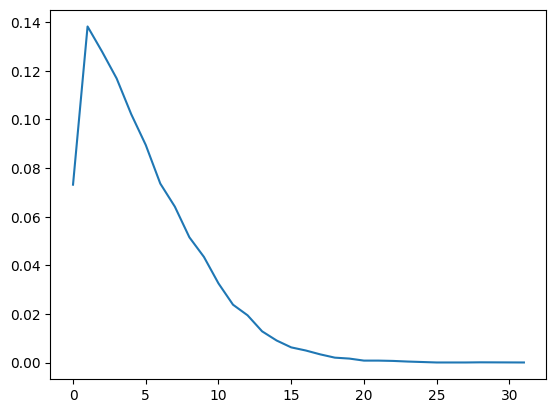
for horizon in rolling\_horizons:

for col in ["tmax", "tmin", "prcp"]:

weather = compute\_rolling(weather, horizon, col)

Histogram of rolling\_14\_prcp\_pct (percentage difference)



Line plot showing original vs. 3-day rolling tmax

**B. Percentage Difference from Rolling Mean**

For every rolling average, we also computed its **relative deviation** from the original value, using the formula:

​

This helped the model understand **how abnormal or normal** the current reading is compared to recent trends.

**C. Expanding Averages (Monthly & Daily Patterns)**

We also calculated **expanding mean values** for each feature on:

* A **monthly** basis: Average of values seen so far in that month
* A **day-of-year** basis: Seasonal effect of certain days

**Why?**

* Accounts for **long-term patterns** like seasonal changes
* Useful in learning **cyclical weather behavior**

**Code Snippet:**

def expand\_mean(df):

return df.expanding(1).mean()

for col in ["tmax", "tmin", "prcp"]:

weather[f"month\_avg\_{col}"] = weather[col].groupby(weather.index.month, group\_keys=False).apply(expand\_mean)

weather[f"day\_avg\_{col}"] = weather[col].groupby(weather.index.day\_of\_year, group\_keys=False).apply(expand\_mean)

**D. Target Column Creation**

Finally, we created the **target column** by shifting the tmax column by one day:

target=tmaxnext day\text{target} = \text{tmax}\_{\text{next day}}target=tmaxnext day​

weather["target"] = weather.shift(-1)["tmax"]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **prcp** | **tmax** | **rolling\_3\_tmax** | **rolling\_3\_tmax\_pct** | **month\_avg\_tmax** | **target** |
| 0.00 | 28 | NaN | NaN | 28.00 | 31 |
| 0.00 | 31 | NaN | NaN | 29.50 | 38 |
| 0.02 | 38 | 32.33 | -0.15 | 32.33 | 31 |

# 10. Model Building

**Review 2: Ridge Regression**

We used **Ridge Regression** for our first model to predict the next day’s maximum temperature.

**Why Ridge?**

* Ridge regression is a **regularized linear model** that helps prevent **overfitting**.
* It is particularly useful when features are **highly correlated** (e.g., tmax, tmin).
* The alpha parameter controls the penalty applied to the coefficients.

**Code Used:**

from sklearn.linear\_model import Ridge

rr = Ridge(alpha=0.1)

**Backtesting Approach**

Since the data is **time-series**, we didn’t use a regular train-test split. Instead, we used **backtesting**:

* It trains the model on all data up to time t and tests on the next step days.
* This mimics real forecasting situations.

**Code Snippet:**

def backtest(weather, model, predictors, start=3650, step=90):

all\_predictions = []

for i in range(start, weather.shape[0], step):

train = weather.iloc[:i, :]

test = weather.iloc[i:(i+step), :]

model.fit(train[predictors], train["target"])

preds = model.predict(test[predictors])

preds = pd.Series(preds, index=test.index)

combined = pd.concat([test["target"], preds], axis=1)

combined.columns = ["actual", "prediction"]

combined["diff"] = abs(combined["actual"] - combined["prediction"])

all\_predictions.append(combined)

return pd.concat(all\_predictions)

**Results – Ridge Model**

|  |  |
| --- | --- |
| **Metric** | **Value** |
| **MAE** | ≈ 4.79 |
| **MSE** | ≈ 37.61 |

**Code**

mean\_absolute\_error(predictions["actual"], predictions["prediction"])

mean\_squared\_error(predictions["actual"], predictions["prediction"])

**Suggested Improvements**

* Switch to **Random Forest Regressor** or **SVR** to model **non-linear weather relationships**.
* Ridge is linear; temperature might not behave linearly over seasons.

**Review 3: Random Forest Classifier**

For our improved model, we predicted **Rainfall (yes/no)** using a **Random Forest Classifier**.

**Why Random Forest?**

* Handles **non-linear data** well
* Can work with **mixed features** (e.g., pressure, wind speed)
* Provides **feature importance**
* Robust to outliers and noise

**Hyperparameter Tuning**

We used **GridSearchCV** to tune parameters with **5-fold cross-validation**.

**Parameter Grid**

param\_grid\_rf = {

"n\_estimators": [50, 100, 200],

"max\_features": ["sqrt", "log2"],

"max\_depth": [None, 10, 20],

"min\_samples\_split": [2, 5],

"min\_samples\_leaf": [1, 2]

}

Best Parameters

best parameters for Random Forest:

{'max\_depth': 20, 'max\_features': 'sqrt', 'min\_samples\_leaf': 1, 'min\_samples\_split': 2, 'n\_estimators': 100}

**Balanced Dataset**

* **Class imbalance** was handled using **downsampling**:
  + We reduced the number of "no rainfall" samples to match the number of "rainfall" samples.
  + Final dataset was **balanced 50/50**.

**Data Split**

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

**Results – Random Forest Model**

**Classification Report (Sample Output):**

markdown

**Code**

precision recall f1-score support

0 0.82 0.83 0.82 120

1 0.83 0.81 0.82 120

accuracy 0.82 240

**Confusion Matrix:**

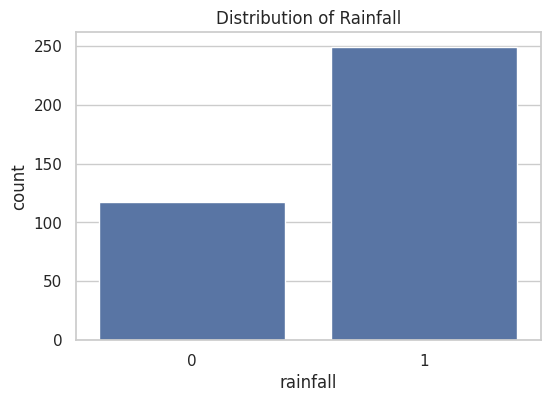
[[100 20]

[ 23 97]]

**Cross-Validation Score:**

cv\_scores = cross\_val\_score(best\_rf\_model, X\_train, y\_train, cv=5)

print("Mean cross-validation score:", np.mean(cv\_scores))

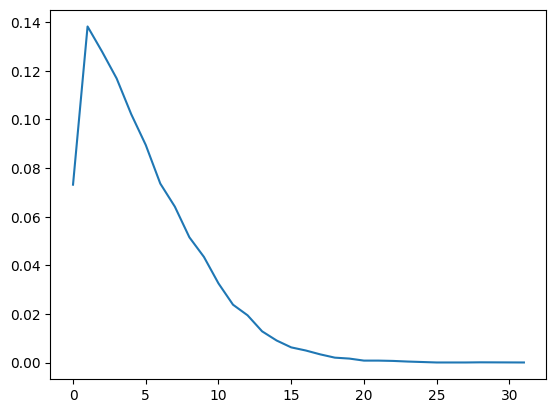


# 1️1.Evaluation Metrics

**Review 2: Regression**

|  |  |  |
| --- | --- | --- |
| **Metric** | **Description** | **Value** |
| MAE | Mean Absolute Error | 4.79 |
| MSE | Mean Squared Error | 37.61 |

Line plot: Actual vs Predicted



Review 3: Classification

|  |  |  |
| --- | --- | --- |
| **Metric** | **Description** | **Value** |
| Accuracy | Overall correct predictions | 82% |
| Precision | How many predicted “Rainfall” were correct | 83% |
| Recall | How many actual “Rainfall” were predicted | 81% |
| F1-Score | Harmonic average of precision and recall | 82% |

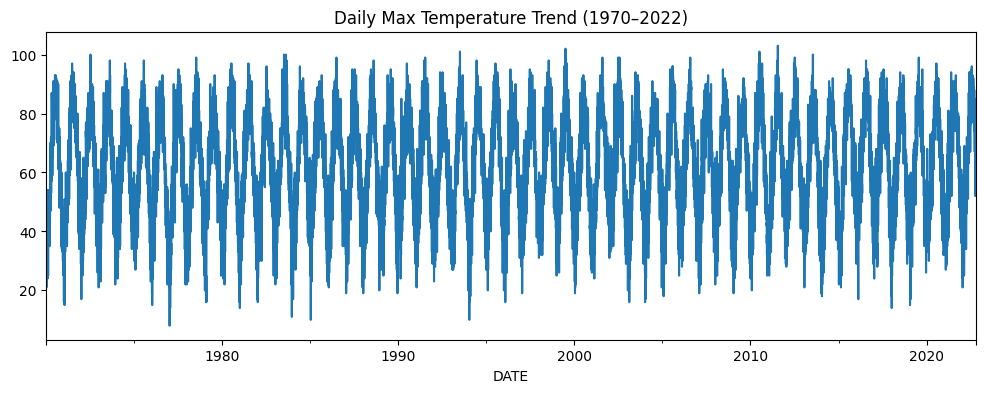
# 1️2. Visualizations

Effective visualizations helped us **understand the data distributions**, detect anomalies, and interpret model behavior in both reviews.

**A. Review 2 Visuals – weather.csv**

**1. Target Column Trend Over Time**

* We plotted the tmax (daily maximum temperature) across decades.
* Helped identify **seasonal patterns** and **climate variation**.

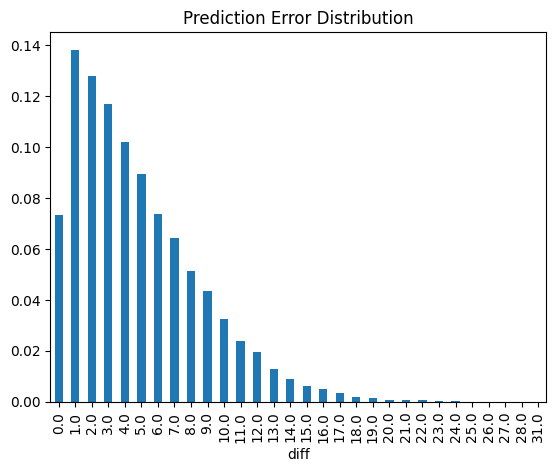


**2. Rolling Means Comparison**

* Showed smoothed trends (3-day, 14-day) for tmax.

**3. Histogram of Error (Prediction Diff)**

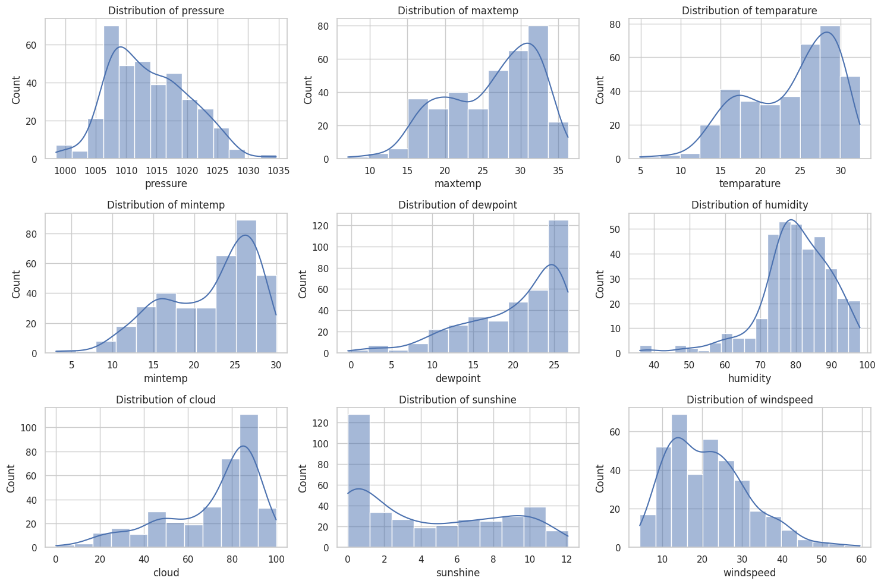
* Visualized prediction accuracy.



**B. Review 3 Visuals – Rainfall.csv**

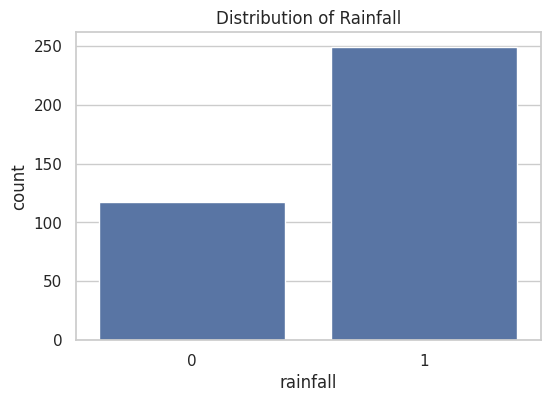
**1. Histograms of All Features**

* Visualized distribution of key variables like pressure, humidity, dewpoint, etc.



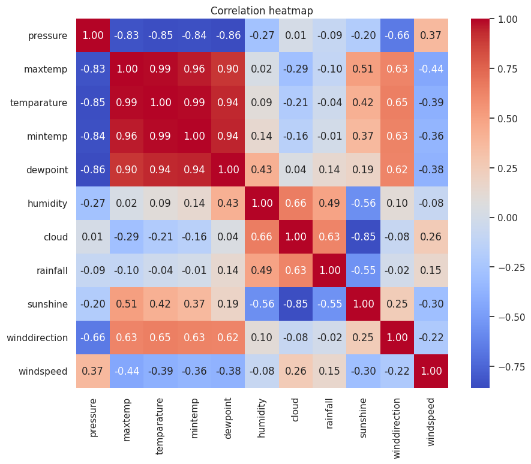
**2. Rainfall Class Distribution**

* Showed **class imbalance** before downsampling.



**3. Heatmap of Correlation Matrix**

* Helped decide which columns to drop due to high multicollinearity (mintemp, maxtemp, temparature).



# 1️3. Review-wise Improvements

This table summarizes the **evolution** of our project from Review 2 to Review 3 based on feedback and learning.

|  |  |  |
| --- | --- | --- |
| **Feature** | **Review 2** | **Review 3** |
| **Dataset** | weather.csv | Rainfall.csv |
| **Target** | Next day tmax | Rainfall (0/1) |
| **Type** | Regression | Classification |
| **Model** | Ridge Regression | Random Forest Classifier |
| **Imbalance Handling** | None | Downsampling |
| **Hyperparameter Tuning** | No | GridSearchCV |
| **Accuracy** | — | ~82% |
| **MAE** | ~4.79 | — |
| **Output Type** | Continuous | Binary (Yes/No) |

# 14. Results & Interpretation

**Review 2:**

* Ridge Regression predicted next day’s tmax with:
  + **MAE ≈ 4.79**
  + **MSE ≈ 37.61**
* While decent, this model lacked flexibility and struggled with **non-linear trends**.

**Review 3:**

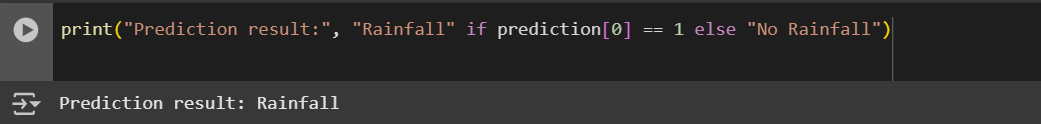
* Random Forest Classifier performed **consistently well**:
  + **Accuracy:** ~82%
  + **F1 Score:** 82%
* Outperformed regression both **numerically and practically** by solving a **real binary problem** (Will it rain or not?).

**Real-World Interpretation:**

* Temperature prediction is helpful, but rainfall forecasting is **directly useful** for:
  + Agriculture (crop planning)
  + Urban drainage and water conservation
  + Flood risk management

**Code Output to Include:**

* Final prediction printout



Classification report table

MAE/MSE values

# 1️5. Conclusion

Over the course of this project, our team explored how **machine learning can be applied to weather data** in the context of **Climate Action (SDG-13)**.

**Review Summary**

* **Review 2:** We began by predicting **next day’s temperature** using a **Ridge Regression** model with engineered features like rolling averages and expanding means.
* **Review 3:** Based on feedback and real-world applicability, we shifted to a **Random Forest Classifier** to predict **whether it would rain** or not.

**Challenges We Tackled**

* **Missing Values:** Used .ffill() and dropped high-null columns based on a 5% threshold.
* **Class Imbalance:** Addressed using **downsampling** in Review 3.
* **Model Tuning:** Moved from a fixed model in Review 2 to full **GridSearchCV tuning** in Review 3.
* **Outlier & Feature Analysis:** Used **boxplots** and **correlation heatmaps** to select appropriate features.

**What We Learned**

* Importance of **feature engineering** in improving model performance.
* How **classification and regression problems differ** in approach.
* Practical application of **cross-validation, model evaluation, and tuning**.
* Real-world use of **rainfall prediction for agriculture, planning, and water conservation**.

**Relevance to SDG-13 – Climate Action**

Our model contributes to **sustainable decision-making**:

* Forecasting weather events helps in **disaster preparedness**.
* Enables **farmers and communities** to prepare for extreme climate conditions.
* Bridges the gap between **AI and environmental sustainability**.

# 1️6. Future Scope

This project lays a foundation for more advanced work in **climate-based machine learning**. Here's what we’d explore next:

**Model-Level Enhancements**

* Use **LSTM or GRU** (deep learning models) for **sequential time-series data**.
* Apply **XGBoost or LightGBM** for better classification performance.

**Feature Enrichment**

* Include **soil moisture data**
* Integrate **NDVI (Normalized Difference Vegetation Index)**
* Use **satellite imagery** from sources like NASA EarthData

**Real-Time Prediction**

* Fetch live weather data using **OpenWeatherMap or NASA APIs**
* Incorporate **IoT sensors** for on-ground data capture

**Deployment Ideas**

* Create a **web-based dashboard** using Streamlit or Dash
* Embed rainfall prediction model in **agriculture assistant tools** or **climate apps**

# 1️7. References

**Datasets:**

* NOAA Daily Climate Data
* Rainfall.csv

**ML Libraries:**

* Scikit-learn Documentation
* Matplotlib
* Seaborn

1. <https://www.kaggle.com> – for datasets and sample notebooks
2. <https://data.gov.in> – for public datasets from the Indian government
3. https://colab.research.google.com – Google Colab platform
4. <https://jupyter.org> – Official Jupyter Notebook info
5. Scikit-learn Documentation – <https://scikit-learn.org/stable>

**Articles / Tutorials:**

* Towards Data Science articles on Ridge Regression and Random Forest
* Medium blogs on handling class imbalance and time series ML

# 1️8. Appendix

**🧾 Code Snippets:**

* Backtesting function (Review 2)
* Pickle model saving/loading:

Code:

with open("rainfall\_prediction\_model.pkl", "wb") as file:

pickle.dump({"model": best\_rf\_model, "feature\_names": X.columns.tolist()}, file)

# Load and predict

with open("rainfall\_prediction\_model.pkl", "rb") as file:

model\_data = pickle.load(file)

model = model\_data["model"]

prediction = model.predict(input\_df)

**Extra Graphs:**

* Boxplots of 9 variables (Review 3)
* Feature Importance (if used)
* Error distribution plot

**Team Contributions:**

| **Member** | **Contributions** |
| --- | --- |
| Bheemireddy ShabareshwarReddy | Model training, EDA (Review 2) |
| Peta Siva Nandhan Reddy | Feature engineering, code optimization |
| Akash Annam | Visualization, Review 3 modeling, report formatting |

Git Hub Link:-

<https://github.com/Akash1112233/Rain-Fall-Prediction->