



# ForecastIQ: Applying Regression based Machine Learning for accurate Weather Predictions

Presented by: Rohan Seetepalli

# Overview

- Introduction
- Data- Preprocessing
- Exploratory Data Analysis(EDA)
- Analysis and Model Fitting
- Real time Prediction
- Summary
- Future Scope

The top corners of the slide are decorated with stylized, wavy lines in shades of light orange and brown, resembling clouds or abstract patterns.

# Introduction






**Weather prediction is a crucial aspect of environmental monitoring, helping societies prepare for daily conditions and extreme events. Among various parameters, precipitation forecasting is particularly important due to its impact on agriculture, transportation, and public safety.**

**This project focuses on predicting daily precipitation levels using regression techniques based on historical weather data which includes various key features like wind, weather-type, maximum and minimum temperature and date of rainfall. By applying advanced regression models, the goal is to achieve accurate and real-time precipitation forecasts for improved weather analysis and decision-making.**

# Data Pre-processing

# Dataset

This dataset was obtained from Kaggle. The various features are as follows:-

-  **Date:** Indicates the specific calendar day of the weather observation.
-  **Precipitation:** Measures the amount of rainfall received on a given day, typically in millimeters.
-  **Temp-Max:** The highest temperature recorded during the day.
-  **Temp-Min:** The lowest temperature recorded during the day.
-  **Weather:** Describes the general atmospheric condition (e.g., rain, snow, sun, fog, drizzle) observed on that day.

# Data Cleaning and Feature Engineering

The various pre-processing steps are as follows:-

- Date was split into day, month and year as separate columns.
- The column “Weather” was one-hot encoded into weather\_rain, weather\_snow, weather\_sun, weather\_drizzle, weather\_fog and later weather\_drizzle was dropped.
- Two new features- “avg\_temp”-(average of the columns- temp\_max and temp\_min) and temp\_range- (difference of temp\_max and temp-min) were created and temp\_max and temp\_min were dropped.

```
df= pd.read_csv('seattle-weather.csv')
df
```

	date	precipitation	temp_max	temp_min	wind	weather
0	2012-01-01	0.0	12.8	5.0	4.7	drizzle
1	2012-01-02	10.9	10.6	2.8	4.5	rain
2	2012-01-03	0.8	11.7	7.2	2.3	rain
3	2012-01-04	20.3	12.2	5.6	4.7	rain
4	2012-01-05	1.3	8.9	2.8	6.1	rain
...	...	...	...	...	...	...
1456	2015-12-27	8.6	4.4	1.7	2.9	rain
1457	2015-12-28	1.5	5.0	1.7	1.3	rain
1458	2015-12-29	0.0	7.2	0.6	2.6	fog
1459	2015-12-30	0.0	5.6	-1.0	3.4	sun

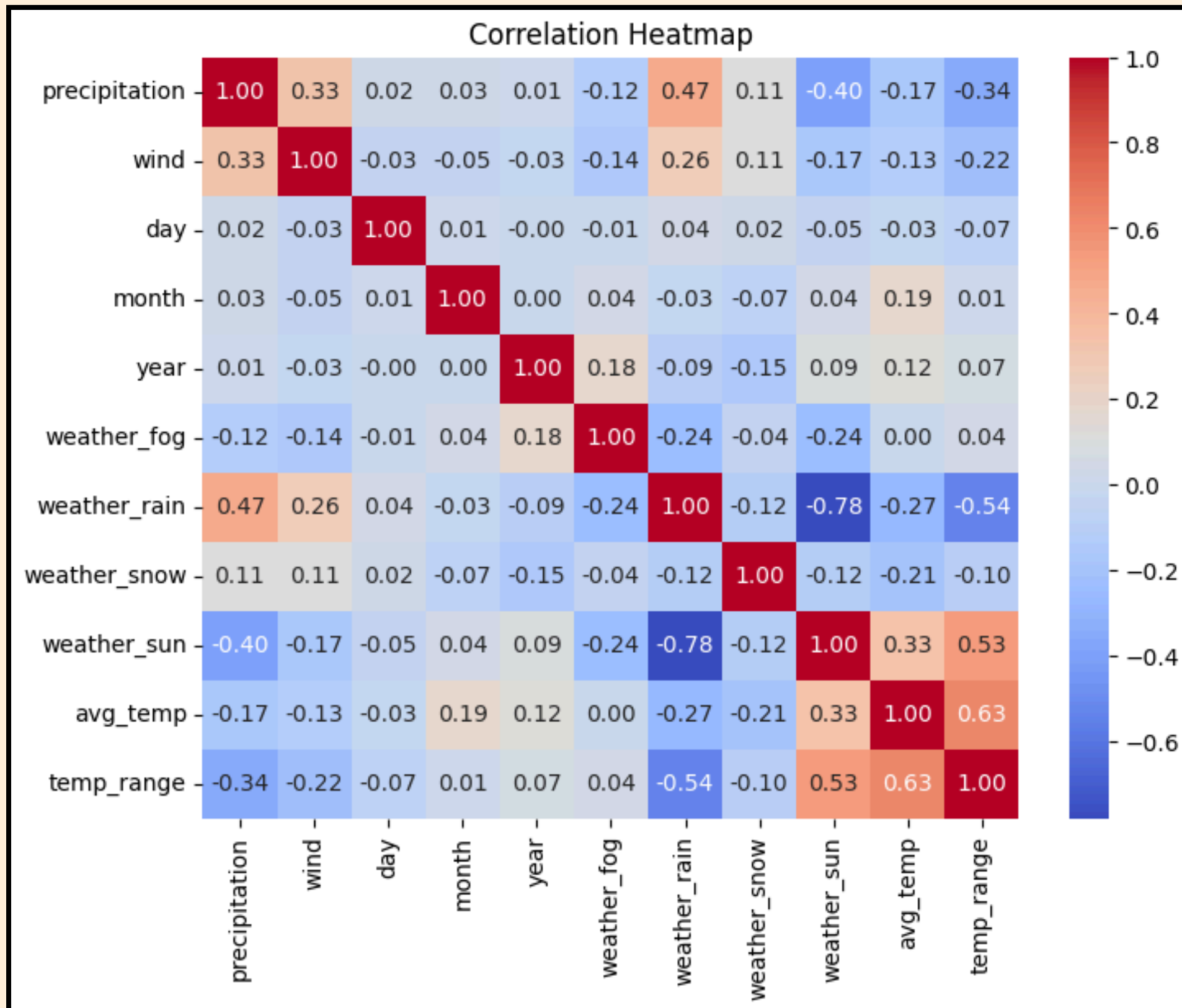
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1461 entries, 0 to 1460
Data columns (total 8 columns):
#   Column          Non-Null Count  Dtype
---  -
0   precipitation    1461 non-null   float64
1   temp_max        1461 non-null   float64
2   temp_min        1461 non-null   float64
3   wind            1461 non-null   float64
4   weather         1461 non-null   object
5   day             1461 non-null   int32
6   month           1461 non-null   int32
7   year            1461 non-null   int32
dtypes: float64(4), int32(3), object(1)
memory usage: 74.3+ KB
```

# Exploratory Data Analysis

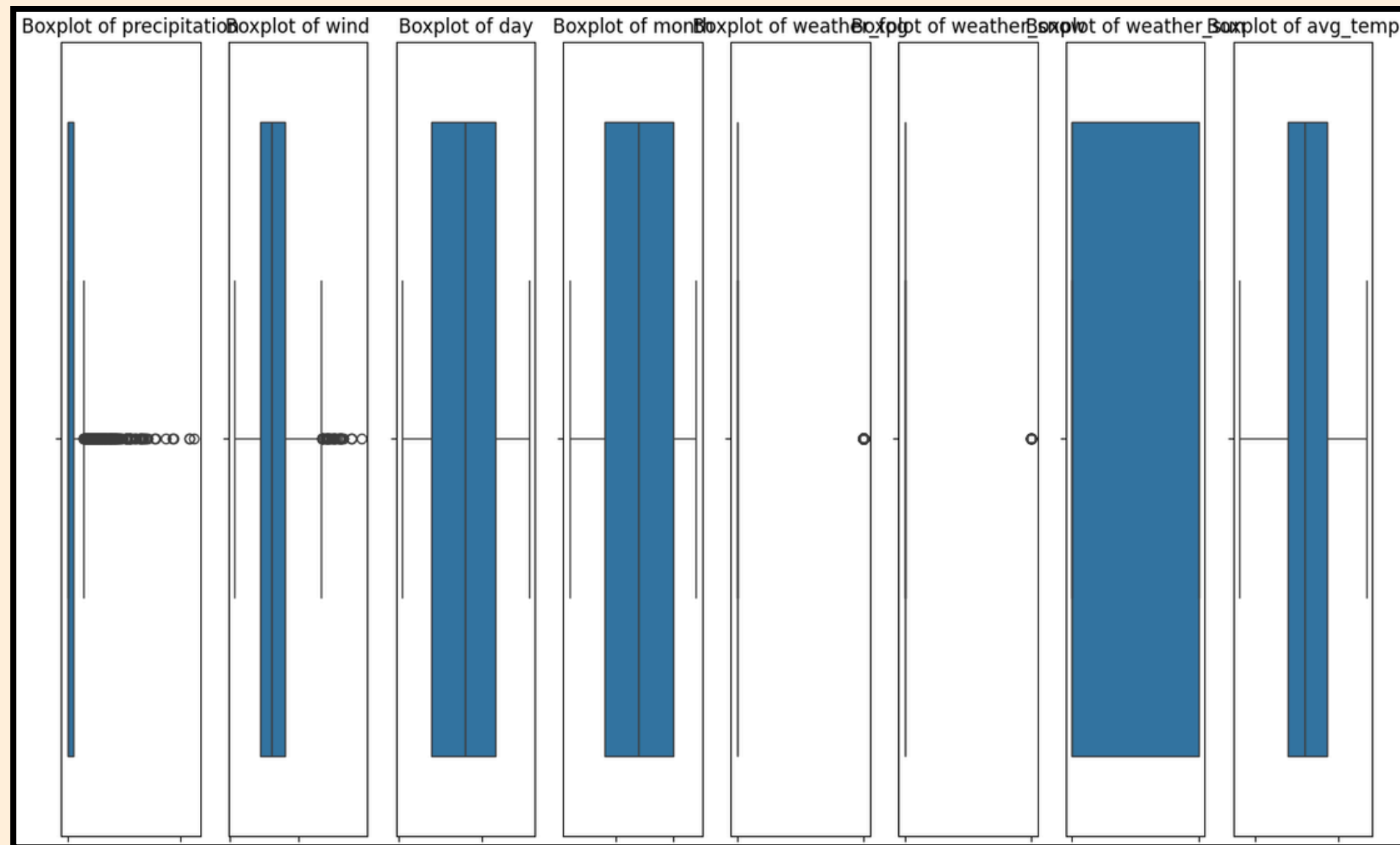


# Correlation Heatmap



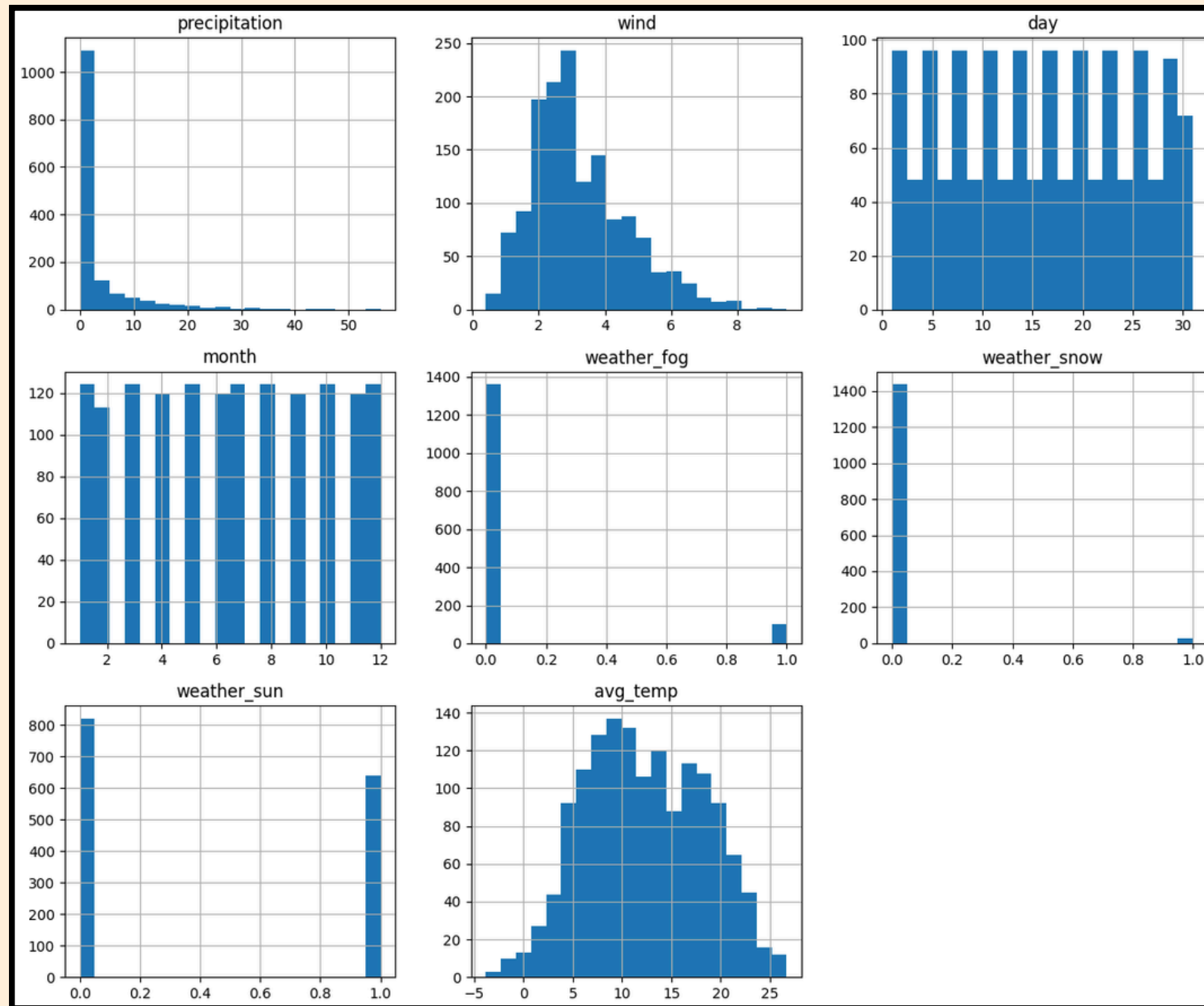
A graphical representation that shows the strength and direction of relationships between numerical variables using color-coded cells.

# Box-Plot



A graphical representation of the distribution of a numerical variable that displays the median, quartiles, and potential outliers, helping identify variability and skewness in the data.

# Histogram



A bar graph that displays the distribution of a numerical variable by grouping data into ranges (bins).

# Multi-Collinearity Check

# Variance Inflation Factor

	Feature	VIF
0	precipitation	1.701864
1	wind	7.180201
2	day	4.231927
3	month	4.883435
4	year	47.744256
5	weather_fog	2.918321
6	weather_rain	14.242188
7	weather_snow	1.595989
8	weather_sun	13.376715
9	avg_temp	9.561812
10	temp_range	12.984167

Variance Inflation Factor (VIF) is a metric that quantifies how much a feature is correlated with other features, helping detect multicollinearity in regression models.

Due to High VIF, the columns- year, temp\_range and weather\_rain were removed.

# Analysis

# Algorithms

**Decision Trees**: A tree-based model that splits data into branches based on feature values to make predictions.

**XGBoost**: An optimized gradient boosting algorithm that builds a series of decision trees to improve prediction accuracy.

**SVR (Support Vector Regression)**: A regression model that uses the principles of Support Vector Machines to find a function within a margin of tolerance.

**ANN (Artificial Neural Network)**: A computational model inspired by the human brain that uses interconnected layers of nodes to learn patterns in data.

**Random Forests**: An ensemble method that builds multiple decision trees and combines their outputs for more robust predictions.

# Metrics

**Mean Squared Error (MAE):** Mean Squared Error is the average of the absolute differences between the actual and predicted values in a regression model.

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$$

**R Squared ( $R^2$ ):** R Squared measures the proportion of the variance in the dependent variable that is explained by the independent variables in the model.

$$R^2 = 1 - \frac{\sum (y_i - \hat{y}_i)^2}{\sum (y_i - \bar{y})^2}$$



# Mean- Absolute Error

<b>Splits</b> <b>Algorithms</b>	<b>80-20 SPLIT</b>	<b>75-25 SPLIT</b>	<b>70-30 SPLIT</b>	<b>60-40 SPLIT</b>
<b>Decision Trees</b>	<b>2.86</b>	<b>2.69</b>	<b>2.72</b>	<b>2.87</b>
<b>XG Boost</b>	<b>2.92</b>	<b>2.74</b>	<b>2.92</b>	<b>2.95</b>
<b>ANN</b>	<b>3.06</b>	<b>2.77</b>	<b>2.98</b>	<b>2.87</b>
<b>Random Forests</b>	<b>2.72</b>	<b>2.74</b>	<b>2.85</b>	<b>2.90</b>
<b>SVR</b>	<b>2.35</b>	<b>2.33</b>	<b>2.47</b>	<b>2.54</b>

# R-Squared

<b>Splits</b> <b>Algorithms</b>	<b>80-20 SPLIT</b>	<b>75-25 SPLIT</b>	<b>70-30 SPLIT</b>	<b>60-40 SPLIT</b>
<b>Decision Trees</b>	<b>0.16</b>	<b>0.22</b>	<b>0.31</b>	<b>0.25</b>
<b>XG Boost</b>	<b>0.23</b>	<b>0.31</b>	<b>0.26</b>	<b>0.25</b>
<b>ANN</b>	<b>0.22</b>	<b>0.26</b>	<b>0.27</b>	<b>0.29</b>
<b>Random Forests</b>	<b>0.27</b>	<b>0.28</b>	<b>0.26</b>	<b>0.26</b>
<b>SVR</b>	<b>0.22</b>	<b>0.22</b>	<b>0.20</b>	<b>0.18</b>

# Real-time Prediction

# Real-time Prediction

The project includes a real-time prediction feature that allows users to input current weather data and instantly receive a precipitation forecast.

- The model takes inputs such as date, maximum temperature, minimum temperature, and weather condition.
- These inputs are processed through the trained XGBoost model to generate a precipitation prediction.
- The system provides fast and practical forecasts, useful for daily planning and weather-based decision-making.

## ▼ REAL TIME PREDICTION

```
import numpy as np
import pandas as pd
from xgboost import XGBRegressor

def real_time_weather_prediction(input_features, model):
    """
    Predicts precipitation (continuous value) using a pre-trained XGBoost regression model.

    Parameters:
    | input_features (list): A list of feature values in the following order:
    | | [wind, weather_fog, weather_snow, weather_sun, avg_temp]
    | model (XGBRegressor): Pre-trained XGBoost model.

    Returns:
    | | float: Predicted precipitation in millimeters (mm).
    """

    # Define the feature columns (order must match the model's expected feature order)
    columns = ['wind', 'weather_fog', 'weather_snow', 'weather_sun', 'avg_temp']

    # Convert the input to a DataFrame
    input_df = pd.DataFrame([input_features], columns=columns)

    # Get the predicted precipitation from the model
    prediction = model.predict(input_df)[0]

    return prediction

# === Example Usage ===
# Assuming your model is already trained and ready:
xgb_model = XGBRegressor(random_state=42)
xgb_model.fit(X_train_2, y_train_2)

# === Real-time Input ===
new_weather_input = [3.4, 0, 0, 1, 40] # Example: 12 km/h wind, snow, no fog, no sun, 20.5°C avg temperature

# === Get Real-Time Prediction ===
predicted_precipitation = real_time_weather_prediction(new_weather_input, xgb_model)
print(f"☁ Predicted Precipitation: {predicted_precipitation:.2f} mm")
```

☁ Predicted Precipitation: -0.18 mm

# Summary



# Summary

**This project aimed to predict daily precipitation using regression and utilizing various Machine Learning and Deep Learning models by taking a dataset based on weather data. The XGBoost model, trained with a 75-25 split, was found to be the most optimal, making it ideal for real-time prediction scenarios.**

# Future Scope





# Future Scope

**Feature Expansion**: Incorporating additional variables like humidity, wind speed, and atmospheric pressure for improved prediction accuracy.

**Model Deployment**: Developing a real-time application (web or mobile) to provide daily precipitation forecasts to users.

**Real-Time Data Integration**: Integrating live weather data from APIs to enable dynamic and automated predictions for continuous forecasting.



Thank You