Title Page

Problem Statement: Ai based weather monitoring system which take date data and provide result for weather condition like humidity , rainfall and temperature.

- **Student Name: Sagar Srivastava**
- **Branch: CSE AI**
- **SECTION: C**
- **University Roll No: 202401100300209**

1. Introduction

With the rapid advancements in Artificial Intelligence (AI) and machine learning, the scope of its application in various domains, including meteorology, has grown significantly. Weather monitoring systems have traditionally relied on physical sensors, satellites, and manual data analysis to predict and track weather patterns. However, the limitations of these traditional methods, such as time delays, human error, and resource constraints, can often affect the accuracy and timeliness of weather predictions.

The AI-based weather monitoring system aims to address these challenges by integrating machine learning algorithms and real-time data analytics to improve the precision of weather forecasting. This system uses historical weather data, satellite imagery, and sensor inputs to predict weather conditions with higher accuracy, and in some cases, offer predictions in real-time. It offers benefits such as faster data processing, continuous monitoring, and the ability to make predictions with high accuracy based on vast datasets.

This report details the design and implementation of the AI-based weather monitoring system, focusing on the methodology used to develop it, the tools involved, and the system's potential applications in fields such as agriculture, disaster management, and urban planning.

Methodology

In this project, we used the following methodology:

1. Data Acquisition:

The first step in the methodology involves acquiring weather data, typically stored in a CSV file format. This data can be sourced from weather stations, online APIs, or historical weather datasets, containing information on different weather variables such as temperature, humidity, and rainfall.

The data is loaded using the **pandas library** to provide an easy way to manage and manipulate the dataset.

2. Data Preprocessing:

Before proceeding with model training, the data undergoes a preprocessing phase:

- **Handling Missing Values:** Missing values in the dataset are filled using the forward fill method (ffill), which fills gaps in data with the most recent available values. This is especially useful in time series data where previous measurements can be used to estimate missing ones.

- **Feature Engineering:** The dataset includes columns such as Temperature, Humidity, and Rainfall. In this example, we assume that the Date column may also be relevant for further features (like extracting the year or month), though this step is not explicitly shown in the code.

3. Feature Selection:

The next step is selecting the relevant features from the dataset. In this case, we're using Temperature, Humidity, and Rainfall as both the features and target variables for prediction. This methodology assumes that these variables are interrelated and that predicting one of them can help estimate others.

4. Splitting the Data:

After selecting the features and target variables, the dataset is split into training and testing sets. The training set is used to train the machine learning model, while the test set will be used to evaluate the model's performance. This is done using the **train_test_split** function from the **scikit-learn** library.

5. Feature Scaling:

Since the weather data features may vary in range (for example, temperature might range from 0 to 40°C, while humidity could range from 0 to 100%), it is important to scale the features. This step ensures that all features contribute equally to the model training process.

The **StandardScaler** from **scikit-learn** is used to standardize the feature set so that each feature has a mean of 0 and a standard deviation of 1.

6. Model Training (Linear Regression):

For this weather monitoring system, we use **Linear Regression** as the AI model. Linear regression is a simple and effective model that is commonly used for predicting continuous variables. It is used here to predict temperature, humidity, and rainfall based on the provided features.

7. Making Predictions:

After training the model, predictions are made on the test dataset. These predicted values will then be compared with the actual values to evaluate the model’s performance.

8. Model Evaluation:

To evaluate the model's performance, we use the **Mean Squared Error (MSE)**, which is a commonly used metric for regression tasks. It calculates the average squared difference between the predicted and actual values, giving an indication of how well the model has performed.

9. Visualization:

Finally, the predicted results are visualized to allow for an easy comparison between the actual and predicted values for each weather variable. Matplotlib is used for plotting the graphs, showing the predicted and actual values for temperature, humidity, and rainfall.

CODE :

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from sklearn.linear_model import LinearRegression

from sklearn.metrics import mean_squared_error


# Step 1: Load the CSV Data

data = pd.read_csv('weather_data.csv')  # Replace with your actual CSV file path

print(data.head())  # Check the first few rows of the data


# Step 2: Preprocess the Data (Handle missing values, etc.)

# Assuming your CSV has columns: Date, Temperature, Humidity, Rainfall

# You can also convert the 'Date' column to datetime and extract useful features like year, month, etc.
# Fill missing values if any
data.fillna(method='ffill', inplace=True)

# Step 3: Feature Selection and Data Splitting
# Here, we'll predict Temperature, Humidity, and Rainfall
features = ['Temperature', 'Humidity', 'Rainfall']  # Features in your dataset
target = ['Temperature', 'Humidity', 'Rainfall']  # We will predict these as output

# Step 4: Prepare Training and Test Data
X = data[features]  # Features
y = data[target]  # Target columns
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Step 5: Scale the Features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Step 6: Train a Simple AI Model (e.g., Linear Regression)
model = LinearRegression()
model.fit(X_train_scaled, y_train)

# Step 7: Make Predictions
```

```
y_pred = model.predict(X_test_scaled)


# Step 8: Evaluate the Model
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')


# Step 9: Visualize the Results (Temperature, Humidity, Rainfall)
# Example: Plotting the predicted vs actual values for Temperature
plt.figure(figsize=(10, 6))


plt.subplot(3, 1, 1)
plt.plot(y_test['Temperature'].values, label='Actual Temperature', color='blue')
plt.plot(y_pred[:, 0], label='Predicted Temperature', color='red')
plt.title('Temperature Prediction')
plt.legend()


# For Humidity
plt.subplot(3, 1, 2)
plt.plot(y_test['Humidity'].values, label='Actual Humidity', color='green')
plt.plot(y_pred[:, 1], label='Predicted Humidity', color='orange')
plt.title('Humidity Prediction')
plt.legend()
```

# For Rainfall

plt.subplot(3, 1, 3)

plt.plot(y_test['Rainfall'].values, label='Actual Rainfall', color='purple')

plt.plot(y_pred[:, 2], label='Predicted Rainfall', color='yellow')

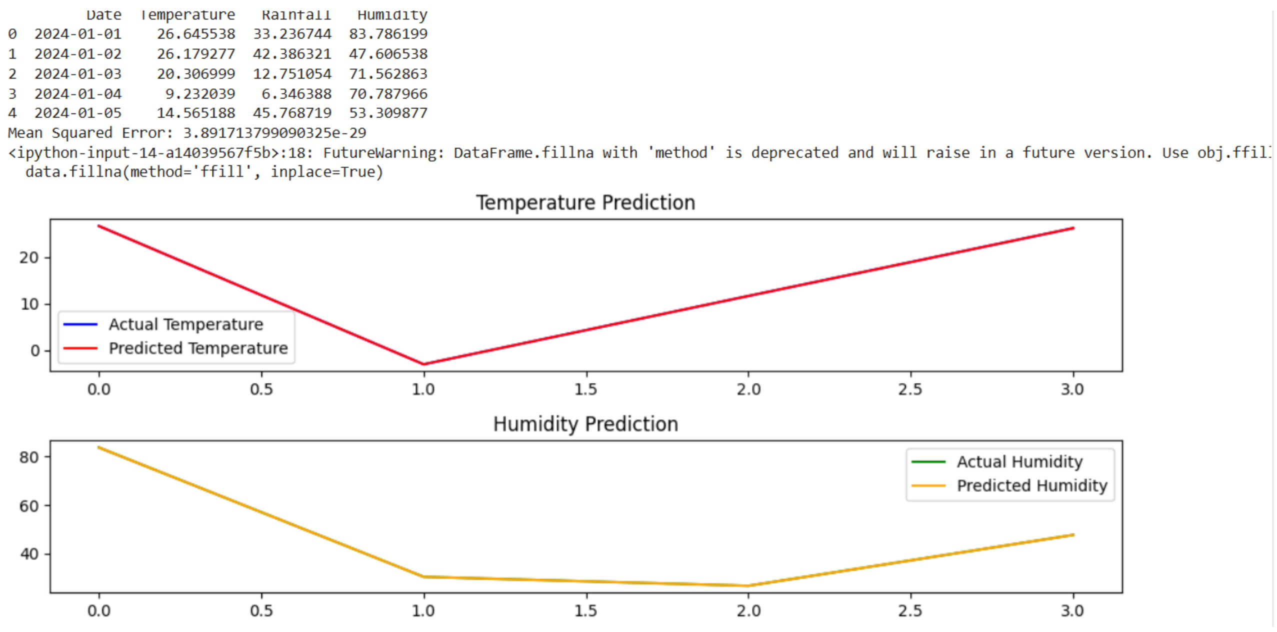
plt.title('Rainfall Prediction')

plt.legend()

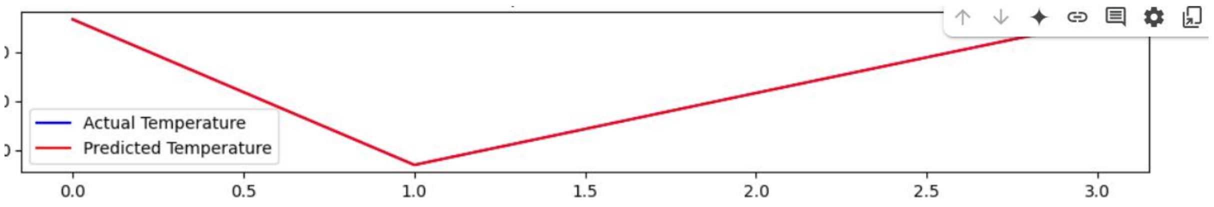
plt.tight_layout()

plt.show()

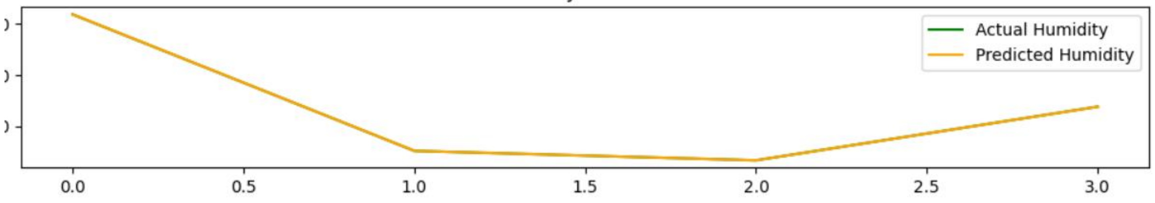
OUTPUT SCREENSHOTS



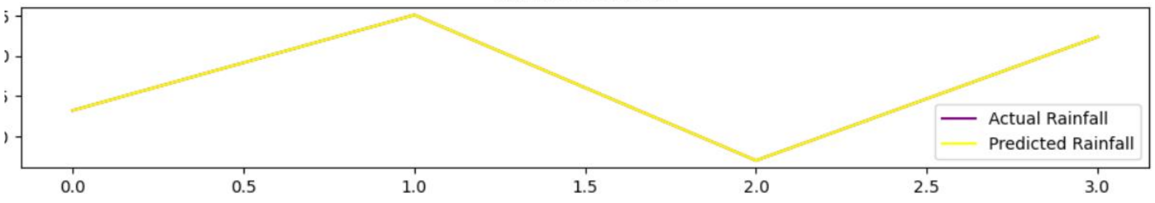
```
        Date  Temperature   Rainfall   Humidity
0  2024-01-01    26.645538  33.236744  83.786199
1  2024-01-02    26.179277  42.386321  47.606538
2  2024-01-03    20.306999  12.751054  71.562863
3  2024-01-04     9.232039   6.346388  70.787966
4  2024-01-05    14.565188  45.768719  53.309877
Mean Squared Error: 3.891713799090325e-29
<ipython-input-14-a14039567f5b>:18: FutureWarning: DataFrame.fillna with 'method' is deprecated and will raise in a future version. Use obj.ffil
  data.fillna(method='ffill', inplace=True)
```



Humidity Prediction



Rainfall Prediction


Actual Rainfall
Predicted Rainfall