Weather Prediction Notebook Report

1. Introduction

This report provides a step-by-step analysis of the weather prediction notebook. It details data loading, preprocessing, exploratory data analysis (EDA), visualization, and model training procedures.

2. Data Loading

The dataset is loaded using pandas from a CSV file named weather\_data.csv.

The dataset is stored in a DataFrame for further processing.

df = pd.read\_csv("/content/weather\_data.csv")

3. Libraries Used

The following libraries are utilized in the notebook:

pandas: Data manipulation.

matplotlib.pyplot & seaborn: Data visualization.

statsmodels.tsa.seasonal: Time series decomposition.

sklearn.linear\_model (LinearRegression): Predictive modeling.

numpy: Numerical operations.

4. Data Preprocessing

The notebook does not include explicit data cleaning steps, such as handling missing values (fillna), dropping columns (drop), or converting data types (astype).

5. Exploratory Data Analysis (EDA) and Visualization

Several EDA techniques are used to understand the dataset:

5.1 Time Series Analysis

A 7-day moving average is calculated and plotted to observe temperature trends over time.

plt.figure(figsize=(10, 5))  
daily\_avg\_temp.plot(label='Daily Avg Temp', alpha=0.5)  
daily\_avg\_temp.rolling(window=7).mean().plot(label='7-Day Moving Avg', linestyle='dashed', color='red')  
plt.xlabel('Date')  
plt.ylabel('Temperature (°C)')  
plt.title('Temperature Trends Over Time with Moving Average')  
plt.legend()  
plt.show()

5.2 Correlation Analysis (Temperature vs. Humidity)

A scatter plot is created to analyze the relationship between temperature and humidity, with a regression trend line.

from scipy.stats import linregress  
plt.figure(figsize=(8, 5))  
sns.scatterplot(x=df['Humidity\_pct'], y=df['Temperature\_C'], alpha=0.5)  
slope, intercept, \_, \_, \_ = linregress(df['Humidity\_pct'], df['Temperature\_C'])  
plt.plot(df['Humidity\_pct'], slope \* df['Humidity\_pct'] + intercept, color='red', linestyle='dashed', label='Trend Line')  
plt.xlabel('Humidity (%)')  
plt.ylabel('Temperature (°C)')  
plt.title('Temperature vs Humidity with Trend Line')  
plt.legend()  
plt.show()

5.3 Location-Based Temperature Analysis

The top 10 locations with the highest average temperature are identified and visualized using a bar chart.

avg\_temp = df.groupby('Location')['Temperature\_C'].mean().nlargest(10)  
plt.figure(figsize=(8, 5))  
avg\_temp.plot(kind='bar', color='skyblue')  
plt.xlabel('Location')  
plt.ylabel('Average Temperature (°C)')  
plt.title('Top 10 Locations with Highest Avg Temperature')  
plt.xticks(rotation=45)  
plt.show()

6. Model Training (Linear Regression)

The notebook imports LinearRegression but does not explicitly fit a model.

No .fit() function is used to train the model, nor is there any evaluation of its performance using metrics like r2\_score.

from sklearn.linear\_model import LinearRegression

7. Missing Components

The notebook lacks the following essential steps:

Data Preprocessing: Handling missing values, feature selection, and scaling.

Model Training: Using LinearRegression().fit(X\_train, y\_train).

Model Evaluation: Using metrics such as r2\_score or mean\_squared\_error.

Predictions: Applying .predict() on new data.

8. Recommendations for Improvement

Preprocess Data: Handle missing values and normalize numerical features.

Define Features (X) and Target (y): Select relevant columns for training.

Train Model: Fit LinearRegression and evaluate its performance.

Make Predictions: Use .predict() to forecast weather conditions.

Improve Visualization: Include heatmaps and pair plots for deeper insights.

9. Conclusion

The notebook primarily focuses on data visualization and trend analysis but does not implement a complete predictive modeling pipeline. Adding preprocessing, model training, and evaluation steps would enhance its effectiveness for weather prediction.