```
# Import Libraries
import pandas as pd
import numpy as np
import random
from datetime import datetime, timedelta
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor  # Using Random Forest instead of Linear Regression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.preprocessing import OneHotEncoder


# Set Random Seeds for Reproducibility
random.seed(42)
np.random.seed(42)


# Generate Realistic Synthetic Weather Data
locations = ["New York", "Los Angeles", "Chicago", "Houston", "Phoenix", "San Diego", "Dallas", "Philadelphia"]
num_samples = 1000



# Create Date and Location Data
data = {
    "Location": [random.choice(locations) for _ in range(num_samples)],
    "Date_Time": [(datetime(2024, 1, 1) + timedelta(days=random.randint(0, 365), hours=random.randint(0, 23), minutes=random.randint(0, 59))).strftime('%d-%m-%Y %H:%M') for
}

df = pd.DataFrame(data)


# Add Seasonal Temperature Patterns
def seasonal_temp(month):
    if month in [12, 1, 2]:  # Winter
        return np.random.uniform(-10, 10)
    elif month in [3, 4, 5]:  # Spring
        return np.random.uniform(5, 20)
    elif month in [6, 7, 8]:  # Summer
        return np.random.uniform(15, 40)
    else:  # Fall
        return np.random.uniform(5, 25)



# Generate Weather Features
df["Date_Time"] = pd.to_datetime(df["Date_Time"], format="%d-%m-%Y %H:%M")
df["Month"] = df["Date_Time"].dt.month
df["Hour"] = df["Date_Time"].dt.hour
df["Temperature_C"] = df["Month"].apply(seasonal_temp)
df["Humidity_pct"] = df["Temperature_C"].apply(lambda x: np.random.uniform(30, 70) if x > 20 else np.random.uniform(60, 100))
df["Precipitation_mm"] = df["Month"].apply(lambda m: np.random.uniform(0, 10) if m in [6, 7, 8] else np.random.uniform(5, 20))
df["Wind_Speed_kmh"] = np.random.uniform(5, 30, num_samples).round(2)
```

```
# One-Hot Encoding for Locations
encoder = OneHotEncoder(sparse=False)
encoded_locations = encoder.fit_transform(df[["Location"]])
location_df = pd.DataFrame(encoded_locations, columns=encoder.get_feature_names_out(["Location"]))
df = pd.concat([df.drop("Location", axis=1), location_df], axis=1)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-f4b3e1759367> in <cell line: 0>()
      1 # One-Hot Encoding for Locations
----> 2 encoder = OneHotEncoder(sparse=False)
      3 encoded_locations = encoder.fit_transform(df[["Location"]])
      4 location_df = pd.DataFrame(encoded_locations, columns=encoder.get_feature_names_out(["Location"]))
      5 df = pd.concat([df.drop("Location", axis=1), location_df], axis=1)

NameError: name 'OneHotEncoder' is not defined
```

```
# One-Hot Encoding for Locations
encoder = OneHotEncoder(handle_unknown='ignore') # remove sparse argument
encoded_locations = encoder.fit_transform(df[["Location"]]).toarray() # convert sparse matrix to dense array
location_df = pd.DataFrame(encoded_locations, columns=encoder.get_feature_names_out(["Location"]))
df = pd.concat([df.drop("Location", axis=1), location_df], axis=1)


# Define Features and Targets
X = df.drop(columns=["Temperature_C", "Humidity_pct", "Precipitation_mm", "Date_Time"])
y_temp = df["Temperature_C"]
y_humidity = df["Humidity_pct"]
y_precipitation = df["Precipitation_mm"]


# Train-Test Split
X_train, X_test, y_temp_train, y_temp_test = train_test_split(X, y_temp, test_size=0.2, random_state=42)
_, _, y_humidity_train, y_humidity_test = train_test_split(X, y_humidity, test_size=0.2, random_state=42)
_, _, y_precip_train, y_precip_test = train_test_split(X, y_precipitation, test_size=0.2, random_state=42)

# Train Models using Random Forest
models = {}
for target, y_train in zip(["Temperature", "Humidity", "Precipitation"], [y_temp_train, y_humidity_train, y_precip_train]):
    model = RandomForestRegressor(n_estimators=100, random_state=42)
    model.fit(X_train, y_train)
    models[target] = model


# Predictions
y_temp_pred = models["Temperature"].predict(X_test)
y_humidity_pred = models["Humidity"].predict(X_test)
y_precip_pred = models["Precipitation"].predict(X_test)



# Evaluate Models
for target, y_test, y_pred in zip(["Temperature", "Humidity", "Precipitation"], [y_temp_test, y_humidity_test, y_precip_test], [y_temp_pred, y_humidity_pred, y_precip_pred])
```

```
    print(f"Model Performance for {target}:")
    print(f"Mean Absolute Error (MAE): {mean_absolute_error(y_test, y_pred):.2f}")
    print(f"Mean Squared Error (MSE): {mean_squared_error(y_test, y_pred):.2f}")
    print(f"R² Score (Accuracy): {r2_score(y_test, y_pred):.2f}")
    print("=" * 40)
```

```
Model Performance for Temperature:
Mean Absolute Error (MAE): 5.45
Mean Squared Error (MSE): 41.84
R² Score (Accuracy): 0.66
========================================
Model Performance for Humidity:
Mean Absolute Error (MAE): 12.75
Mean Squared Error (MSE): 244.35
R² Score (Accuracy): 0.19
========================================
Model Performance for Precipitation:
Mean Absolute Error (MAE): 3.80
Mean Squared Error (MSE): 20.94
R² Score (Accuracy): 0.26
========================================
```

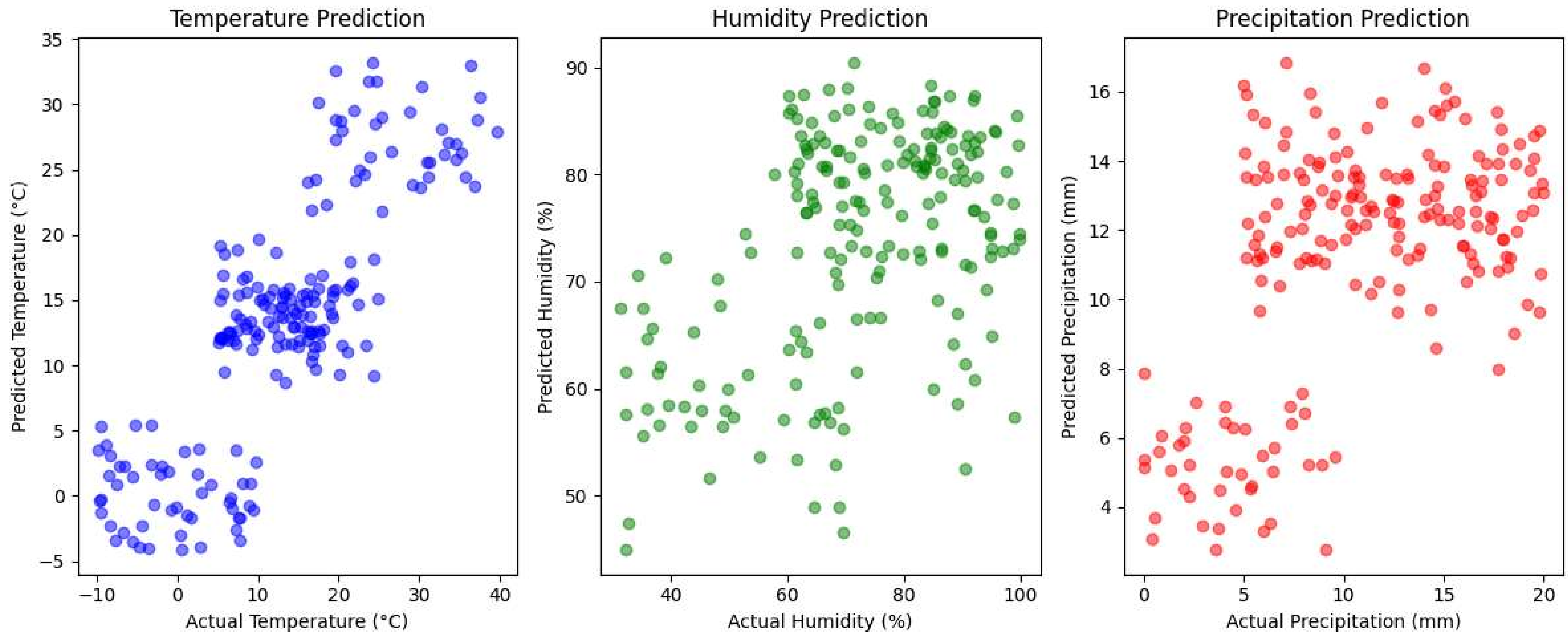
```
# Visualization
plt.figure(figsize=(12, 5))

plt.subplot(1, 3, 1)
plt.scatter(y_temp_test, y_temp_pred, alpha=0.5, color="blue")
plt.xlabel("Actual Temperature (°C)")
plt.ylabel("Predicted Temperature (°C)")
plt.title("Temperature Prediction")

plt.subplot(1, 3, 2)
plt.scatter(y_humidity_test, y_humidity_pred, alpha=0.5, color="green")
plt.xlabel("Actual Humidity (%)")
plt.ylabel("Predicted Humidity (%)")
plt.title("Humidity Prediction")

plt.subplot(1, 3, 3)
plt.scatter(y_precip_test, y_precip_pred, alpha=0.5, color="red")
plt.xlabel("Actual Precipitation (mm)")
plt.ylabel("Predicted Precipitation (mm)")
plt.title("Precipitation Prediction")

plt.tight_layout()
plt.show()
```


Temperature Prediction
Predicted Temperature (°C)
Actual Temperature (°C)
Humidity Prediction
Predicted Humidity (%)
Actual Humidity (%)
Precipitation Prediction
Predicted Precipitation (mm)
Actual Precipitation (mm)