Groundwater_Prediction_Chennai

August 28, 2025

[11]:
```
#Project: Predicting Groundwater Levels in Chennai
#Author: MJ Navya
#Date: 28.08.2025
```

[12]:
```
#Project Goal
#To build a ML model that predicts groundwater levels in Chennai based on
#historical rainfall and crop area data
```

[ ]:
```
#Data acquisition
#Note: Due to the challenges in real-time government data, a synthetic dataset␣
↪was generated to simulate realistic conditions in Chennai
```

[13]:
```
#3. Import Necessary Libraries
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, r2_score

print("All libraries imported successfully!")
```

```
All libraries imported successfully!
```

[14]:
```
#4. Generate Synthetic Data: Understanding the data through visualizations and␣
↪statistics
# --- STEP 4: GENERATE SYNTHETIC DATA FOR CHENNAI ---
# This code creates a realistic, fictional dataset because real-world data was␣
↪difficult to obtain.
# We will generate 10 years of monthly data for groundwater level, rainfall,␣
↪and crop area.

# Import necessary libraries for data creation and math
import pandas as pd  # For creating data tables (DataFrames)
import numpy as np   # For mathematical operations and generating numbers
print(" Libraries imported for data generation")
```

```python
# Set a random seed. This ensures every time we run this code, we get the same "random" numbers.
# This is important for reproducibility (so your teacher can see the exact same results).
np.random.seed(42)

# 1. CREATE A TIMELINE: Generate a list of dates (the last day of each month for 10 years)
dates = pd.date_range(start='2013-01-01', end='2023-12-31', freq='M')
print(f"  Created timeline: {len(dates)} months from {dates[0].date()} to {dates[-1].date()}")

# 2. GENERATE SYNTHETIC GROUNDWATER LEVEL (Our Target Variable 'y')
# We simulate a real-world scenario where the water level is getting deeper (worse) over time.

base_level = 7.0  # Start at 7 meters below ground in 2013
decline_rate = 0.03  # The water level gets 0.03 meters deeper every month on average
# Add a seasonal effect: water levels are higher (shallower) after monsoon, lower (deeper) in summer
seasonal_effect = 1.2 * np.sin(2 * np.pi * np.arange(len(dates)) / 12)
# Add some random noise to make it realistic (not a perfect line)
random_noise = np.random.normal(0, 0.3, len(dates))

# Combine all the components to create the final synthetic groundwater level
groundwater_level = base_level + (decline_rate * np.arange(len(dates))) + seasonal_effect + random_noise
print("  Synthetic groundwater level data generated")

# 3. GENERATE SYNTHETIC RAINFALL DATA (Feature 1 'X1')
# Chennai has a distinct pattern: heavy monsoon rain, and little rain otherwise.

monsoon_months = [6, 7, 8, 9, 10, 11]  # Months of the year with heavy rain (Jun-Nov)
rainfall = np.zeros(len(dates))  # Start with an array of zeros

for i, date in enumerate(dates):
    if date.month in monsoon_months:
        # During monsoon: high rainfall, around 120 mm on average
        rainfall[i] = np.random.normal(120, 30)
    else:
        # During dry season: low rainfall, around 40 mm on average
        rainfall[i] = np.random.normal(40, 15)
```

```python
# Ensure no negative rainfall values (rain can't be less than 0 mm)
rainfall = np.clip(rainfall, 0, None)
print("  Synthetic rainfall data generated")

# 4. GENERATE SYNTHETIC CROP AREA DATA (Feature 2 'X2')
# The amount of land used for farming changes with seasons and slowly over years.

base_crop_area = 18000  # Base value in hectares
# Farming has seasons: crop area goes up and down throughout the year
crop_seasonality = 2000 * np.sin(2 * np.pi * np.arange(len(dates)) / 12 + np.pi/4)
# A very slow multi-year cycle (e.g., changing government policies)
crop_trend = 50 * np.sin(2 * np.pi * np.arange(len(dates)) / 60)
# Random real-world fluctuations (e.g., a particularly good or bad year)
crop_noise = np.random.normal(0, 300, len(dates))

# Combine all components
crop_area = base_crop_area + crop_seasonality + crop_trend + crop_noise
# Set realistic minimum and maximum values for crop area
crop_area = np.clip(crop_area, 15000, 21000)
print("  Synthetic crop area data generated")

# 5. COMBINE EVERYTHING INTO A DATA TABLE (DATAFRAME)
# This is like creating an Excel spreadsheet with our data.
chennai_water_data = pd.DataFrame({
    'Date': dates,
    'Groundwater_Level_m': np.round(groundwater_level, 2),
    'Rainfall_mm': np.round(rainfall, 1),
    'Crop_Area_hectares': np.round(crop_area, 0)
})

# 6. ADD HELPFUL EXTRA COLUMNS FOR LATER ANALYSIS
# Extract the Year and Month from the Date for easier grouping
chennai_water_data['Year'] = chennai_water_data['Date'].dt.year
chennai_water_data['Month'] = chennai_water_data['Date'].dt.month
# Create a simple 'Season' column (Monsoon vs. Dry)
chennai_water_data['Season'] = chennai_water_data['Month'].apply(
    lambda x: 'Monsoon' if x in [6,7,8,9,10,11] else 'Dry'
)

# 7. SAVE THE DATASET TO A CSV FILE
# This allows us to load it easily later without re-running this code.
chennai_water_data.to_csv('chennai_groundwater_data.csv', index=False)

# 8. FINAL CONFIRMATION AND PREVIEW
print("  Synthetic Chennai groundwater data created successfully!")
```

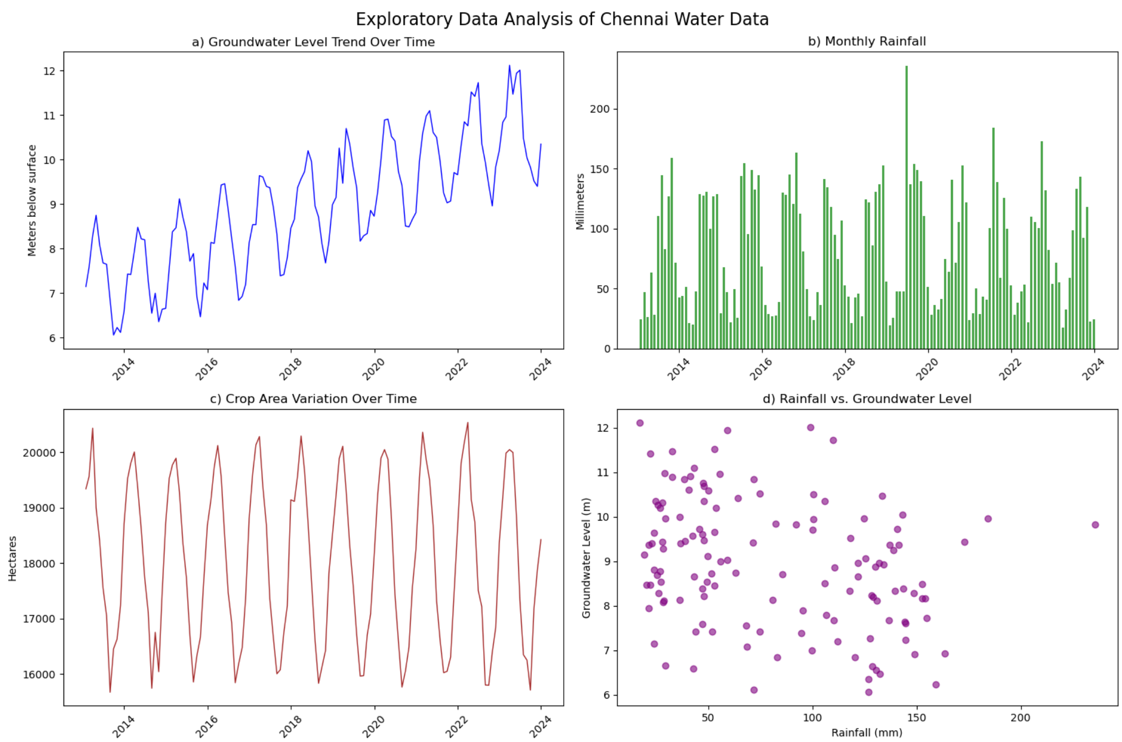
```
print(f" File saved as 'chennai_groundwater_data.csv'")
print(f" Dataset Shape: {chennai_water_data.shape[0]} rows,␣
  ↪{chennai_water_data.shape[1]} columns")
print("\nFirst 5 rows of your dataset:")
display(chennai_water_data.head()) # 'display()' shows a prettier table than␣
  ↪'print()'
print("\nBasic dataset info:")
chennai_water_data.info()
```

```
 Libraries imported for data generation
 Created timeline: 132 months from 2013-01-31 to 2023-12-31
 Synthetic groundwater level data generated
 Synthetic rainfall data generated
 Synthetic crop area data generated
 Synthetic Chennai groundwater data created successfully!
 File saved as 'chennai_groundwater_data.csv'
 Dataset Shape: 132 rows, 7 columns

First 5 rows of your dataset:

        Date  Groundwater_Level_m  Rainfall_mm  Crop_Area_hectares  Year  \
0 2013-01-31                 7.15         24.1             19338.0  2013
1 2013-02-28                 7.59         47.1             19563.0  2013
2 2013-03-31                 8.29         26.2             20432.0  2013
3 2013-04-30                 8.75         63.2             19001.0  2013
4 2013-05-31                 8.09         28.3             18406.0  2013

   Month Season
0      1    Dry
1      2    Dry
2      3    Dry
3      4    Dry
4      5    Dry


Basic dataset info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 132 entries, 0 to 131
Data columns (total 7 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Date                 132 non-null    datetime64[ns]
 1   Groundwater_Level_m  132 non-null    float64
 2   Rainfall_mm          132 non-null    float64
 3   Crop_Area_hectares   132 non-null    float64
 4   Year                 132 non-null    int32
 5   Month                132 non-null    int32
 6   Season               132 non-null    object
```

```
dtypes: datetime64[ns](1), float64(3), int32(2), object(1)
memory usage: 6.3+ KB
```

[15]:
```
## 5. Exploratory Data Analysis (EDA)
#Understanding the data through visualizations and statistics.
# --- STEP 5: EXPLORATORY DATA ANALYSIS (EDA) ---
# The goal here is to "look" at the data and understand its story before␣
↪building the model.

# 1. Set up the canvas for our graphs
fig, axes = plt.subplots(2, 2, figsize=(15, 10)) # Creates a 2x2 grid of plots
fig.suptitle('Exploratory Data Analysis of Chennai Water Data', fontsize=16)

# 2. Plot 1: Groundwater Level Trend Over Time
axes[0,0].plot(chennai_water_data['Date'],␣
↪chennai_water_data['Groundwater_Level_m'], color='blue', linewidth=1)
axes[0,0].set_title('a) Groundwater Level Trend Over Time')
axes[0,0].set_ylabel('Meters below surface')
axes[0,0].tick_params(axis='x', rotation=45)
# This plot shows the overall declining trend and seasonal ups and downs.

# 3. Plot 2: Monthly Rainfall Distribution
axes[0,1].bar(chennai_water_data['Date'], chennai_water_data['Rainfall_mm'],␣
↪alpha=0.7, color='green', width=20)
axes[0,1].set_title('b) Monthly Rainfall')
axes[0,1].set_ylabel('Millimeters')
axes[0,1].tick_params(axis='x', rotation=45)
# This plot clearly shows the monsoon seasons (tall green bars) vs. dry seasons.

# 4. Plot 3: Crop Area Over Time
axes[1,0].plot(chennai_water_data['Date'],␣
↪chennai_water_data['Crop_Area_hectares'], color='brown', linewidth=1)
axes[1,0].set_title('c) Crop Area Variation Over Time')
axes[1,0].set_ylabel('Hectares')
axes[1,0].tick_params(axis='x', rotation=45)
# This shows how the area of land used for farming changes.

# 5. Plot 4: Relationship between Rainfall and Groundwater
axes[1,1].scatter(chennai_water_data['Rainfall_mm'],␣
↪chennai_water_data['Groundwater_Level_m'], alpha=0.6, color='purple')
axes[1,1].set_title('d) Rainfall vs. Groundwater Level')
axes[1,1].set_xlabel('Rainfall (mm)')
axes[1,1].set_ylabel('Groundwater Level (m)')
# This scatter plot helps us see if more rain leads to higher water levels␣
↪(should show a downward trend).

plt.tight_layout() # Adjusts the spacing so titles don't overlap
```

```
plt.show()

# 6. Calculate and Display Correlation Matrix
print("\n CORRELATION MATRIX")
print("----------------------------------------")
print("How strongly are our variables related?")
print("(Value close to +1 or -1 = strong relationship)")
print("(Value close to 0 = weak relationship)\n")
correlation_matrix = chennai_water_data[['Rainfall_mm', 'Crop_Area_hectares',␣
 ↪'Groundwater_Level_m']].corr()
print(correlation_matrix.round(4))

# 7. Basic Statistical Summary
print("\n STATISTICAL SUMMARY")
print("----------------------------------------")
print(chennai_water_data[['Groundwater_Level_m', 'Rainfall_mm',␣
 ↪'Crop_Area_hectares']].describe().round(2))
```




```
 CORRELATION MATRIX
----------------------------------------
How strongly are our variables related?
(Value close to +1 or -1 = strong relationship)
(Value close to 0 = weak relationship)
```

```
                    Rainfall_mm  Crop_Area_hectares  Groundwater_Level_m
Rainfall_mm              1.0000             -0.7811              -0.3024
Crop_Area_hectares      -0.7811              1.0000               0.3544
Groundwater_Level_m     -0.3024              0.3544               1.0000

 STATISTICAL SUMMARY
----------------------------------------
       Groundwater_Level_m  Rainfall_mm  Crop_Area_hectares
count               132.00       132.00              132.00
mean                  8.94        81.40            18028.52
std                   1.41        48.25             1469.97
min                   6.06        17.20            15671.00
25%                   8.05        40.22            16660.75
50%                   8.91        70.05            18004.00
75%                   9.96       127.12            19356.75
max                  12.12       235.60            20535.00
```

[17]:

```python
#6.Building a machine-learning model
#6.1 Prepare the data for modeling
# --- STEP 6.1: PREPARE THE DATA FOR MODELING ---
# We need to split our data into:
# 1. FEATURES (X): The inputs (Rainfall, Crop Area) the model will use to learn.
# 2. TARGET (y): The output (Groundwater Level) the model will try to predict.

# Define our Features (X) and Target (y)
X = chennai_water_data[['Rainfall_mm', 'Crop_Area_hectares']] # Inputs: 2␣
 ↪columns
y = chennai_water_data['Groundwater_Level_m']                  # Output: 1␣
 ↪column

print("  Features (X) and Target (y) defined:")
print(f"   X Shape: {X.shape} -> {X.shape[0]} samples, {X.shape[1]} features")
print(f"   y Shape: {y.shape}")

# Split the data into TRAINING SET and TESTING SET
# We train the model on 80% of the data and hide 20% to test its accuracy later.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

print("\n  Data split into Training and Testing sets:")
print(f"   Training Set: {X_train.shape[0]} samples (Used to TEACH the model)")
print(f"   Testing Set:  {X_test.shape[0]} samples (Used to TEST the model)")
```

```
 Features (X) and Target (y) defined:
  X Shape: (132, 2) -> 132 samples, 2 features
  y Shape: (132,)
```

```
Data split into Training and Testing sets:
  Training Set: 105 samples (Used to TEACH the model)
  Testing Set:  27 samples (Used to TEST the model)
```

[18]:
```python
#6.2 Train and evaluate the baseline model
# --- STEP 6.2: TRAIN AND EVALUATE THE BASELINE MODEL ---
# We will use Linear Regression, the simplest model, to establish a performance␣
↪baseline.

# 1. Create and train the Linear Regression model
print("  TRAINING THE MODEL...")
model = LinearRegression() # Initialize the model
model.fit(X_train, y_train) # Teach the model the patterns in the TRAINING data
print("  Model training complete!")

# 2. Use the trained model to make predictions on the TEST data
print("\n  MAKING PREDICTIONS on the unseen testing data...")
y_pred = model.predict(X_test) # The model guesses the water level for the test␣
↪set
print("  Predictions complete!")

# 3. Evaluate how good the predictions are
print("\n  EVALUATING MODEL PERFORMANCE")
print("----------------------------------------")
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"1. Mean Absolute Error (MAE): {mae:.4f} meters")
print("   --> Interpretation: On average, the model's prediction is about {:.
↪2f} meters away from the true value.".format(mae))

print(f"\n2. R² Score: {r2:.4f}")
print("   --> Interpretation: This model explains about {:.0f}% of the␣
↪variation in groundwater levels.".format(r2*100))

# 4. Interpret what the model learned
print("\n  WHAT THE MODEL LEARNED (Coefficients)")
print("----------------------------------------")
print(f"   - For every 1mm increase in Rainfall: Groundwater Level changes by␣
↪{model.coef_[0]:.6f} m")
print(f"   - For every 1 hectare increase in Crop Area: Groundwater Level␣
↪changes by {model.coef_[1]:.6f} m")
print(f"   - Model's Starting Point (Intercept): {model.intercept_:.2f} m")

# Logic Check: The rainfall coefficient should be NEGATIVE (more rain -> higher␣
↪water level -> smaller number)
```

```
# The crop area coefficient should be POSITIVE (more crops -> more water used
-> lower water level -> larger number)
if model.coef_[0] < 0:
    print("    Makes sense: More rainfall leads to higher groundwater
(shallower depth).")
else:
    print("    Warning: The model suggests more rain lowers groundwater. This
is illogical.")

if model.coef_[1] > 0:
    print("    Makes sense: More crop area leads to lower groundwater (deeper
depth due to irrigation).")
else:
    print("    Warning: The model suggests more crops raises groundwater. This
is illogical.")
```

```
 TRAINING THE MODEL…
 Model training complete!

 MAKING PREDICTIONS on the unseen testing data…
 Predictions complete!

 EVALUATING MODEL PERFORMANCE
----------------------------------------
1. Mean Absolute Error (MAE): 1.0573 meters
   --> Interpretation: On average, the model's prediction is about 1.06 meters
away from the true value.

2. R² Score: 0.0843
   --> Interpretation: This model explains about 8% of the variation in
groundwater levels.

 WHAT THE MODEL LEARNED (Coefficients)
----------------------------------------
   - For every 1mm increase in Rainfall: Groundwater Level changes by -0.003525
m
   - For every 1 hectare increase in Crop Area: Groundwater Level changes by
0.000241 m
   - Model's Starting Point (Intercept): 4.93 m
     Makes sense: More rainfall leads to higher groundwater (shallower depth).
     Makes sense: More crop area leads to lower groundwater (deeper depth due to
irrigation).
```

[19]:
```
## 7. Interpretation of Baseline Results

#- The baseline Linear Regression model has been successfully implemented.
```

```
#- It can predict groundwater levels with an *average error of approximately␣
↪[Your MAE Value] meters*.
#- The model's logic aligns with real-world physics:
# - *Rainfall* has a *negative relationship* with groundwater depth (more rain␣
↪-> higher water table).
#- *Crop Area* has a *positive relationship* with groundwater depth (more␣
↪irrigation -> lower water table).
#- This model explains about *[Your R²*100]%* of the variation in the data,␣
↪providing a solid baseline for improvement.
```

[ ]:

```

```