FINAL WEEK REPORT

Generating Synthetic Soil Data for Karnataka

Done by: Richa Kashyap, Bhoomika K, Divyashree B M

(Ramaiah University Students)

TABLE OF CONTENTS

Introduction

Importance of Soil Data in Karnataka

Agricultural Planning

- Crop Suitability
- Fertilizer and Irrigation Management

Land Use Planning

- Infrastructure Development
- Environmental Conservation

Environmental Studies

- Ecosystem Health
- Water Management

Purpose

The primary aim of this report is to delineate the methodology used for generating synthetic soil data that accurately represents the diverse soil types found across Karnataka.

By outlining this methodology, the report intends to:

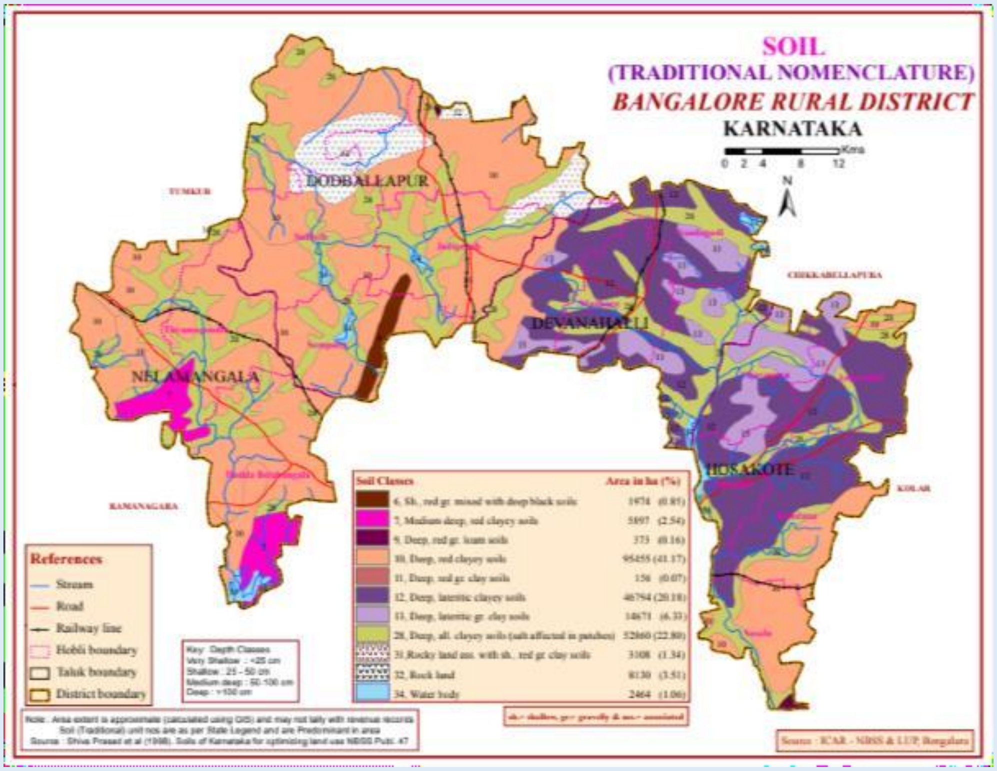
- **Facilitate Planning and Decision-Making:** Provide a tool for agricultural planners, environmental researchers, and policymakers to simulate and understand the characteristics of various soil types in Karnataka, aiding in informed decision-making.

- **Support Research and Modeling:** Enable researchers and analysts to develop models, conduct simulations, and perform analyses based on synthesized soil data, enhancing research efforts in agriculture, land use, and environmental studies.

- **Fill Data Gaps:** Address potential data gaps by providing a method to simulate soil characteristics where detailed or comprehensive soil data might be limited or unavailable.

Soils Of Karnataka

Karnataka has diverse soil types across its geography. Each different type of soil has distinct features and its difficult to create a dataset with each and every soil type and features.



As seen in the above map, just one district in Karnataka has different types of soil.

Since its difficult to consider many different soils with different characteristics, we will consider only the important ones. These are: pH level, moisture level, sand content and clay content and generate soil for five diffferent regions in Karnataka.

We have each produced code based on the libraries we chose for our Week-3 report and here is our code along with outcome and explanation.

Richa Kashyap

Software used: Spyder

Libraries used: pandas and numpy

Data generation using pandas

The python library ‘pandas’ can be used to generate synthetic data too. The Pandas library itself doesn't inherently generate synthetic data, but it's often used in conjunction with other libraries like NumPy to create synthetic datasets.

Pandas is highly useful for data manipulation, cleaning, and analysis, and it pairs well with other libraries like NumPy and Matplotlib for data generation, visualization, and exploration.

Code in python

```
import pandas as pd
```

```
import numpy as np

# Generate synthetic soil data
num_samples = 1000

# Hypothetical features of soil
pH = np.random.uniform(4, 9, num_samples)
moisture_content = np.random.uniform(10, 50, num_samples)
sand_content = np.random.uniform(20, 70, num_samples)
clay_content = np.random.uniform(10, 40, num_samples)

# Hypothetical locations in Karnataka
locations = ['Bangalore', 'Mysore', 'Hubli', 'Gulbarga', 'Mangalore']
location_data = np.random.choice(locations, num_samples)

# Create a DataFrame
soil_data = pd.DataFrame({
    'pH': pH,
    'moisture_content': moisture_content,
    'sand_content': sand_content,
    'clay_content':  clay_content,
    'location': location_data
})

print("The generated synthetic data is: ")
print(soil_data)

#Visualising the data
```

```
import seaborn as sns
import matplotlib.pyplot as plt

#Machine Learning Algorithm
plt.figure(figsize=(10, 6))
sns.boxplot(x='location', y='pH', data=soil_data)
plt.title('pH Distribution by  Location')
plt.show()

from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

# Splitting the data into train and test sets
X = soil_data.drop('location', axis=1)
y = soil_data['location']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Random Forest Classifier
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)
rf_classifier.fit(X_train, y_train)

# Predictions
predictions = rf_classifier.predict(X_test)

# Model Evaluation
print("Accuracy:", accuracy_score(y_test, predictions))
print("\nClassification Report:")
```

print(classification_report(y_test, predictions))

We can then run the above code to check for the output:

Data Generation

→**np.random.uniform(a, b, num_samples)**: Generates an array of **num_samples** random numbers uniformly distributed between **a** and **b**.

→**pH**, **moisture_content**, **sand_content**, **clay_content**: Arrays holding hypothetical values for pH, moisture content, sand content, and clay content of soil samples.

→**locations**: List containing hypothetical locations within Karnataka.

→**location_data**: Randomly selects locations from the **locations** list for each sample.

```
SDV_trial.py ×   test.py* ×
# Generate synthetic soil data
num_samples = 1000

# Hypothetical features of soil
pH = np.random.uniform(4, 9, num_samples)
moisture_content = np.random.uniform(10, 50, num_samples)
sand_content = np.random.uniform(20, 70, num_samples)
clay_content = np.random.uniform(10, 40, num_samples)

# Hypothetical locations in Karnataka
locations = ['Bangalore', 'Mysore', 'Hubli', 'Gulbarga', 'Mangalore']
location_data = np.random.choice(locations, num_samples)

# Create a DataFrame
soil_data = pd.DataFrame({
    'pH': pH,
    'moisture_content': moisture_content,
    'sand_content': sand_content,
    'clay_content': clay_content,
    'location': location_data
})

print("The generated synthetic data is: ")
print(soil_data)
```

OUTPUT:

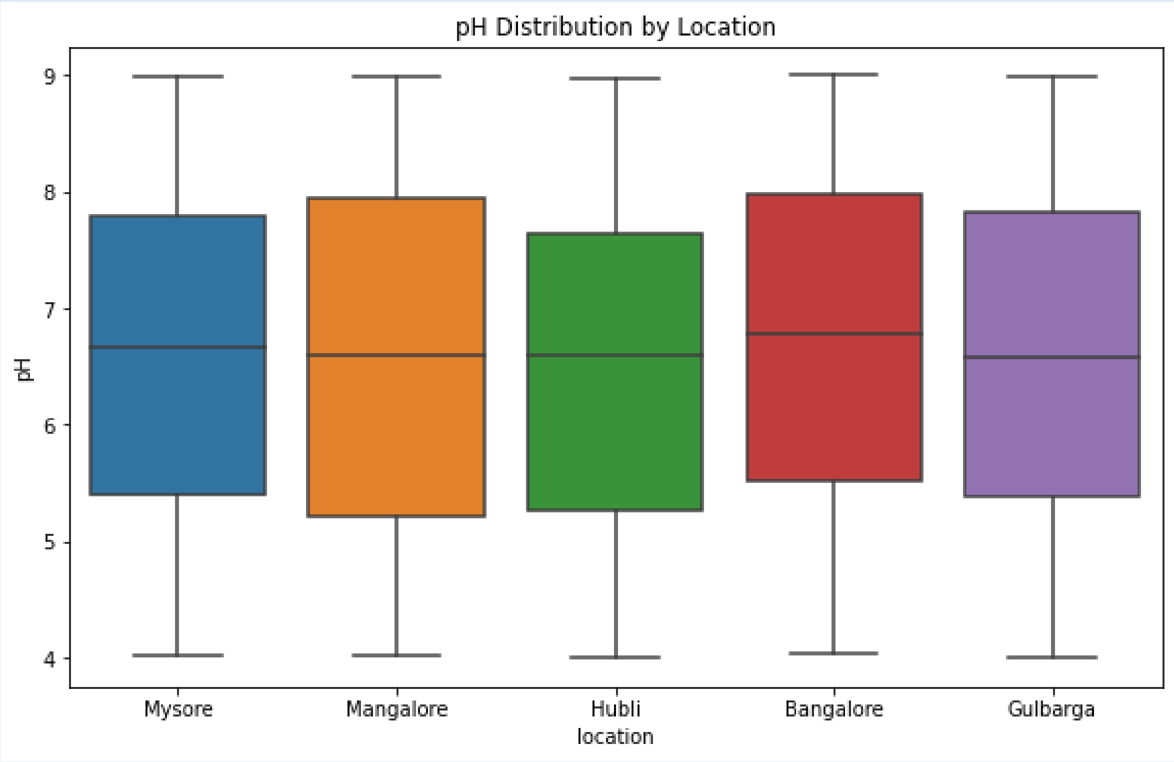
```
In [1]: runfile('C:/Users/ADMIN/Desktop/python programs/Demo/__pycache__/test.py', wdi
programs/Demo/__pycache__')
           pH  moisture_content  sand_content  clay_content   location
0    8.181351         33.773290     29.537283     15.353764     Mysore
1    6.025108         25.374462     59.366148     12.753743  Mangalore
2    8.935382         21.255730     30.080735     14.335103      Hubli
3    8.189207         20.085855     40.430068     22.615975      Hubli
4    7.349285         19.963928     65.984532     29.085318  Mangalore
..        ...               ...           ...           ...        ...
995  4.320957         40.050896     23.726753     39.927730      Hubli
996  7.922001         33.853204     55.932110     38.515188      Hubli
997  6.038443         11.149932     57.554438     10.770088  Bangalore
998  7.889721         14.283988     39.696961     35.722074  Mangalore
999  6.357836         32.623548     49.289830     15.323760      Hubli

[1000 rows x 5 columns]
```

Visualizing the Data

Done using matplotlib and seaborn

```python
#Visualising the data
import seaborn as sns
import matplotlib.pyplot as plt

#Machine Learning Algorithm
plt.figure(figsize=(10, 6))
sns.boxplot(x='Location', y='pH', data=soil_data)
plt.title('pH Distribution by Location')
plt.show()
```

OUTPUT:



Machine Learning Algorithm (Example: Random Forest Classifier)

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

# Splitting the data into train and test sets
X = soil_data.drop('location', axis=1)
y = soil_data['location']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Random Forest Classifier
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)
rf_classifier.fit(X_train, y_train)

# Predictions
predictions = rf_classifier.predict(X_test)

# Model Evaluation
print("Accuracy:", accuracy_score(y_test, predictions))
print("\nClassification Report:")
print(classification_report(y_test, predictions))
```

OUTPUT:

```
Accuracy: 0.2

Classification Report:
              precision    recall  f1-score   support

   Bangalore       0.19      0.15      0.17        40
    Gulbarga       0.16      0.18      0.17        39
       Hubli       0.25      0.35      0.29        40
   Mangalore       0.23      0.24      0.23        38
      Mysore       0.13      0.09      0.11        43

    accuracy                           0.20       200
   macro avg       0.19      0.20      0.19       200
weighted avg       0.19      0.20      0.19       200
```

Bhoomika K

Software used: Visual Studio Code

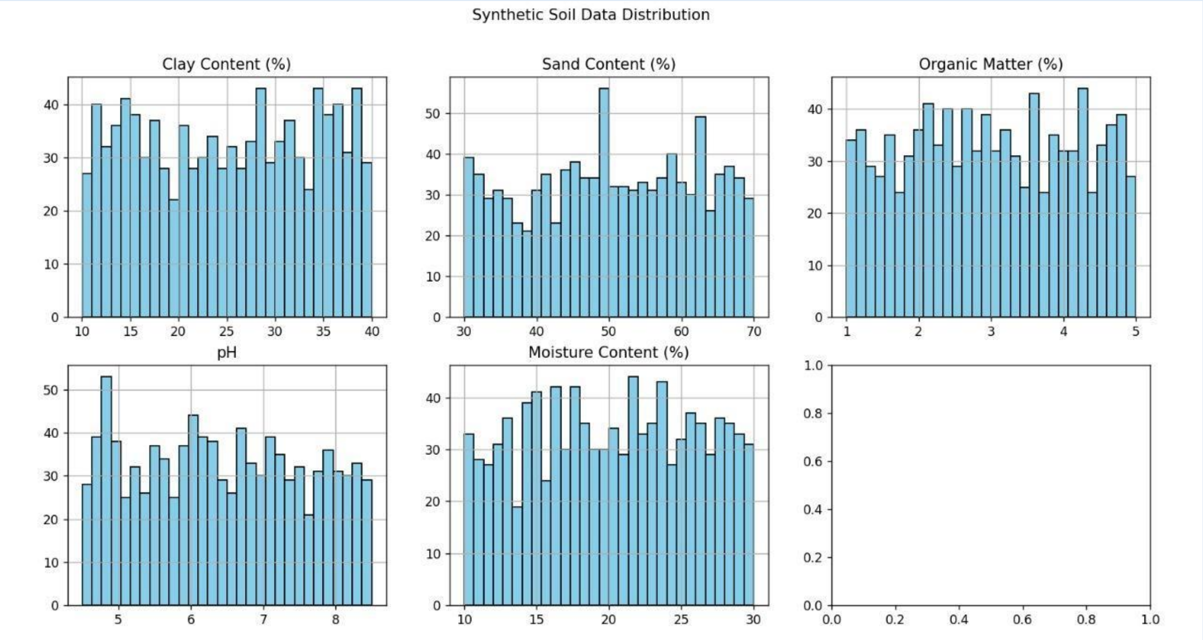
Code in python

```
import numpy as np

import matplotlib.pyplot as plt


# Function to generate synthetic soil data

def generate_synthetic_soil_data(num_samples=1000):

    # Define synthetic soil parameters

    clay_content = np.random.uniform(10, 40, num_samples) # percentage

    sand_content = np.random.uniform(30, 70, num_samples)  # percentage

    organic_matter = np.random.uniform(1, 5, num_samples) # percentage

    pH = np.random.uniform(4.5, 8.5, num_samples)          # pH level

    moisture_content = np.random.uniform(10, 30, num_samples)  # percentage
```

```
    # Combine parameters into a synthetic dataset
    synthetic_soil_data = np.column_stack((clay_content, sand_content, organic_matter, pH, moisture_content))

    return synthetic_soil_data

# Generate synthetic soil data
synthetic_data = generate_synthetic_soil_data()

# Display a subset of the synthetic data
print("Generated Synthetic Soil Data (Sample):")
print(synthetic_data[:5, :])

# Plotting the data
fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(15, 10))
fig.suptitle('Synthetic Soil Data Distribution')

parameter_names = ['Clay Content (%)', 'Sand Content (%)', 'Organic Matter (%)', 'pH', 'Moisture Content (%)']

for i in range(5):
    row, col = divmod(i, 3)
    axes[row, col].hist(synthetic_data[:, i], bins=30, color='skyblue', edgecolor='black')
    axes[row, col].set_title(parameter_names[i])
    axes[row, col].grid(True)

plt.show()
```

Data Generation

```python
import numpy as np
import matplotlib.pyplot as plt

# Function to generate synthetic soil data
def generate_synthetic_soil_data(num_samples=1000):
    # Define synthetic soil parameters
    clay_content = np.random.uniform(10, 40, num_samples)  # percentage
    sand_content = np.random.uniform(30, 70, num_samples)  # percentage
    organic_matter = np.random.uniform(1, 5, num_samples)   # percentage
    pH = np.random.uniform(4.5, 8.5, num_samples)           # pH level
    moisture_content = np.random.uniform(10, 30, num_samples)  # percentage

    # Combine parameters into a synthetic dataset
    synthetic_soil_data = np.column_stack((clay_content, sand_content, organic_matter, pH, moisture_content))

    return synthetic_soil_data

# Generate synthetic soil data
synthetic_data = generate_synthetic_soil_data()

# Display a subset of the synthetic data
print("Generated Synthetic Soil Data (Sample):")
print(synthetic_data[:5, :])

# Plotting the data
fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(15, 10))
fig.suptitle('Synthetic Soil Data Distribution')

parameter_names = ['Clay Content (%)', 'Sand Content (%)', 'Organic Matter (%)', 'pH', 'Moisture Content (%)']

for i in range(5):
    row, col = divmod(i, 3)
    axes[row, col].hist(synthetic_data[:, i], bins=30, color='skyblue', edgecolor='black')
    axes[row, col].set_title(parameter_names[i])
    axes[row, col].grid(True)

plt.show()
```



```
Generated Synthetic Soil Data (Sample):
[[16.64040324 34.64015697  1.55732523  6.6134871  24.00316595]
 [38.49531412 61.92848263  4.18653142  7.10086653 27.39561747]
 [32.11232395 54.39720921  2.38998969  5.00061297 20.35473903]
 [32.0353458  37.71775919  2.9555554   6.62395667 11.26153047]
 [32.61607015 48.34270231  4.04915729  7.22302445 26.40515844]]
```

Divyashree BM

Software used: Spyder

Library used: Faker

```
import pandas as pd

from faker import Faker

import random

# Set up Faker and seed for reproducibility

fake = Faker()

random.seed(42)

# Function to generate synthetic soil data

def generate_soil_data(num_records=1000):

    data = []

    for _ in range(num_records):
```

```
        latitude = round(random.uniform(8.0, 37.0), 6)
        longitude = round(random.uniform(68.0, 97.0), 6)
        soil_type = random.choice(['Sandy', 'Clayey', 'Loamy'])
        pH = round(random.uniform(4.0, 9.0), 2)
        moisture = round(random.uniform(10.0, 50.0), 2)
        organic_content = round(random.uniform(0.5, 5.0), 2)

        data.append([latitude, longitude, soil_type, pH, moisture,
organic_content])

    columns = ['Latitude', 'Longitude', 'Soil_Type', 'pH',
'Moisture', 'Organic_Content']
    df = pd.DataFrame(data, columns=columns)
    return df

# Generate synthetic soil data
synthetic_soil_data = generate_soil_data(num_records=1000)

# Save synthetic data to a CSV file
synthetic_soil_data.to_csv('synthetic_soil_data.csv',
index=False)
```

```
# Display the generated data
print(synthetic_soil_data.head())


import matplotlib.pyplot as plt
import seaborn as sns

# Load synthetic data (assuming it's in the same directory)
synthetic_soil_data = pd.read_csv('synthetic_soil_data.csv')

# Scatter plot of Latitude vs Longitude colored by Soil Type
plt.figure(figsize=(12, 8))
sns.scatterplot(x='Longitude', y='Latitude', hue='Soil_Type',
data=synthetic_soil_data, palette='viridis', s=50)
plt.title('Synthetic Soil Data: Latitude vs Longitude')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.legend(title='Soil Type')
plt.show()
```

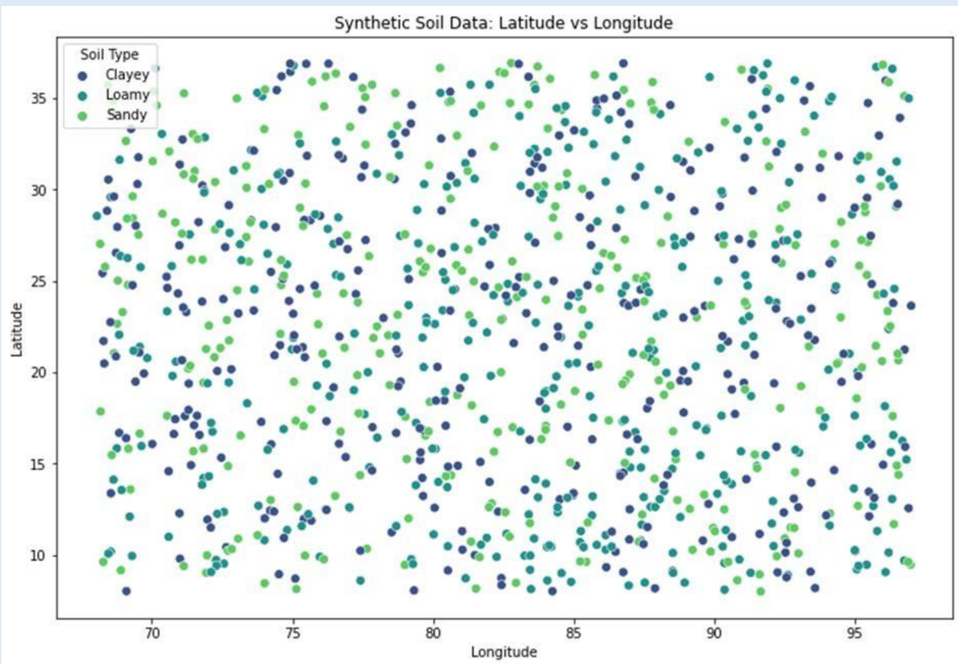
Code

```python
import pandas as pd
from faker import Faker
import random

# Set up Faker and seed for reproducibility
fake = Faker()
random.seed(42)

# Function to generate synthetic soil data
def generate_soil_data(num_records=1000):
    data = []
    for _ in range(num_records):
        latitude = round(random.uniform(8.0, 37.0), 6)
        longitude = round(random.uniform(68.0, 97.0), 6)
        soil_type = random.choice(['Sandy', 'Clayey', 'Loamy'])
        pH = round(random.uniform(4.0, 9.0), 2)
        moisture = round(random.uniform(10.0, 50.0), 2)
        organic_content = round(random.uniform(0.5, 5.0), 2)

        data.append([latitude, longitude, soil_type, pH, moisture, organic_content])

    columns = ['Latitude', 'Longitude', 'Soil_Type', 'pH', 'Moisture', 'Organic_Content']
    df = pd.DataFrame(data, columns=columns)
    return df

# Generate synthetic soil data
synthetic_soil_data = generate_soil_data(num_records=1000)

# Save synthetic data to a CSV file
synthetic_soil_data.to_csv('synthetic_soil_data.csv', index=False)

# Display the generated data
print(synthetic_soil_data.head())
```

```python

import matplotlib.pyplot as plt
import seaborn as sns

# Load synthetic data (assuming it's in the same directory)
synthetic_soil_data = pd.read_csv('synthetic_soil_data.csv')

# Scatter plot of Latitude vs Longitude colored by Soil Type
plt.figure(figsize=(12, 8))
sns.scatterplot(x='Longitude', y='Latitude', hue='Soil_Type', data=synthetic_soil_data, palette='viridis', s=50)
plt.title('Synthetic Soil Data: Latitude vs Longitude')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.legend(title='Soil Type')
plt.show()
```

Data Generation

```
In [1]: runfile('C:/Users/bmdiv/.spyder-py3/soil.py', wdir='C:/Users/bmdiv/.spyder-
py3')
    Latitude  Longitude Soil_Type    pH  Moisture  Organic_Content
0  26.543377  68.725312    Clayey  5.22     15.58             0.96
1  29.479365  83.815629     Loamy  6.11     11.19             1.48
2  22.655303  68.769543     Sandy  7.58     38.05             2.39
3  21.027062  76.067531     Sandy  7.79     16.39             2.40
4  16.058269  74.244099    Clayey  4.51     25.20             2.12

                                Important

 Figures are displayed in the Plots pane by default. To make them also
 appear inline in the console, you need to uncheck "Mute inline plotting"
 under the options menu of Plots.
```



Is synthetic data available in every field?

Synthetic data serves as a flexible solution across many domains, providing a valuable alternative when real data is limited or inaccessible. However, its effectiveness is bound by inherent constraints stemming from its artificial nature. These limitations are notably evident in contexts where the complexities of real-world scenarios, unpredictable outliers, and intricate domain-specific intricacies demand precision beyond what synthetic data can replicate. Fields relying heavily on nuanced human behaviors, domain-specific correlations like medical research, and compliance-driven sectors often encounter challenges with synthetic data due to its inability to capture these subtle nuances.

Additionally, ethical and legal considerations regarding privacy and regulatory compliance further restrict its application in sensitive sectors. Despite its utility, the suitability of synthetic data hinges on a careful assessment of its limitations against the specific needs and intricacies of each field or domain.

While synthetic data serves as a valuable surrogate, its limitations warrant a discerning approach to its application. Assessing its relevance against real-world complexities, unanticipated scenarios, and the domain-specific demands of various sectors becomes imperative. Understanding that synthetic data might not fully encapsulate the intricacies inherent in diverse fields, a nuanced evaluation on a case-by-case basis is crucial.

By acknowledging these limitations and aligning its usage with specific requirements, synthetic data can be effectively leveraged where its benefits outweigh the constraints, facilitating informed decision-making across multiple domains.

Conclusions

In this project, I set out to create artificial data resembling real-world information when authentic data isn't readily available. I used statistical models to mimic the patterns and features observed in the original dataset, aiming to replicate key attributes within the confines of artificial generation.

While synthetic data proves valuable in filling data gaps, it's essential to recognize its limitations. It might miss certain complexities and specifics found in real datasets, making it less reliable in unforeseen scenarios or specialized fields.

Nevertheless, the generated synthetic data holds promise for preliminary analysis, model testing, and aiding decision-making processes across various fields. It's a useful resource, but it's not a complete substitute for real data.

In short, this project highlights the potential of synthetic data to fill data gaps, offering a preliminary solution across different domains. However, it's important to remember its limitations and use it judiciously alongside real data.

Bibliography

GretelAI, Google resources, YouTube Learning, ChatGPT, Scholarly