College Name: VIT Bhopal
Student Name: Garvit

Synthetic Data Generation for Soil Data

Phase 2: Project Execution and Demonstration

1. Project Title:

Synthetic Data Generation for Soil Data

2. Objective Recap:

The objective of this project is to **generate synthetic data for soil properties** using the **CTGAN (Conditional Tabular Generative Adversarial Networks)** model. This aims to address privacy concerns associated with sharing real soil data while providing a realistic dataset for research and development purposes.

3. Technologies Used:

- Python
- SDV (Synthetic Data Vault) library
- Pandas
- Plotly
- Scikit-learn
- Google Colab

4. Full Code Implementation:

Step 1: Install Necessary Libraries

```
pip install sdv
```

Step 2: Data Loading and Preprocessing

```
import pandas as pd
df = pd.read_csv('soil.csv')

from sklearn.preprocessing import LabelEncoder
lr = LabelEncoder()
df['District_id'] = lr.fit_transform(df['District '])
df = df.drop('District ',axis =1)
```

Step 3: Metadata Definition and Validation

```
from sdv.metadata import SingleTableMetadata
metadata = SingleTableMetadata()
metadata.detect_from_dataframe(df)
metadata.validate_data(data=df)
```

Step 4: Synthesizer Training

```
from sdv.single_table import CTGANSynthesizer
synthesizer = CTGANSynthesizer(metadata, enforce_rounding=False,
epochs=3000, verbose=True)
synthesizer.fit(df)
```

Step 5: Synthetic Data Generation

```
synthetic_data = synthesizer.sample(num_rows=10000)
```

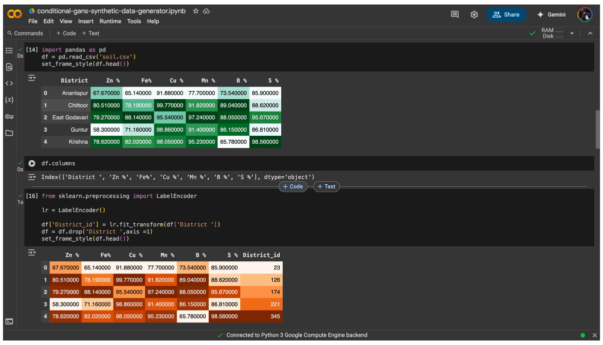
Step 6: Evaluation and Visualization

```
from sdv.evaluation.single_table import get_column_plot, evaluate_quality

# Generate plots for comparing real and synthetic data distributions
# ... (Code for creating Plotly subplots) ...

# Evaluate the quality of the synthetic data
quality_report = evaluate_quality(df, synthetic_data, metadata)
```

5. Output Screenshots:


conditional-gans-synthetic-data-generator.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text
Share
Gemini
RAM Disk

```
[14] import pandas as pd
     df = pd.read_csv('soil.csv')
     set_frame_style(df.head())
```

	District	Zn %	Fe%	Cu %	Mn %	B %	S %
0	Anantapur	67.670000	65.140000	91.880000	77.700000	73.540000	85.900000
1	Chittoor	80.510000	78.190000	99.770000	91.820000	89.040000	88.620000
2	East Godavari	79.270000	88.140000	95.540000	97.240000	88.050000	95.670000
3	Guntur	58.300000	71.160000	98.860000	91.400000	86.150000	86.810000
4	Krishna	78.620000	82.020000	98.050000	95.230000	65.780000	98.560000

```
df.columns
```

```
Index(['District ', 'Zn %', 'Fe%', 'Cu %', 'Mn %', 'B %', 'S %'], dtype='object')
```

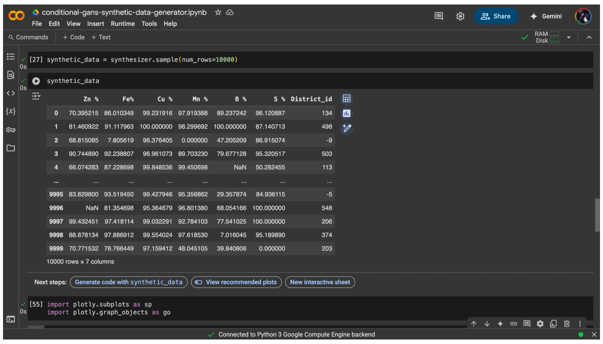
```
[16] from sklearn.preprocessing import LabelEncoder

     lr = LabelEncoder()

     df['District_id'] = lr.fit_transform(df['District '])
     df = df.drop('District ',axis =1)
     set_frame_style(df.head())
```

	Zn %	Fe%	Cu %	Mn %	B %	S %	District_id
0	67.670000	65.140000	91.880000	77.700000	73.540000	85.900000	23
1	80.510000	78.190000	99.770000	91.820000	89.040000	88.620000	126
2	79.270000	88.140000	95.540000	97.240000	88.050000	95.670000	174
3	58.300000	71.160000	98.860000	91.400000	86.150000	86.810000	221
4	78.620000	82.020000	98.050000	95.230000	65.780000	98.560000	345

Connected to Python 3 Google Compute Engine backend


conditional-gans-synthetic-data-generator.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text
Share
Gemini
RAM Disk

```
[27] synthetic_data = synthesizer.sample(num_rows=10000)
```

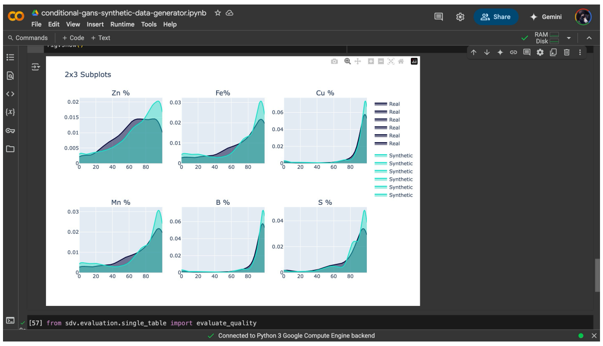
```
synthetic_data
```

	Zn %	Fe%	Cu %	Mn %	B %	S %	District_id
0	70.395215	86.010349	99.231916	97.919388	89.237242	96.120887	134
1	81.460922	91.117963	100.000000	98.299692	100.000000	87.140713	498
2	68.815085	7.805619	96.376405	0.000000	47.205209	86.915074	-9
3	90.744890	92.238807	96.961073	89.703230	79.677128	95.320517	503
4	66.074283	87.228698	99.848536	99.450698	NaN	50.282455	113
...	...	...	...	...	...	...	...
9995	83.829800	93.519450	99.427946	95.356862	29.357874	84.936115	-5
9996	NaN	81.354698	95.364679	96.801380	68.054166	100.000000	548
9997	99.432451	97.418114	99.032291	92.784103	77.541025	100.000000	206
9998	88.878134	97.886912	99.554024	97.618530	7.016045	95.189890	374
9999	70.771532	76.766449	97.159412	48.045105	39.840806	0.000000	203

10000 rows × 7 columns

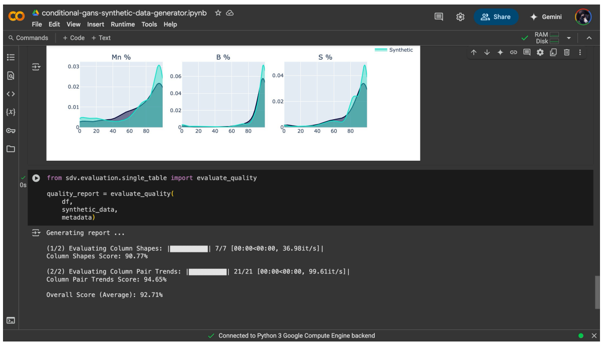
```
[55] import plotly.subplots as sp
     import plotly.graph_objects as go
```

Connected to Python 3 Google Compute Engine backend



```
[57] from sdv.evaluation.single_table import evaluate_quality
```

Connected to Python 3 Google Compute Engine backend



```
from sdv.evaluation.single_table import evaluate_quality

quality_report = evaluate_quality(
    df,
    synthetic_data,
    metadata)
```

```
Generating report ...

(1/2) Evaluating Column Shapes: |██████████| 7/7 [00:00<00:00, 36.98it/s]|
Column Shapes Score: 90.77%

(2/2) Evaluating Column Pair Trends: |██████████| 21/21 [00:00<00:00, 99.61it/s]|
Column Pair Trends Score: 94.65%

Overall Score (Average): 92.71%
```

Connected to Python 3 Google Compute Engine backend

6. Conclusion:

This project successfully generated **synthetic soil data** using the **CTGAN** model. The evaluation results and visualizations demonstrate the quality and similarity of the synthetic data to the original dataset. This synthetic data can be used for research and development while protecting the privacy of sensitive information in the original soil data.

7. References:

- SDV (Synthetic Data Vault) documentation
- CTGAN research paper
- Pandas documentation
- Plotly documentation
- Scikit-learn documentation