Using SDMetric Demonstration

**1. Import Required Libraries**

1. import pandas as pd
2. from sdv.single\_table import CTGANSynthesizer
3. from sdmetrics.reports.single\_table import QualityReport
4. from sdv.metadata import SingleTableMetadata

* **Pandas**: Used for data manipulation and analysis. Provides data structures like DataFrame for handling tabular data.
* **CTGANSynthesizer**: A part of the SDV (Synthetic Data Vault) library that uses a Conditional Generative Adversarial Network (GAN) to generate synthetic tabular data.
* **QualityReport**: From the SDMetrics library, used for evaluating the quality of synthetic data.
* **SingleTableMetadata**: Used to define and manage metadata for a single table. Metadata includes details about the column types, constraints, and other structural information of the dataset.

**2. Load Real Data**

1. data = pd.read\_csv('./data/customer\_data.csv')
2. real\_data = pd.DataFrame(data)

* **Loading Data:** pd.read\_csv reads the CSV file located at ./data/customer\_data.csv and loads it into a pandas DataFrame called real\_data. This step is crucial as it provides the real dataset for training the synthesizer and for evaluation purposes.

**3. Initialize the Metadata Object**

1. metadata = SingleTableMetadata()
2. metadata.detect\_from\_dataframe(data)

* **SingleTableMetadata**: This object is created to manage metadata for the data table.
* detect\_from\_dataframe: Automatically detects and defines metadata from the provided DataFrame, identifying column types (e.g., numerical, categorical), primary keys, and any constraints present in the data.

**4. Initialize the Synthesizer**

1. model = CTGANSynthesizer(metadata)

* **CTGANSynthesizer**: This synthesizer is initialized with the metadata object. It uses a Conditional GAN model to learn from the real data and generate synthetic data that adheres to the same structure and constraints specified in the metadata.

**5. Fit the Synthesizer to the Real Data**

1. model.fit(data)

* **fit Method**: Trains the CTGANSynthesizer on the real data. The model learns the distribution and relationships between different columns in the dataset. This involves:
  + **Generator Network**: Creates synthetic data that mimics the real data.
  + **Discriminator Network**: Distinguishes between real and synthetic data. The two networks iteratively improve until the generator produces realistic synthetic data that the discriminator cannot easily distinguish from real data.

**6. Generate Synthetic Data**

1. synthetic\_data = model.sample(num\_rows=10)

* **sample Method**: Generates new synthetic data based on the learned patterns from the real data. Here, 10 rows of synthetic data are created.

**7. Convert Metadata to Dictionary**

1. metadata\_dict = metadata.to\_dict()

* **to\_dict Method**: Converts the metadata object into a dictionary format, which is needed for the evaluation step.

**8. Evaluate the Synthetic Data Quality**

1. report = QualityReport()
2. report.generate(real\_data=data, synthetic\_data=synthetic\_data, metadata=metadata\_dict)

* **QualityReport**: An object from the SDMetrics library used to evaluate the quality of the synthetic data.
* **generate Method**: Generates a quality report by comparing the real data to the synthetic data using the metadata dictionary. This report includes various metrics to assess the effectiveness of the synthetic data generation process.

**9. Display Synthetic Data and Quality Report**

1. print("Synthetic Data:")
2. print(synthetic\_data)
3. print("\nQuality Score:")
4. print(report.get\_score())

* **Display Data**: Prints the generated synthetic data to the console.
* **get\_score Method**: Retrieves and prints the overall quality score from the quality report. This score provides a quantitative measure of how well the synthetic data matches the real data.

**Explanation of Key Components**

* **Pandas**: Essential for data manipulation, allowing for efficient handling of tabular data structures.
* **CTGANSynthesizer**: Utilizes a Conditional GAN model to learn and generate synthetic data that maintains the statistical properties and relationships found in the real dataset.
* **SingleTableMetadata**: Manages metadata, ensuring the synthetic data generation process respects the structure and constraints of the real data.
* **QualityReport**: Provides a comprehensive evaluation of the synthetic data quality, highlighting strengths and areas for improvement.
* **fit Method**: Crucial for training the synthesizer, ensuring it learns the correct data distributions and relationships.
* **sample Method**: Generates synthetic data that can be used for various purposes, including testing, machine learning training, and more.
* **to\_dict Method**: Converts metadata to a format suitable for evaluation.
* **generate Method**: Evaluates the synthetic data, ensuring it meets the desired quality standards.
* **get\_score Method**: Quantitatively measures the quality of the synthetic data, providing a clear metric for assessment.

For more detailed information, you can refer to the [SDV Documentation](https://docs.sdv.dev/sdv/single-table-data/modeling/synthesizers/ctgansynthesizer/).

Resources for this lecture