Below is an overview of the **folders** you have in your CTGAN/SDV project and how they fit into a typical Python development workflow with **GitHub** and **VS Code**. The **short version** is:

1. **ctgan** (the main code)
2. **tests** (test suite)
3. **venv** (virtual environment)
4. **my\_project** (your personal project folder)
5. Various other support folders (ctgan.egg-info, scripts, etc.)

Following that, you’ll see **step-by-step** instructions on how to use them in VS Code and GitHub.

**1. Explanation of Each Folder**

**1. The ctgan folder**

* **What it is**: The **core library code** for CTGAN.
* **Inside**:
  + **synthesizers** subfolder: Contains Python files (ctgan.py, tvae.py, etc.) implementing different generative models (CTGAN, TVAE, etc.).
  + **Other modules** (e.g., errors.py): Helper classes and error definitions.

Essentially, this folder **is** the Python package. When you run pip install -e . in the **root** directory, it installs this ctgan folder as a local package so you can do:

python

Copy

from ctgan import CTGANSynthesizer

in your scripts.

**2. The tests folder**

* **What it is**: Contains **test files** (like test\_ctgan.py, test\_tasks.py) that verify the correctness of the library.
* **How to use**:
  + If you’re in the **root folder** (the same level as ctgan and tests), you can run:

bash

Copy

pytest tests

or

bash

Copy

python -m unittest discover tests

* + This ensures that Python can find the ctgan package and run all tests.

**3. The venv folder**

* **What it is**: A **virtual environment** folder containing Python binaries and installed packages.
* **How to use**:
  + Activate it before installing or running anything:
    - **Windows (PowerShell)**:

bash

Copy

venv\Scripts\activate

* + - **macOS/Linux**:

bash

Copy

source venv/bin/activate

* + Once activated, pip install commands go into this environment, preventing conflicts with your system Python.

**4. The my\_project folder**

* **What it is**: A **personal folder** you created for your own scripts, data, or code that **uses** CTGAN.
* **Inside**:
  + **data**: Where you might keep CSV or other data files.
  + **scripts**: Your personal scripts for analyzing data or generating synthetic data with CTGAN.
* **How to use**:
  + Put your **Python scripts** that import CTGAN here (e.g., run\_ctgan.py).
  + Example:

python

Copy

from ctgan import CTGANSynthesizer

import pandas as pd

df = pd.read\_csv("data/my\_data.csv")

ctgan = CTGANSynthesizer(epochs=100)

ctgan.fit(df)

synthetic = ctgan.sample(500)

* + This keeps your **custom** code separate from the **library** code in ctgan.

**5. ctgan.egg-info folder**

* **What it is**: Automatically generated **metadata** about the installed package (name, version, dependencies).
* **How it appears**: When you run pip install -e . or python setup.py install, Python creates an .egg-info or .dist-info folder with package details.
* **You generally don’t edit** this folder. It’s for Python’s packaging system.

**6. scripts folder (top-level)**

* **What it is**: Contains **utility scripts** for maintainers or release processes (e.g., release\_notes\_generator.py).
* **You probably don’t need** these for normal usage unless you’re contributing to the official repository’s release cycle.

**7. Other folders (e.g., .github, examples, etc.)**

* **.github**: Holds GitHub Actions CI/CD workflows or issue/PR templates.
* **examples**: Example usage notebooks or scripts.
* **pyproject.toml, setup.py**: Project configuration for building/installing the package.

**2. Step-by-Step Workflow in VS Code and GitHub**

Below is a **common** way to use these folders with **GitHub** and **VS Code**.

**Step 1: Open the Repository in VS Code**

1. **Clone** the CTGAN repository (or your fork) from GitHub:

bash

Copy

git clone https://github.com/sdv-dev/CTGAN.git

1. **Open** that folder in VS Code:
   * Go to File → Open Folder... → select the CTGAN folder.

**Step 2: Create/Activate the Virtual Environment**

1. **Create** a virtual environment (if not already created):

bash

Copy

python -m venv venv

1. **Activate** it:

bash

Copy

# Windows:

venv\Scripts\activate

# macOS/Linux:

source venv/bin/activate

1. **Check** that (venv) appears in your terminal prompt.

**Step 3: Install CTGAN in Editable Mode**

In the **root** of CTGAN (the same level as setup.py and the ctgan folder), run:

bash

Copy

pip install -e .

This tells Python: “Install this folder (ctgan package) so I can import it from anywhere on this machine.”

**Step 4: Run Tests (Optional)**

To confirm everything works:

bash

Copy

pytest tests

or

bash

Copy

python -m unittest discover tests

If all tests pass, your environment is set up correctly.

**Step 5: Work in my\_project Folder**

1. **Create** a new script in my\_project/scripts, say run\_ctgan.py:

python

Copy

from ctgan import CTGANSynthesizer

import pandas as pd

# Suppose your CSV is in my\_project/data

df = pd.read\_csv("data/my\_data.csv")

ctgan = CTGANSynthesizer(epochs=300)

ctgan.fit(df)

synthetic\_data = ctgan.sample(1000)

print(synthetic\_data.head())

1. **Run** it from the integrated terminal:

bash

Copy

cd my\_project/scripts

python run\_ctgan.py

1. **Version control** your scripts in my\_project if you want them on GitHub. You can do git add my\_project/scripts/run\_ctgan.py, etc., and commit/push changes.

**Step 6: Contribute (Optional)**

* If you want to **contribute** code changes back to CTGAN, you’d typically:
  1. **Fork** the sdv-dev/CTGAN repo on GitHub.
  2. **Add** your fork as a remote.
  3. **Create** a new branch for your changes.
  4. **Commit** and **push** to your fork.
  5. **Open a pull request** to the official repo.

**3. Tips & Best Practices**

1. **Keep library code and your project code separate**
   * ctgan folder is the library.
   * my\_project is your personal usage.
2. **Use the venv**
   * Always activate it before installing or running code.
3. **Run tests from the root**
   * If you run pytest or python -m unittest from CTGAN/, Python can discover tests and the ctgan package.
4. **Ignore ctgan.egg-info**
   * It’s auto‐generated metadata. You typically don’t modify it directly.
5. **Use Git**
   * Commit changes to your own scripts in my\_project.
   * If you modify ctgan source code, commit on a new branch so you can keep track of changes.

**4. In Summary**

* **ctgan**: The main code for the CTGAN library.
* **tests**: Automated tests to ensure the library works.
* **venv**: Your virtual environment for Python dependencies.
* **my\_project**: Where you put your own data and scripts that **use** the CTGAN library.

**Workflow**:

1. Clone repo → open in VS Code.
2. Create/activate venv.
3. pip install -e . in the root folder.
4. (Optional) Run pytest tests.
5. Create your own scripts in my\_project to train or use CTGAN.
6. Use Git to manage changes.

That’s it! With this setup, you have a **clean separation** between the library’s source code (ctgan) and your personal usage code (my\_project), all under version control with Git and easily editable in VS Code.