This Python code is designed to generate a synthetic dataset of product information, using the pandas, numpy, and Faker libraries, along with some built-in Python modules like random. Let's break it down step by step:

**1. Class Definition**

python

Copy code

class SyntheticProductDataGenerator:

The class SyntheticProductDataGenerator is created to encapsulate the process of generating synthetic product data. This class allows you to easily generate customizable datasets of products by specifying the number of products you need.

**2. \_\_init\_\_ Method**

python

Copy code

def \_\_init\_\_(self, num\_products=1000):

* This is the constructor method, which initializes the class. It takes an optional argument num\_products, which defaults to 1000.
* It initializes the Faker library (used for generating random fake data) and sets up lists of predefined product categories and brands.

python

Copy code

self.categories = ['Electronics', 'Clothing', 'Home & Garden', 'Sports & Outdoors', 'Books', 'Toys & Games']

self.brands = ['TechPro', 'FashionStyle', 'HomeComfort', 'SportsMaster', 'BookWorm', 'PlayFun']

**3. Methods for Generating Data**

The class has several helper methods that generate various components of each product entry.

**a. generate\_product\_id()**

python

Copy code

def generate\_product\_id(self):

return f'P{self.fake.unique.random\_number(digits=4)}'

* Generates a unique 4-digit product ID prefixed with "P", e.g., P1234.

**b. generate\_product\_name()**

python

Copy code

def generate\_product\_name(self, category):

adjectives = ['Amazing', 'Incredible', 'Fantastic', 'Awesome', 'Ultimate', 'Premium']

...

return f"{random.choice(adjectives)} {random.choice(nouns)}"

* This method generates product names by combining a random adjective with a category-specific noun. For example, for Electronics, the product name could be "Amazing Smartphone".

**c. generate\_description()**

python

Copy code

def generate\_description(self, name, category):

return f"A high-quality {category.lower()} product. This {name} is perfect for {self.fake.sentence(nb\_words=6)}"

* Generates a simple description for the product, where the category and name are included. It uses the Faker library to generate a random 6-word sentence as part of the description.

**d. generate\_price()**

python

Copy code

def generate\_price(self, category):

if category == 'Electronics':

return round(random.uniform(50, 2000), 2)

...

* This method generates a price for the product based on its category, with a specific price range for each type of product (e.g., Electronics: $50–$2000).

**4. Generating the Dataset (generate\_dataset)**

python

Copy code

def generate\_dataset(self):

data = []

for \_ in range(self.num\_products):

category = random.choice(self.categories)

...

data.append(product)

return pd.DataFrame(data)

* Loops through the number of products (self.num\_products) to create a dataset.
* For each product, it randomly selects a category, generates a name, description, price, rating (1.0 to 5.0), and number of reviews (0 to 1000).
* The generated product details are stored in a dictionary and appended to a list.
* Finally, the list of dictionaries is converted into a Pandas DataFrame and returned.

**5. Saving the Dataset (save\_dataset)**

python

Copy code

def save\_dataset(self, df, file\_path):

df.to\_csv(file\_path, index=False)

print(f"Dataset saved to {file\_path}")

* Saves the generated dataset as a CSV file.

**6. Usage Example**

python

Copy code

generator = SyntheticProductDataGenerator(num\_products=1000)

product\_df = generator.generate\_dataset()

generator.save\_dataset(product\_df, 'synthetic\_products.csv')

* Creates an instance of SyntheticProductDataGenerator with 1000 products.
* Generates the dataset and stores it in product\_df.
* Saves the dataset to a CSV file.

**7. Output and Summary**

python

Copy code

print(product\_df.head())

print(product\_df.describe())

print(product\_df['category'].value\_counts(normalize=True))

* Displays the first few rows of the dataset.
* Prints summary statistics (like mean, min, max) of the dataset’s numerical columns.
* Prints the distribution of products across different categories, normalized as percentages.

This script is highly customizable and useful for generating product data for testing or simulation purposes.

4o

**You said:**

appreve it so i can make jupeyternotebook

**ChatGPT said:**

ChatGPT

To use this code in a Jupyter Notebook, you can break it into cells for better modularity and understanding. Here's an approved and notebook-friendly version of the code with comments for each section.

**1. Import Required Libraries**

python

Copy code

# Cell 1: Importing necessary libraries

import pandas as pd

import numpy as np

from faker import Faker

import random

**2. Define the Class**

python

Copy code

# Cell 2: Defining the SyntheticProductDataGenerator class

class SyntheticProductDataGenerator:

def \_\_init\_\_(self, num\_products=1000):

self.num\_products = num\_products

self.fake = Faker()

# Categories and Brands

self.categories = ['Electronics', 'Clothing', 'Home & Garden', 'Sports & Outdoors', 'Books', 'Toys & Games']

self.brands = ['TechPro', 'FashionStyle', 'HomeComfort', 'SportsMaster', 'BookWorm', 'PlayFun']

def generate\_product\_id(self):

return f'P{self.fake.unique.random\_number(digits=4)}'

def generate\_product\_name(self, category):

adjectives = ['Amazing', 'Incredible', 'Fantastic', 'Awesome', 'Ultimate', 'Premium']

if category == 'Electronics':

nouns = ['Smartphone', 'Laptop', 'Headphones', 'Smartwatch', 'Camera', 'Tablet']

elif category == 'Clothing':

nouns = ['T-shirt', 'Jeans', 'Dress', 'Jacket', 'Shoes', 'Hat']

elif category == 'Home & Garden':

nouns = ['Sofa', 'Lamp', 'Rug', 'Plant', 'Pillow', 'Vase']

elif category == 'Sports & Outdoors':

nouns = ['Bicycle', 'Tent', 'Backpack', 'Kayak', 'Yoga Mat', 'Running Shoes']

elif category == 'Books':

return f"The {self.fake.word().title()} {self.fake.word().title()}"

else: # Toys & Games

nouns = ['Board Game', 'Puzzle', 'Action Figure', 'Doll', 'Building Blocks', 'Remote Control Car']

return f"{random.choice(adjectives)} {random.choice(nouns)}"

def generate\_description(self, name, category):

return f"A high-quality {category.lower()} product. This {name} is perfect for {self.fake.sentence(nb\_words=6)}"

def generate\_price(self, category):

if category == 'Electronics':

return round(random.uniform(50, 2000), 2)

elif category == 'Clothing':

return round(random.uniform(10, 200), 2)

elif category == 'Home & Garden':

return round(random.uniform(20, 500), 2)

elif category == 'Sports & Outdoors':

return round(random.uniform(15, 1000), 2)

elif category == 'Books':

return round(random.uniform(5, 50), 2)

else: # Toys & Games

return round(random.uniform(10, 100), 2)

def generate\_dataset(self):

data = []

for \_ in range(self.num\_products):

category = random.choice(self.categories)

name = self.generate\_product\_name(category)

product = {

'product\_id': self.generate\_product\_id(),

'name': name,

'category': category,

'brand': random.choice(self.brands),

'description': self.generate\_description(name, category),

'price': self.generate\_price(category),

'rating': round(random.uniform(1, 5), 1),

'num\_reviews': random.randint(0, 1000)

}

data.append(product)

return pd.DataFrame(data)

def save\_dataset(self, df, file\_path):

df.to\_csv(file\_path, index=False)

print(f"Dataset saved to {file\_path}")

**3. Instantiate the Generator and Generate Data**

python

Copy code

# Cell 3: Usage example - Generate synthetic product data

generator = SyntheticProductDataGenerator(num\_products=1000)

product\_df = generator.generate\_dataset()

**4. Save the Dataset (Optional)**

python

Copy code

# Cell 4: Save the dataset to a CSV file

generator.save\_dataset(product\_df, 'synthetic\_products.csv')

**5. Display the First Few Rows of the Data**

python

Copy code

# Cell 5: Display the first few rows of the dataset

product\_df.head()

**6. Summary Statistics**

python

Copy code

# Cell 6: Display summary statistics

product\_df.describe()

**7. Category Distribution**

python

Copy code

# Cell 7: Display the distribution of products across categories

product\_df['category'].value\_counts(normalize=True)

This setup will work perfectly in a Jupyter Notebook, where each section can be run independently in cells. You can simply copy and paste the code into your notebook cells and run them step by step.

4o

Top of Form

'

Bottom of Form

ChatGPT can make mistake