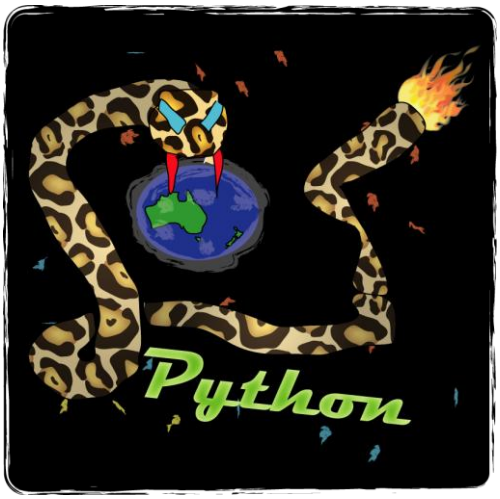
Synthesizing Data with Mimesis



A client, Inta-Galactic Gaming, has just built an online game that's taking the world by storm. It's called sPython ("Space Python").

Think of the old 'Snake Game'. But in space. With thousands of other players. Your snake-like space-ship devours planets, and gets bigger. You can't crash into yourself or the other players. They make money from in-game purchases.

They wish to build the infrastructure for their eventual data pipeline. However, it's in bad shape! They don't have data in the correct place or format (Bob, the Java dude, copies and pastes it by hand at the moment). They would also like some data generated for a potential marketing campaign.

The client have commissioned you to produce two deliverables: a synthetic database of users, and an Excel spreadsheet that has the resulting sales from marketing at various intensities at each customer (again, synthesized). They want the data to be somewhat realistic.

---

1. Use the Python `mimesis` library to synthesize a population of 10,000 customers. The details are: customer ID, first name, surname, gender, email address, a hashed password, their address, age, credit_card_number and expiry date and their 'security answer' (hint: mimesis has the Person(), Address(), Payment() and Text() classes).

2. Give each customer a nominal account balance. For realism, make the account balance follow a Pareto distribution, so that the top 20% of customers have 80% of the total in-game balance (the so-called '80-20 rule'). Test that this is indeed the case (you may use the `pareto` function from scipy.stats).

3. Make this dataset a little more realistic. Ten percent of the customers weren't comfortable with volunteering their gender. A similar fraction did not give their address. Make this data 'go missing'. Duplicate a random 10% of the data in the dataset.

4. The client wants this customer-level data as a SQL database. Create a SQLite database with this synthetic data.

5. Generate sales as a linear function of the customers' age (as a weak power), account balance and the interaction between a marketing campaign and the age bracket it was intended for (25-35).

6. Make the sales data more realistic by adding a small amount of noise. You may use the `random.normal` function from NumPy to generate normally distributed random numbers for this.

7. The sales data will be used by the marketing department, whom only use Excel. Output the sales data as an Excel spreadsheet.

---

8. Perform some Exploratory Data Analysis (EDA) to see that the population has been synthesized as expected.