Project Documentation:
Synthetic Data Generation for Fraud Detection

Authors: **Alexander Wirths**
a.wirths@student.maastrichtuniversity.nl
i6264519

Linus Jones
ln.jones@student.maastrichtuniversity.nl
i6253097

Robin Steinkühler
r.steinkuhler@student.maastrichtuniversity.nl
i6265070

Supervisor: **dr. Tom van der Zanden**
T.vanderZanden@maastrichtuniversity.nl
Department of Data Analytics and Digitalisation

Date: July 3, 2023

1 Situation

During the 2021-2022 academic year, the eLab II course received positive feedback. However, there were certain issues that hindered the learning experience, particularly in the AHA Supermarket fraud detection case. The use of simple method to generate datasets was challenging for students because the data did not accurately represent real-world scenarios. This limitation made it difficult for students to derive optimal fraud prediction algorithms.

To address these issues, our project aims to develop more advanced and detailed simulations to generate synthetic datasets that better reflect the shopping behavior of customers in the AHA supermarket. By incorporating a broader range of patterns into the simulated data and enriching the product catalog with real-world data, students will have access to a more realistic and representative dataset. This enhancement will not only make the case more engaging, but will also improve students' ability to distinguish real patterns from noise. It will also enable them to build more accurate fraud detection algorithms.

This report describes the scripts and methods used to generate the datasets.

2 Input Data/Files

2.1 Product Data

The product data was taken from the online catalog of Albert Heijn. Since the number of products is quite large, we only collected information about a certain number of products per category. The product information is stored using a file system. It can be accessed in the "product_data" directory and is organized into product categories, followed by product subcategory folders, which then contain the specific product information in JSON format. The product information includes the ID, category, subcategory, product name, unit, weight, calories, ingredients, and other characteristics of the product (vegan, lifestyle, etc.). These characteristics are used for the distance function, which determines which products are bought by the customers.

The supermarket layout, including walking distances and product categories, is based on the AH Plein 1992 in Maastricht.

2.2 Customer Types

The customer types are used to create purchase patterns within the generated purchases. They are stored in the CustomerTypes folder. Customer types can be added by simply providing a new .json file with the same structure as the other customer type files. Customer types can also be removed by simply deleting the file.

The attributes defined in the customer type file are as follows:

- **"name"**: The name of the customer type
- **"n"**: The number of different versions of this customer type (Each version will be slightly different due to the use of distributions)
- **"fraud_probability"**: The probability of committing fraud

- **"time"**: The speed of the customer (mean and std are provided for a normal distribution)
- **"amount"**: The approximate amount of products purchased (mean and std are provided for a normal distribution)
- **"recipes"**: A dictionary of recipes and the probability that this recipe is chosen
- **"attributes"**: A dictionary of attributes used to measure the similarity between a customer and the products (Mean and std are provided for a normal distribution)
  - **"Price Sensitivity"**: Indicates how sensitive the customer is to price
  - **"Meat"**: Indicates if this customer is a meat eater
  - **"Vegetarian"**: Indicates if this customer is a vegetarian
  - **"Vegan"**: Indicates if this customer is vegan
  - **"Lifestyle"**: Indicates how healthy the customer eats
  - **"Sustainability"**: Indicates how important sustainability and organic food is to the customer
  - **"Pets"**: Indicates whether or not the customer has pets
  - **"Children"**: Indicates if the customer has children or not

The values of the attributes can range from -1 to 1 (no preference to high preference). If the resulting value from the specified distribution is outside this interval, the value is automatically adjusted.

If you want to set a value for an attribute that uses a mean and a std, simply set the mean to your desired value and the std to zero. (This method is used, for example, to set meat to -1 if the customer is vegetarian/vegan).

2.3 Recipes

The recipes create association rules between products in the generated purchase datasets. Recipes are stored in the "Recipes.json" file and can be easily customized by adding a recipe name and a list of products required for the recipe. Existing recipes can also be modified or removed.

It is important that the recipes are also in the customer type files and vice versa. If this is not the case, the recipes will be ignored or an error will occur.

3 Code Architecture

The whole project is divided into several classes and scripts. Detailed descriptions of the functions and classes can be found in the files themselves. To run the whole simulation, the file "main.py" has to be executed. You can specify the desired number of purchases (supermarket visits) and the location where the files should be stored. When the script is executed, the progress bars show the status of the program.

"Purchases.py:

- The main class that reads and stores all the input files, creates the supermarket purchases itself and introduces the fraud.

"Customer.py:

- Class representing a customer
- The specific attributes of the customer are stored inside a class object.

"product_catalog.py":

- It contains functions that retrieve the product files (from the json files) and store them as a dataframe.
- This dataframe is stored in a "purchases" class object and used to generate the purchase records.

"store_layout.py":

- Contains functions to create and retrieve the layout from the AH store.
- The layout is stored as an adjacency matrix.

4 Purchase Dataset Generation

The 'Purchases' class is primarily used to simulate a customer's journey as they shop, with a focus on the customer's selection of products and their decision to exit the store. More specifically this class simulates the purchase behavior of different customer types and incorporates fraudulent behavior into the purchases.

The primary **attributes** of the class include:

- The number of purchases.
- Data on purchases, including a purchase id, product id, and timestamp.
- A list of customer indices corresponding to each purchase.
- The amount of fraud associated with each purchase, in monetary terms.
- A list of all customer objects.
- A fraud probability for each type of customer.
- A list of all products and their associated attribute averages per category and sub-category.
- Store layout information.

The **methods** included in the 'Purchases' class perform the following tasks:

- **__init__**: Initializes the class and fetches necessary information.
- **run**: Runs methods to create the fraudulent product dataset.
- **create_customer_types**: Creates n different customer types and stores them as class attributes.
- **purchase_simulation**: Generates the complete purchase dataset with n entries.

- **create_purchase**: Simulates one visit to the store by a given customer.
- **distance_category**: Calculates the distance between the customer and each product category.
- **distance_sub_category**: Calculates the distance between the customer and each product sub-category.
- **distance_product**: Calculates the distance between the customer and each product within a sub-category.
- **convert_to_probabilities**: Converts distances and times into probabilities for product/category selection and exiting.
- **introduce_fraud**: Introduces fraud based on the customer type's fraud probability.

The following describes in more detail how one visit to the store is generated in the **create_purchase** function.

1. **Category Choice:** When a customer is in the store and wants to pick a category, their decision is influenced by multiple factors.
   - `Attribute Similarity:` The customer's attributes are compared with the mean attribute values of each category, creating a "distance" or similarity measure. Smaller distances correspond to higher similarities between a customer's preference and a category.
   - `Previous Purchases:` The number of items previously bought from a category also plays a role. The more items a customer has already bought from a category, the lower the likelihood they will return to it in the same shopping trip. Therefore, the probability of choosing a category is inversely proportional to the number of items previously bought from that category. To avoid division by zero, the number of items bought starts at one.
   - `Predefined Shopping List:` If the customer is following a predefined shopping list or a recipe, the categories containing these preselected items start with a count of one while all other categories start with a count of two. This adjustment increases the initial probability of choosing categories that include items from the recipe.
   - `Time and Distance:` The time required to reach a different category from the current location in the store is also taken into account when choosing the next category. The adjacency matrix from the store layout provides this information.

   The combination of these factors is used to compute a set of probabilities that guide the customer's category choice during their shopping trip.
2. **Sub-category Choice:** Within the chosen category, the customer's attributes are compared to the average attributes of each sub-category. These distances are converted into probabilities, and a sub-category is selected based on these probabilities.
3. **Product Choice:** The customer's attributes are compared to all products in the chosen sub-category, and these distances are converted into probabilities. A product is selected based on these probabilities and added to the customer's purchases.

This entire process is repeated until the customer chooses to exit. The probability of exiting increases as the customer's shopping basket becomes more filled, ensuring the simulation does not continue indefinitely.

5 Fraud Introduction

The objective of this case is to predict fraud incidents occurring at self-checkout desks in the supermarket. Self-checkout fraud involves situations where the items scanned by a customer do not match the items placed in their shopping cart. Two well-known self-checkout fraud schemes are the "banana trick" and the "pass around trick."

The "banana trick" refers to the deceptive act of placing a different, cheaper product (sometimes a banana) in the shopping cart while scanning a more expensive item. On the other hand, the "pass around trick" involves placing a product in the shopping cart without scanning it at all.

In this case, each customer is assigned a specific fraud probability. If a customer is determined to have committed fraud, a Poisson distribution is used to determine the number of items stolen. The stolen items are attributed to either the "banana trick" or the "pass around trick" based on a probability distribution. There is a 20% probability of the "banana trick" being employed and an 80% probability of the "pass around trick" being used. These fraud schemes are randomly introduced during the simulated purchase of a customer if they are declared to have committed fraud.

6 Output Data/Files

There are four different output files that are explained below. The first one is generated once, and the other three are generated each time new purchases are made.

Ah_purchases[].csv:

- This file contains the generated customer purchases.
- Variables: Purchase ID, Product ID, Time (time until next scan).
- The student receives this file for purchase information.

Fraud_monetary[].csv:

- The file contains the amount of fraud in monetary terms (€) per purchase.
- The entry in row i is the fraud value for the purchase with purchase id i.
- This file will NOT be given to the students. It is used to evaluate the students' predictions on the test datasets.

Customer_records[].csv:

- This file contains the customer type for each purchase.
- The entry in row i is the customer type of the purchase with purchase id i.
- This file is NOT given to students. It serves as a record and as a reference for adjusting the generated data.

7 Performance Testing

The generated datasets can be tested by applying predictive models to attempt to identify fraud within these generated datasets. The performance of the predictive models gives an indication of how difficult/easy it is to identify fraud in the datasets. The "PerformaceTesting.py" and "lstm3.py" files contain the models and the script to run all the different predictive models at once. Models can be easily added, modified, or removed.