**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет прикладной математики-процессов управления**

**Программа бакалавриата**

**“Большие данные и распределенная цифровая платформа”**

**ОТЧЁТ**

**По лабораторной работе № 1**

**По дисциплине “Алгоритмы и структуры данных”**

**На тему “Генерация синтетического датасета”**

**Вариант №1**

**Студент гр. 24Б16-пу**

**Скородумов В.В.**

**Преподаватель**

**Дик А.Г.**

**Санкт-Петербург**

**2025 г.**

Оглавление

[**1.** **Цель работы** 3](#_Toc209554251)

[**2.** **Описание задачи** 3](#_Toc209554252)

[**3.** **Теоретическая часть** 3](#_Toc209554253)

[**4.** **Основные шаги программы** 3](#_Toc209554255)

[**5.** **Блок схема программы** 3](#_Toc209554256)

[**6.** **Описание программы** 4](#_Toc209554257)

[**7.** **Рекомендации пользователя** 5](#_Toc209554258)

[**8.** **Рекомендации программиста** 5](#_Toc209554259)

[**9.** **Контрольный пример** 6](#_Toc209554260)

[**10.** **Исследование** 6](#_Toc209554261)

[**Вывод** 6](#_Toc209554262)

1. **Цель работы**

Цель работы — описать и продемонстрировать программу генерации синтетического датасета кассовых чеков, имитирующую реальные данные, основанную на конфигурационном файле и словарях CSV. Программа формирует правдоподобные данные о покупках, загружает подготовленные словари и проверяет корректность входных данных.

1. **Описание задачи**

1) Спроектировать структуру датаеста.

2) Разработать алгоритм генерации (Создать правила и зависимости между

признаками и целевой переменной, чтобы данные были правдоподобными).

3) Реализовать программу генератор.

4) Протестировать результат (в результате должен сгенерирован быть файл

расширения .xlsx).

5) Написать отчет (файл расширения .doc или .docx)

1. **Теоретическая часть**

Генерация основана на случайных выборках с учётом весов вероятностей и нормальном распределении цен в пределах базового диапазона категории. Используются библиотеки pandas, numpy, pyyaml, datetime, random и Path. Pandas – основная библиотека в программе, используются классы Series и DataFrame для работы с табличными данными: хранение в памяти, фильтрация, сборка итогового датасета. Numpy – используется только метод random() для Гауссова распределения в указанном диапазоне цен. Pyyaml – библиотека для парсинга файлов расширения .yml и .yaml, в программе используется для чтения конфигурационного файла программы (config.yml). Стандартная библиотека datetime используется для удобной работы с форматами даты и времени. Random – стандартная библиотека для генерации случайности. В программе используются методы randint() для генерации случайного числа в заданном диапазоне и choices() для выборки с учетом весов. Path – стандартная библиотека, используется для корректной работы с путями к файлам внутри директории проекта.

1. **Основные шаги программы**

1. Загрузка конфигурации и словарей при помощи библиотек pyyaml и pandas.  
2. Валидация входных распределений вероятностей (банки/сети).  
3. Генерация чеков: выбор магазина, даты-времени, категорий и брендов, платёжной сети и номера карты; расчёт цен.  
4. Сбор строк в единый датасет до достижения минимального объёма.  
5. Сохранение результата в .xlsx в директории вывода.  
6. Автоматический запуск через скрипт сборки и запуска.

1. **Блок схема программы**

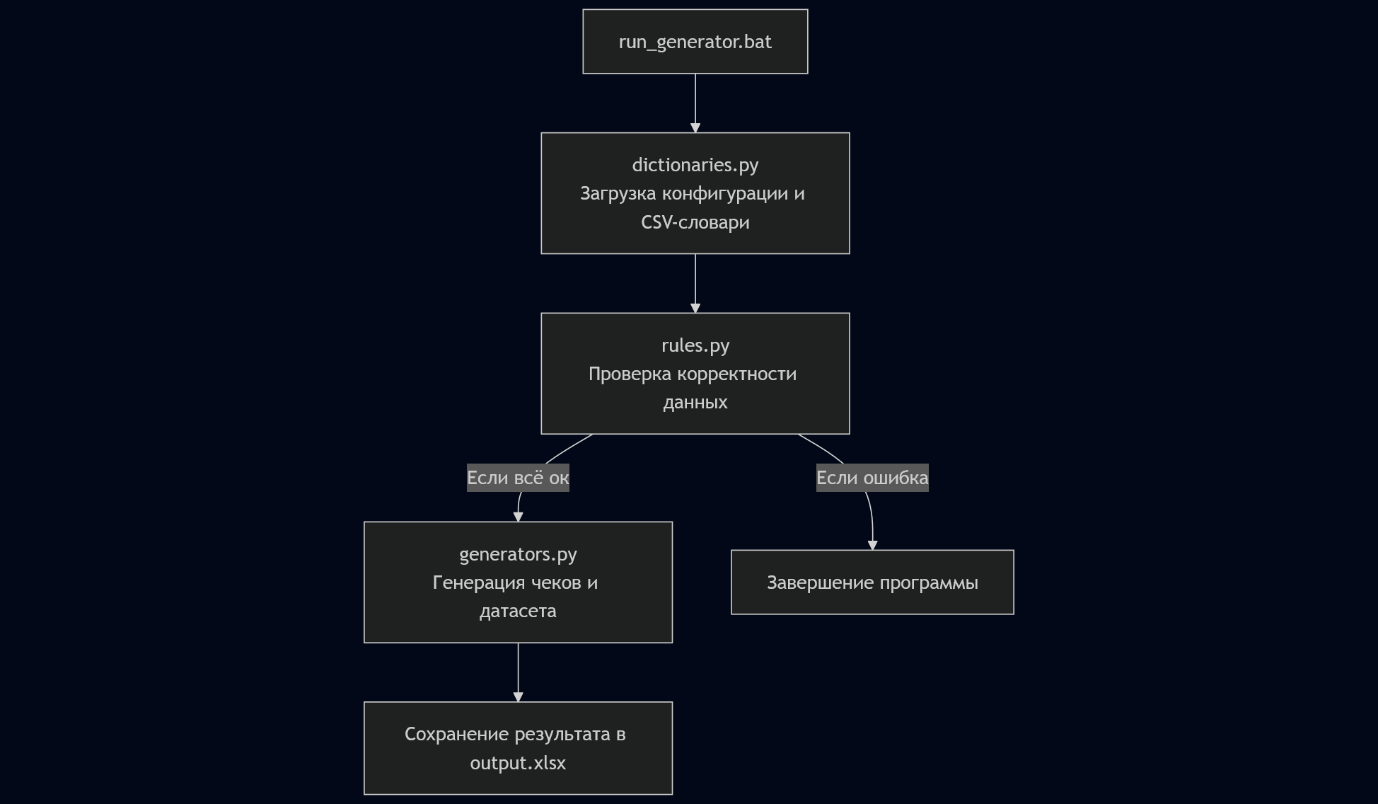


Рис. 1 Блок-схема основной программы

1. **Описание программы**

Программная реализация состоит из модулей: dictionaries.py (загрузка словарей и конфигурации), rules.py (валидация входных данных), generators.py (генерация чеков и итогового датасета). Ниже приведена таблица всех функций.

Таблица 1 Описание модулей и методов программы

|  |  |  |
| --- | --- | --- |
| Метод/Функция | Модуль | Назначение |
| get\_data\_path(file) | dictionaries.py | Путь к файлу данных во входной директории. |
| parse\_stores() | dictionaries.py | Загрузка DataFrame магазинов из stores.csv. |
| parse\_brands() | dictionaries.py | Загрузка DataFrame брендов из brands.csv. |
| parse\_categories() | dictionaries.py | Загрузка DataFrame категорий из categories.csv. |
| parse\_banks() | dictionaries.py | Загрузка DataFrame банков из banks.csv. |
| parse\_networks() | dictionaries.py | Загрузка DataFrame платёжных сетей из networks.csv. |
| read\_config() | dictionaries.py | Чтение config.yml, возврат параметров: (min\_rows, date\_start, date\_end, prod\_min, prod\_max, input\_dir, output\_dir). |
| get\_out\_path(out\_dir, name) | dictionaries.py | Формирование пути для сохранения результата. |
| check\_banks\_weights(banks) | rules.py | Сумма вероятностей банков по каждой сети равна 1. |
| check\_networks\_weights(networks) | rules.py | Сумма вероятностей сетей равна 1. |
| full\_validation() | rules.py | Комплексная проверка входных данных, True/False. |
| get\_time(work\_open, work\_close) | generators.py | Случайное время (HH:MM) в границах рабочего дня магазина. |
| get\_date(date\_start, date\_end) | generators.py | Случайная дата (YYYY-MM-DD) из заданного диапазона. |
| get\_category\_from\_store\_type(store\_type, categories) | generators.py | Выбор категории по типу магазина с учётом весов; возвращает имя, id и базовый ценовой диапазон. |
| get\_brand\_from\_category\_name(category\_id, brands) | generators.py | Выбор бренда в категории и его ценовой фактор. |
| get\_price(base\_price\_min, base\_price\_max, factor) | generators.py | Цена из усечённого нормального распределения, умноженная на фактор бренда. |
| get\_network(networks) | generators.py | Выбор платёжной сети по вероятностям. |
| get\_card\_number(banks, network) | generators.py | BIN-префикс банка по сети + случайные цифры = номер карты. |
| receipt\_generate(...) | generators.py | Генерация строк и столбцов одного чека. |
| dataset\_generate(strings) | generators.py | Повторная генерация чеков до требуемого числа строк; сохранение output.xlsx при запуске модуля как скрипта. |

# **Рекомендации пользователя**

Для запуска:  
1) Убедитесь, что в указанной в конфигурационном файле директории находятся файлы stores.csv, categories.csv, brands.csv, banks.csv, networks.csv с такими же столбцами как в примере. Файл config.yml должен находиться в корне проекта.  
2) Запустите скрипт run\_generator.bat — он установит зависимости из requirements.txt и выполнит генерацию.  
3) Итоговый файл output.xlsx появится в папке, указанной в конфигурации (paths.output\_dir).

# **Рекомендации программиста**

Для корректного функционирования программы, убедитесь, что выполнены следующие пункты:

• Python 3.10+.  
•Установлены библиотеки numpy, pandas, pyyml, openpyxl, указанные в requirements.txt.  
• Форматы CSV должны соответствовать ожидаемым столбцам (например, для категорий: category\_id, category\_name, store\_types, base\_price\_min, base\_price\_max, pick\_weight) с запятой в качестве разделителя.  
• В banks.csv веса по каждой сети суммируются до 1; в networks.csv общая сумма равна 1.  
• Пути и параметры генерации коректно указаны в config.yml (settings и paths).

# **Контрольный пример**

1) Запуск run\_generator.bat.

2) Проверка наличия виртуального окружения в корне проекта и необходимых, создание окружения и загрузка библиотек в случае отсутствия.

3) Загрузка входных данных.

4) Проверка корректности словарей.

5) Генерация каждого чека датасета и объединение их в итоговую таблицу.  
6) В директории вывода появляется output.csv с полями: store\_name, date-time, coordinates, categories, brands, price, cards\_number, number\_of\_products, receipt\_id, total\_cost.

# **Вывод**

В рамках данной работы были изучены библиотеки pandas, numpy, datetime и реализованы функции генерации синтетического датасета.

В итоге было написано полностью работоспособное, настраиваемое програмное обеспечение для генерации датасета по заданным словарям.