Hackaton Coppel

Erick Gonzalez Parada
Emiliano Ruiz Plancarte
Miguel Garcia Diaz de Rivera

March 28, 2025

Abstract

This report analyzes a dataset provided by Coppel regarding customer service queues and times. Using data manipulation techniques, visualization, and queue theory, we propose a solution to optimize customer service attention by prioritizing elderly customers and customers who have waited beyond a threshold time. The implementation involves Python's data science stack and a custom-built priority queue system.

Keywords: Queue, DataFrame, Priority Queue, Waiting Time, Customer Service.

1 Introduction

Theoretical framework

Queueing Theory is the mathematical study of waiting lines, or queues, and has been widely used to model customer service systems [3]. It provides tools to analyze the behavior of customers and servers, helping to understand key metrics such as waiting times, service durations, and queue lengths. In practical settings, especially in customer-facing businesses like Coppel, optimizing these queues is crucial to enhance customer satisfaction and reduce operational bottlenecks.

In traditional queueing models, such as the M/M/1 model, customers are served on a first-come, first-served basis. However, real-life situations often require modifications to handle priority cases. For example, elderly customers may require special attention, or some customers may have waited excessively and deserve to be prioritized.

Our approach integrates this fundamental knowledge by constructing a **dual-priority queueing system** where:

1. Elderly customers are automatically given higher priority.
2. Non-elderly customers who have waited beyond a threshold are promoted to high priority.

This methodology aligns with the goal of improving fairness and efficiency in service processes. By relying on Python's *pandas* library for data manipulation [1] and using the *heapq* module to implement the queue structure, we can simulate and evaluate the impact of this priority mechanism.

The dataset itself was structured into a *DataFrame* [2], which facilitated the analysis of arrival times, service times, and customer segmentation. The priority queue implementation directly leveraged these insights to emulate more realistic queue behavior. The integration of queueing theory with actual data ensures that the solution is not only theoretically sound but also grounded in the operational reality of Coppel's service centers.

Goal

The primary goal of this project is to improve customer service queue management at Coppel branches by analyzing historical service data and designing a priority queue system to optimize service order considering both elderly customers and long waiting times.

Materials

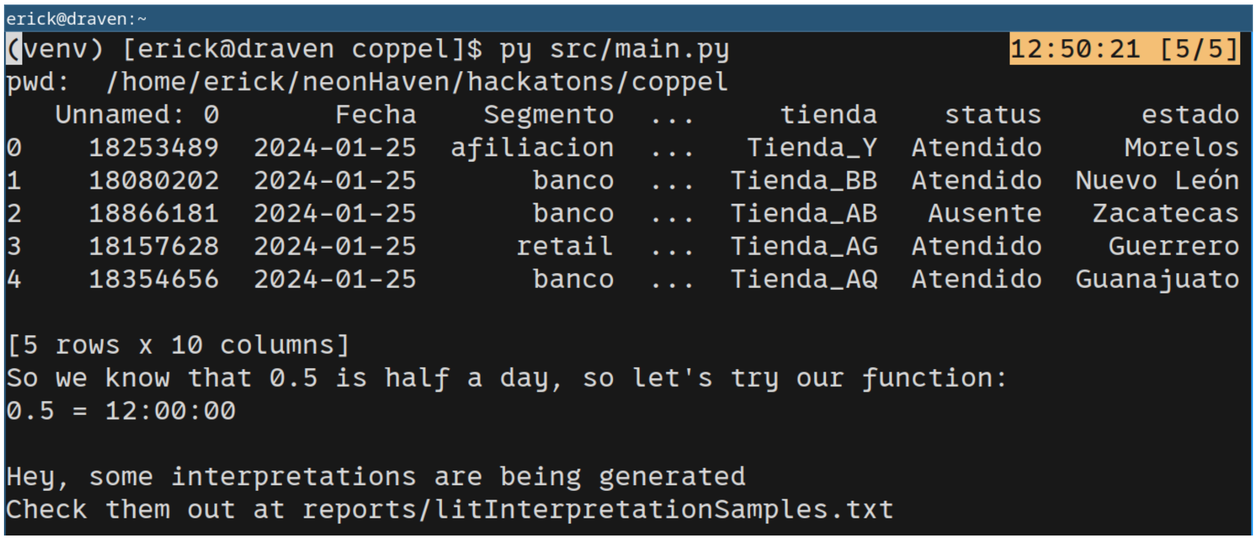
- *Python 3* - Programming language used for analysis and implementation.
- *Pandas* - For data manipulation and exploration [1].
- *Matplotlib* - For visualization.
- *Heapq* - For implementing the priority queue.

What to do

- Analyze the dataset provided by Coppel.
- Study customer arrival, service, and exit times.
- Detect potential bottlenecks.
- Propose a queue system prioritizing elderly people and customers with long waiting times.

2 Methodology

Dataset Overview



```
erick@draven:~
(venv) [erick@draven coppel]$ py src/main.py                    12:50:21 [5/5]
pwd:  /home/erick/neonHaven/hackatons/coppel
   Unnamed: 0       Fecha     Segmento  ...     tienda    status       estado
0    18253489  2024-01-25  afiliacion  ...    Tienda_Y  Atendido      Morelos
1    18080202  2024-01-25       banco  ...   Tienda_BB  Atendido  Nuevo León
2    18866181  2024-01-25       banco  ...   Tienda_AB   Ausente    Zacatecas
3    18157628  2024-01-25      retail  ...   Tienda_AG  Atendido     Guerrero
4    18354656  2024-01-25       banco  ...   Tienda_AQ  Atendido   Guanajuato

[5 rows x 10 columns]
So we know that 0.5 is half a day, so let's try our function:
0.5 = 12:00:00

Hey, some interpretations are being generated
Check them out at reports/litInterpretationSamples.txt
```

Figure 1: dataframe head

The dataset provided by Coppel i.e figure 1, loaded from the CSV file `v2.csv`, consists of **289,231** records with the following key columns:

- `customer_id` – Unique customer identifier.
- `segment` – Service type (*banco*, *retail*, *afiliacion*, etc.).
- `isElderly` – Indicates if the customer is elderly (boolean).
- `arrivalTime`, `serviceStartTime`, `serviceEndTime` – Time fields expressed as fractions of a day.

Exploratory analysis shows:

- The majority of services belong to the *banco* and *retail* segments.
- Around 12% of customers are elderly.
- Average waiting time ranges between 7 to 10 minutes depending on the segment.
- Service durations follow a slightly skewed distribution with most services taking between 5 and 15 minutes.

This information is crucial to justify the design of our dual-priority system, where elderly customers and long-waiting customers are handled more efficiently.

We used Python's Pandas library to load and preprocess the dataset. Missing values were identified, and new features such as *waitTime*, *serviceTime*, and *totalTime* were computed. Exploratory Data Analysis (EDA) was performed to understand distributions of segments, service statuses, and store performance. Finally, we implemented a custom priority queue simulating a real queue at a service branch.

3 Proposed Solution

The proposed solution is based on queueing theory [3]. The system prioritizes:

1. Elderly customers (marked with an `isElderly` attribute).
2. Customers who have waited longer than a specified threshold.

This dual-priority system helps ensure fair and efficient service. We implemented it using Python's `heapq` module, which allows dynamic updating of customer priorities based on waiting time.

Interpretation

Each row in the dataset represents a customer. We converted numerical time formats into human-readable timestamps. For instance, if a customer arrives at `0.5` (fraction of a day), it corresponds to 12:00 PM.

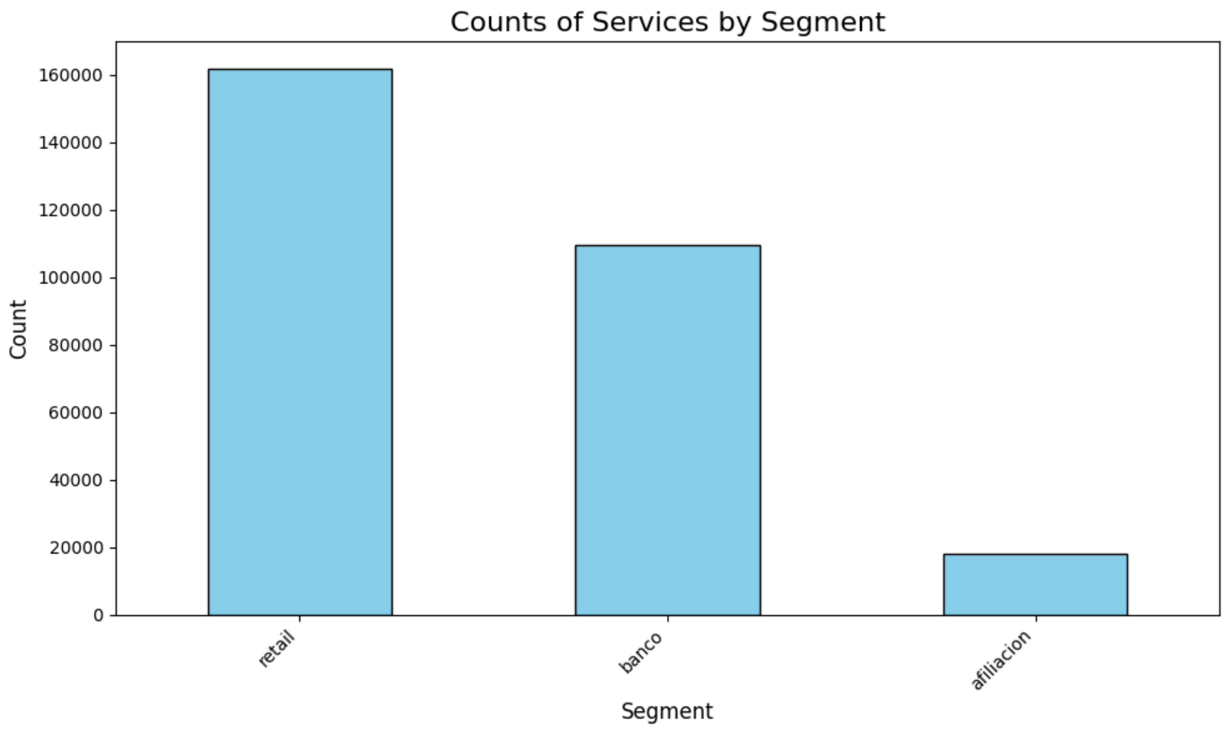


Figure 2: Count of Services by Segment

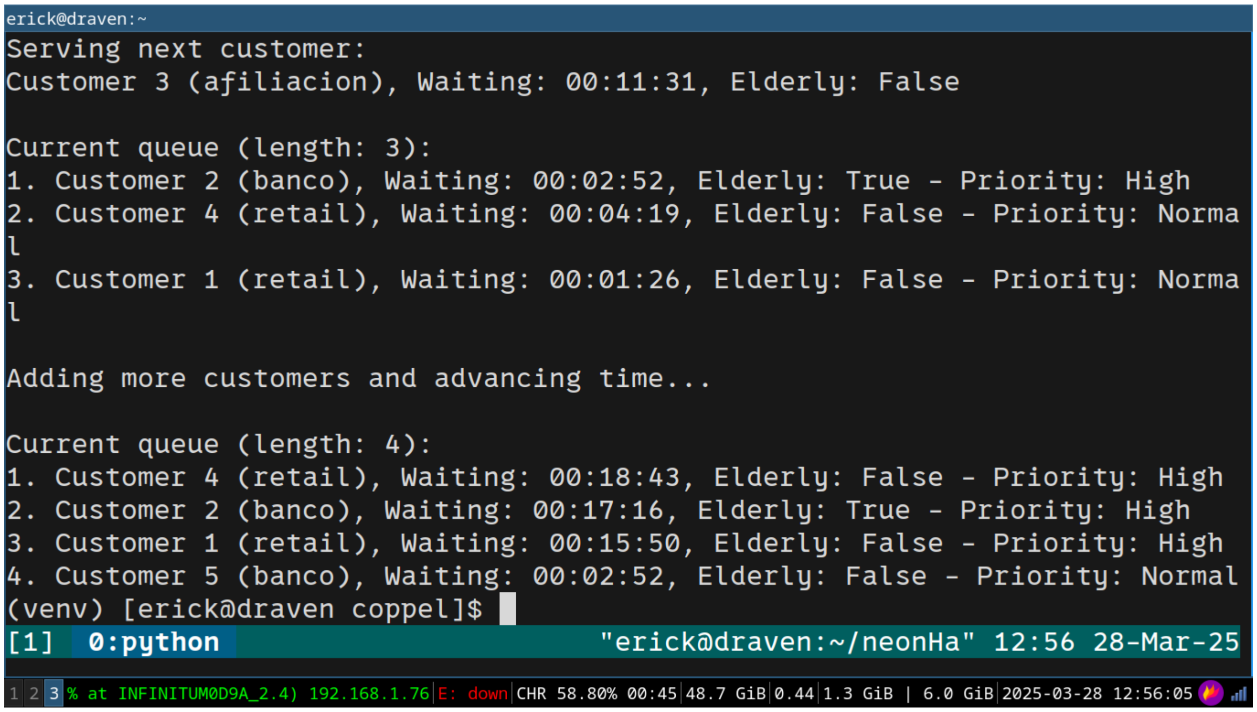
The data shows that:

- The dataset consists of **289,231** records.
- Average customer wait time is roughly 7-10 minutes depending on the segment.
- According to figure 2 the most common segments are *banco*, *retail*, and *afiliacion*.

Priority Queue Simulation

The queue dynamically adjusts as customers are added:

- If a customer is elderly, they automatically get high priority.
- If a non-elderly customer waits beyond the defined threshold (e.g., 10 minutes), they are escalated to high priority.
- The queue maintains a balanced service between elderly customers and those who waited too long.



```
erick@draven:~
Serving next customer:
Customer 3 (afiliacion), Waiting: 00:11:31, Elderly: False

Current queue (length: 3):
1. Customer 2 (banco), Waiting: 00:02:52, Elderly: True - Priority: High
2. Customer 4 (retail), Waiting: 00:04:19, Elderly: False - Priority: Norma
l
3. Customer 1 (retail), Waiting: 00:01:26, Elderly: False - Priority: Norma
l

Adding more customers and advancing time...

Current queue (length: 4):
1. Customer 4 (retail), Waiting: 00:18:43, Elderly: False - Priority: High
2. Customer 2 (banco), Waiting: 00:17:16, Elderly: True - Priority: High
3. Customer 1 (retail), Waiting: 00:15:50, Elderly: False - Priority: High
4. Customer 5 (banco), Waiting: 00:02:52, Elderly: False - Priority: Normal
(venv) [erick@draven coppel]$
[1]  0:python                     "erick@draven:~/neonHa" 12:56 28-Mar-25
1 2 3 % at INFINITUM0D9A_2.4) 192.168.1.76|E: down|CHR 58.80% 00:45|48.7 GiB|0.44|1.3 GiB | 6.0 GiB|2025-03-28 12:56:05
```

Figure 3: Fragment of output of solution.py

This solution reduces waiting time unfairness and can be adapted in real-time to suit Coppel's service flow.

4 Access to Source Code

The source code for this project is publicly available on GitHub at `https://github.com/HugeErick/CoppelHackaton`. The repository includes:

- Data processing scripts.
- Priority queue implementation.
- Visualization tools.
- Sample interpretations.

Follow the instructions in the README to reproduce the results.

5 Conclusions

This project successfully demonstrates how data analysis combined with queueing theory can improve customer service. By prioritizing elderly customers and customers who have waited excessively, service becomes fairer and more efficient. The system is adaptable and could be implemented in Coppel's real operations after testing.

References

[1] Pandas. (n.d.). PyPI. Retrieved March 28, 2025, from `https://pypi.org/project/pandas/`

[2] Pandas DataFrame. (2024, June 13). GeeksforGeeks. Retrieved from `https://www.geeksforgeeks.org/python-pandas-dataframe/`

[3] Queuing theory. (2024, May 27). GeeksforGeeks. Retrieved from `https://www.geeksforgeeks.org/queuing-theory/`