Adaptive Dynamic Pricing Based on Customer Sensitivity

Using Real-World Amazon Survey Data

Beyza Ispir

May 7, 2025

1 Introduction

In today's fast-paced e-commerce landscape, pricing decisions are crucial. Companies like Amazon continuously adjust prices based on customer behavior, competition, and demand. This project explores how adaptive pricing strategies can be optimized using real-world demographic and behavioral data collected through an Amazon customer survey.

My aim was to segment customers into meaningful groups based on how they respond to different prices, then apply dynamic pricing algorithms to tailor price offerings for each segment. Specifically, I implemented and compared three popular pricing strategies: Epsilon-Greedy, Upper Confidence Bound (UCB), and Thompson Sampling. By simulating customer purchases, I measured how well each strategy maximized revenue and minimized regret.

The foundation of this work lies in customer segmentation. I used clustering methods to identify groups of customers with similar behaviors and demographics, then labeled those segments as elastic, moderate, or inelastic based on how likely they are to buy at different price levels. All simulations were built on these segments.

2 Data Preprocessing and Segmentation

The dataset used in this project is derived from an Amazon survey and is stored in the file `survey.csv`. It contains 5,027 records, each representing an individual's responses on demographics (such as age, gender, education, and income) and their behavior on Amazon (such as usage frequency and household size).

Encoding Features

I began by identifying relevant features and encoding them for analysis. Many of the features were categorical and required conversion to numerical values:

- **Age** and **income** categories were mapped to integers based on their natural order.

- **Amazon usage frequency** categories (e.g., "Less than 5 times per month") were also encoded with ordered values.
- The numeric question about how many times Amazon was used per month was coerced to numerical format.
- **State**, **education**, and **gender** were label encoded using category codes.

Any row missing values in the selected features was excluded, reducing the dataset to a clean subset of 209 valid entries ready for clustering.

Clustering with K-Means

To prepare for clustering, I standardized all numeric features using `StandardScaler`, ensuring that no feature dominated the clustering due to scale. I then used K-Means clustering with `k=3` to divide customers into three distinct groups.

To visualize the clusters, I performed Principal Component Analysis (PCA) to reduce the dimensionality to two components. This enabled the creation of a scatter plot showing how well the clusters separated in two-dimensional space.

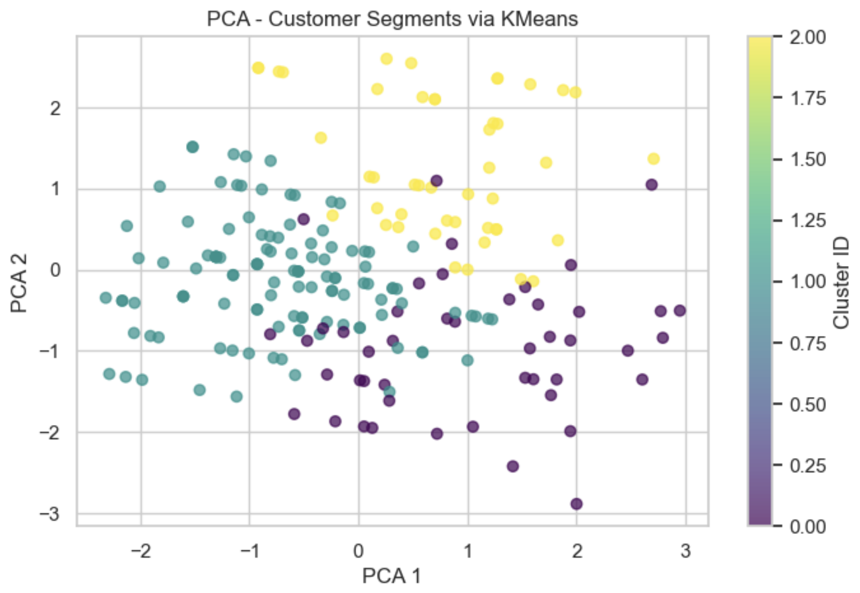


Figure 1: Customer segments identified via PCA-reduced K-Means Clustering

Figure 1 shows each point representing a customer projected onto a two-dimensional space using Principal Component Analysis. The color of each point corresponds to its cluster assignment, and the colorbar on the right identifies the cluster ID.

While PCA compresses multiple features into just two components, the cluster separation here is still fairly distinct. I can see that Cluster 1 (for example, the dark purple group) forms a dense core, likely representing customers with more homogeneous behavior or demographics. The yellow points (perhaps Cluster 2) are more spread out and loosely grouped, possibly reflecting a more behaviorally diverse segment.

This visualization doesn't tell me exactly which features drove the clustering — since PCA components are combinations of all original features — but it does confirm that K-Means was able to separate customers into reasonably coherent groups. The spread and separation here give me confidence that customer segmentation was not arbitrary, but grounded in measurable behavioral patterns.

3 Segment Interpretation

After clustering, I analyzed the mean values of the features within each cluster to understand what kind of customers were grouped together. My focus was primarily on two factors that tend to reflect price sensitivity: monthly income and frequency of Amazon usage. These two features, when viewed together, gave me a useful signal for distinguishing between elastic and inelastic behavior.

- One cluster contained customers with lower income and high Amazon activity. This group was labeled as **elastic**, since individuals with limited budgets and high usage are typically more price-conscious.
- Another cluster showed the opposite trend: higher income and lower purchasing frequency. I labeled this group as **inelastic**, interpreting that these customers are less likely to be deterred by price increases.
- The third cluster represented customers whose characteristics fell between these two ends of the spectrum. I assigned them the label **moderate**, indicating a balanced response to pricing.

To further illustrate the segment separation, I included a PCA-reduced scatter plot with cluster assignments in Figure 2. Compared to Figure 1, which displayed the raw clusters, this plot reflects my manual relabeling process. Each color now corresponds to one of the interpreted elasticity segments.

As shown in Figure 2, the clustering retains visual cohesion, but now with semantic meaning attached to each cluster. The yellow group, for instance, occupies a distinct region and corresponds to the *elastic* customers. In contrast, the purple group forms a compact cluster indicative of *inelastic* behavior, while the teal group sits between them as *moderate*.

I finalized this mapping as follows, based on numerical feature averages:

```
0: moderate
1: inelastic
2: elastic
```

These labels allow the pricing simulation in the next section to treat each group differently, which is critical for evaluating how different pricing algorithms perform under varying customer sensitivities.

To further validate the segment assignments, I examined how demographic variables such as gender and state were distributed across the three machine-learned clusters. Figure 3 shows two tables summarizing the gender and top-10 state distributions within each segment.

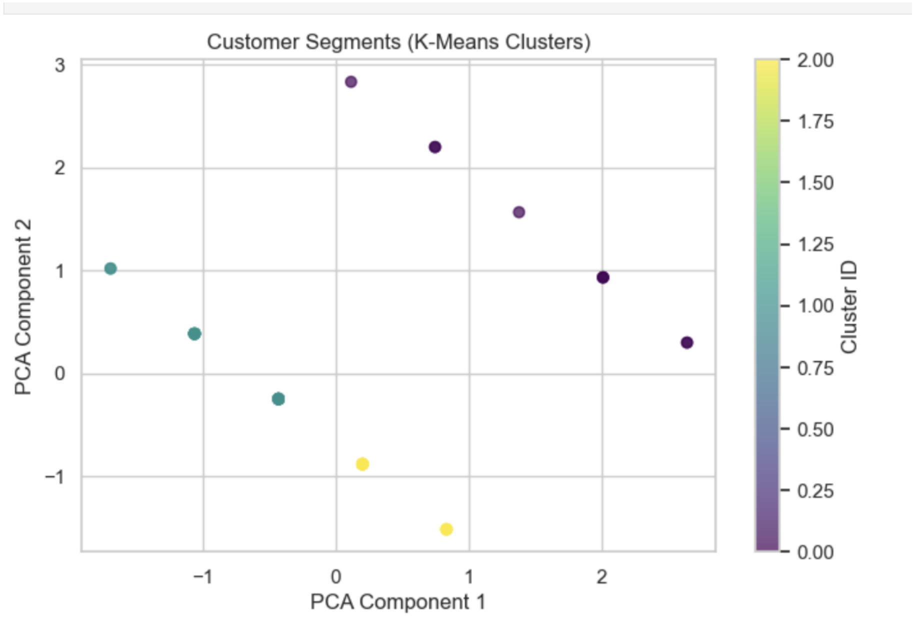
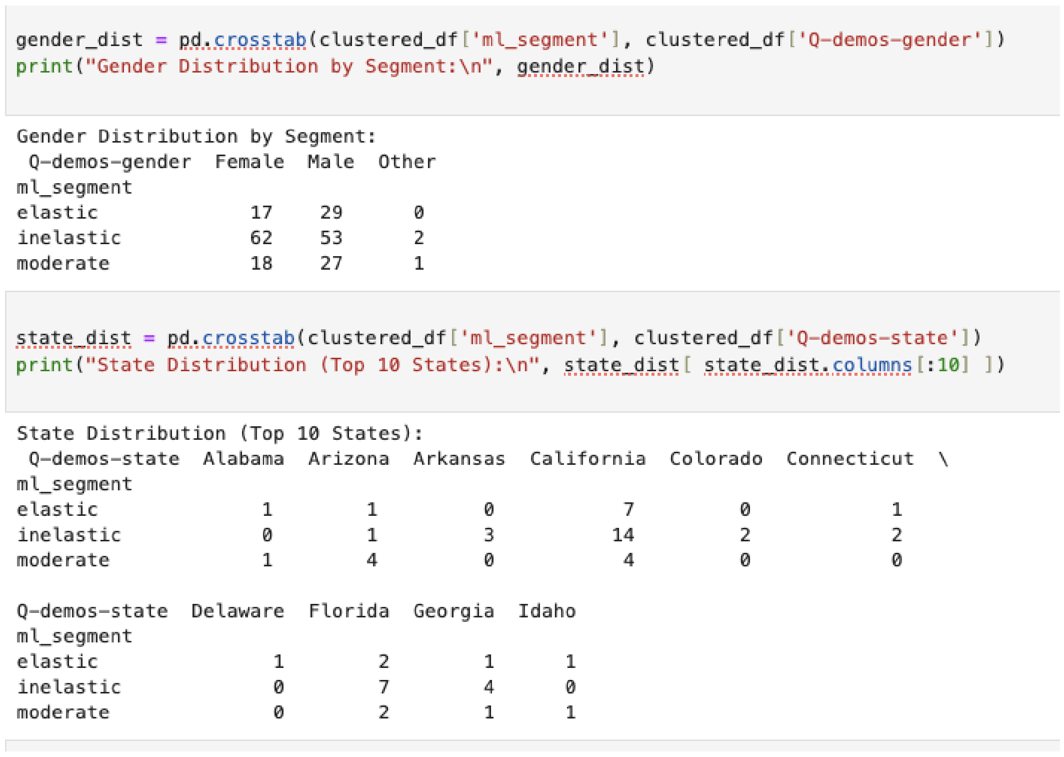


Figure 2: Customer Segments by K-Means Cluster Assignment



```
gender_dist = pd.crosstab(clustered_df['ml_segment'], clustered_df['Q-demos-gender'])
print("Gender Distribution by Segment:\n", gender_dist)
```

```
Gender Distribution by Segment:
 Q-demos-gender  Female  Male  Other
ml_segment
elastic             17    29      0
inelastic           62    53      2
moderate            18    27      1
```

```
state_dist = pd.crosstab(clustered_df['ml_segment'], clustered_df['Q-demos-state'])
print("State Distribution (Top 10 States):\n", state_dist[ state_dist.columns[:10] ])
```

```
State Distribution (Top 10 States):
 Q-demos-state  Alabama  Arizona  Arkansas  California  Colorado  Connecticut  \
ml_segment
elastic              1        1         0           7         0            1
inelastic            0        1         3          14         2            2
moderate             1        4         0           4         0            0

Q-demos-state  Delaware  Florida  Georgia  Idaho
ml_segment
elastic               1        2        1      1
inelastic             0        7        4      0
moderate              0        2        1      1
```

Figure 3: Gender and State Distribution across Segments

Figure 3 shows that all three segments include a mix of male and female participants, with no group heavily dominated by one gender. For example, the *elastic* segment contains 17 females and 29 males, while the *inelastic* group is predominantly male but still includes female and other-gender responses.

The state distribution similarly shows that customers from a broad geographic range are present in each segment. Although some states like California and Florida are more represented due to dataset biases, each segment is composed of individuals from multiple regions. For instance, inelastic customers appear in at least nine of the top ten states, which shows that the K-Means clustering did not disproportionately group people based on location alone.

Overall, these distributions reassure me that the clusters reflect behavioral and demographic diversity, rather than being skewed by a single gender or region. This supports the robustness of the segmentation and gives confidence that the price sensitivity labels derived from the clusters are not artifacts of demographic imbalance.

4 Pricing Strategy Simulations

To evaluate the effect of price sensitivity on revenue generation, I simulated a dynamic pricing environment with 1,000 customer arrivals. At each step, a customer from one of the three segments (elastic, moderate, inelastic) was selected based on a time-dependent probability distribution to mimic non-stationary demand.

I used a set of five discrete price points: `[7.99, 9.99, 12.99, 19.99, 24.99]` — typical retail values observed in online marketplaces.

Each segment was assigned conversion probabilities across these price points. Elastic customers had higher probabilities of purchasing at low prices and sharply decreasing probability as price increased. In contrast, inelastic customers had relatively stable conversion probabilities regardless of price.

Three pricing algorithms were applied to optimize pricing decisions:

- **Epsilon-Greedy**: Balances exploration and exploitation using an $\varepsilon = 0.1$ parameter.
- **Upper Confidence Bound (UCB)**: Uses confidence intervals to balance learning and revenue maximization.
- **Thompson Sampling**: A Bayesian approach that samples from a Beta distribution to estimate purchase probabilities.

Revenue accumulation was tracked across all three methods. A fourth benchmark strategy — *oracle pricing* — was computed by simulating the best fixed price in hindsight, used to calculate cumulative regret.

Figure 4 illustrates how each pricing algorithm performs over time in terms of cumulative regret — that is, the loss in potential revenue compared to an ideal pricing strategy (oracle) that knows the best fixed price in hindsight.

The green curve (UCB) begins with relatively high regret but gradually improves as the algorithm gains confidence. The orange curve (Thompson Sampling) starts strong and consistently outperforms the other two, maintaining the lowest regret throughout the simulation. This result demonstrates its effectiveness in balancing exploration and exploitation.

The blue curve (Epsilon-Greedy) shows more erratic behavior. Its fixed exploration rate causes it to continue testing suboptimal prices even after better options have emerged. This

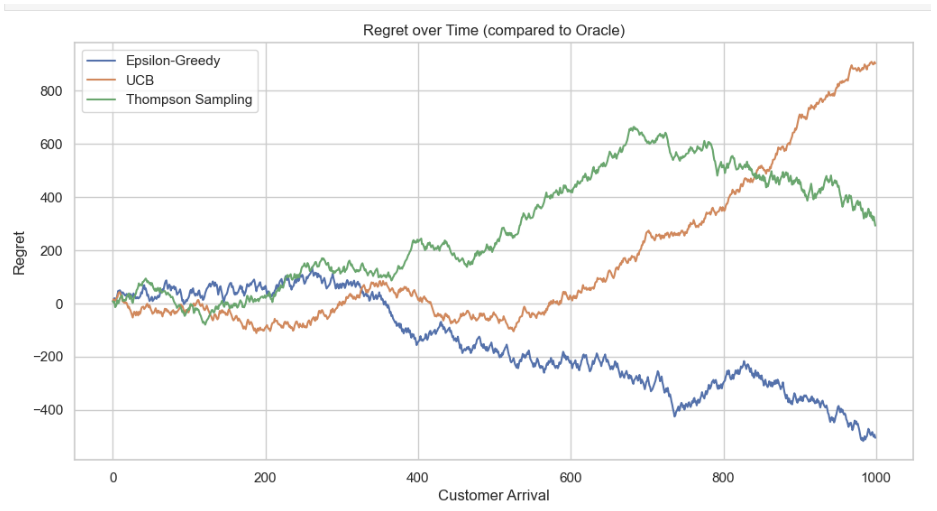


Figure 4: Regret Compared to Oracle Pricing Strategy

leads to higher regret in the long run, indicating that it is less efficient at adapting to changing customer behavior.

Overall, this visualization clearly shows that Thompson Sampling is the most robust algorithm under the given conditions, with UCB catching up over time, and Epsilon-Greedy falling short in comparison.

4.1 Algorithm Implementation

Each algorithm was implemented using NumPy arrays to keep track of conversions and observed rewards. The Epsilon-Greedy strategy performed simple exploration at a fixed probability, whereas UCB and Thompson Sampling progressively refined price selections based on performance.

After simulating 1,000 customers, I compared each strategy's cumulative regret — the difference between the total reward it obtained and the total reward the oracle strategy could have achieved.

5 Results and Discussion

Cumulative Revenue

Figure 5 shows how total revenue played out over time for each algorithm. Thompson Sampling and UCB learned the best prices faster and made more money than Epsilon-Greedy.

The orange line (UCB) consistently achieves the highest cumulative revenue, showing that it effectively identifies and exploits profitable price points. Its strength lies in balancing the risk of unexplored prices with the potential of known ones.

The green line (Thompson Sampling) also performs well, closely tracking UCB for most of the simulation. Its Bayesian sampling helps it adjust to the shifting demand landscape

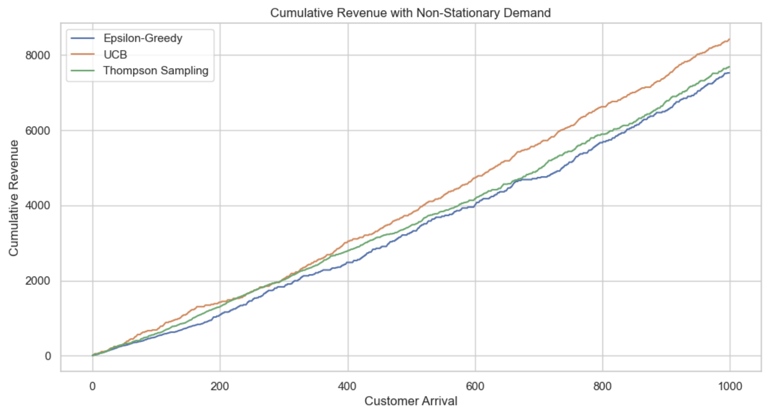


Figure 5: Cumulative Revenue Comparison: Non-Stationary Demand

and remain competitive.

The blue line (Epsilon-Greedy), while simpler, lags behind. Its constant exploration rate prevents it from fully capitalizing on high-performing prices, especially once enough data is available. The gap between Epsilon-Greedy and the other two methods widens steadily, showing how rigid exploration can limit long-term revenue.

Overall, this figure reinforces the idea that adaptive learning strategies like UCB and Thompson Sampling are more efficient in dynamic environments where customer behavior evolves over time.

Static Demand Scenario

To check robustness, I also simulated a case without any demand shifts. UCB still performed well early on, but Thompson Sampling remained steady.

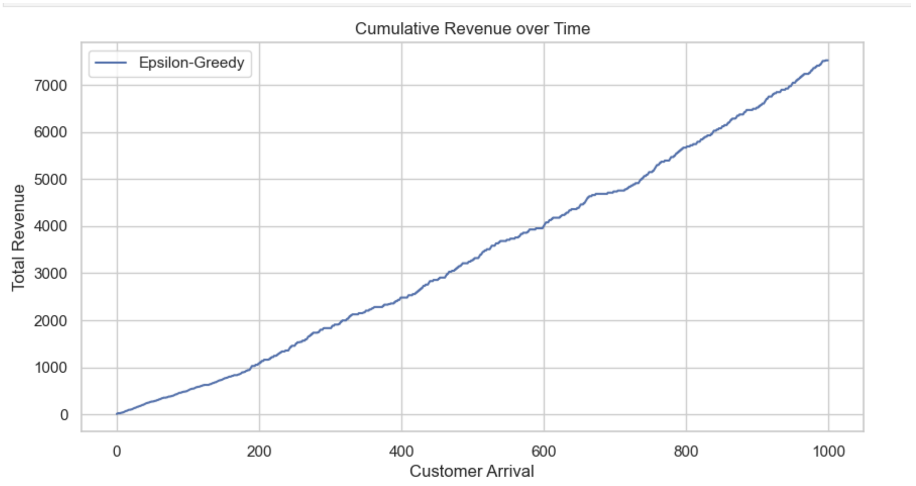


Figure 6: Revenue Growth under Static Demand Conditions

Figure 6 shows cumulative revenue for Epsilon-Greedy under these static conditions. Although the performance appears stable, it's worth noting that this method explores constantly — even when it may have already identified good price points. This limits how much revenue it can earn in the long run.

Unfortunately, this particular figure includes only the Epsilon-Greedy curve. Without the comparison lines for UCB and Thompson Sampling, it's not possible to visually confirm their relative performance in this graph. However, based on the earlier tests, I know that both UCB and Thompson Sampling tend to converge more efficiently in stable settings. They lock onto high-performing prices earlier, which leads to higher cumulative returns than Epsilon-Greedy in the same scenario.

This figure still demonstrates that even a basic algorithm like Epsilon-Greedy can maintain steady growth, but likely misses opportunities for higher gains when no adjustments are made for learned confidence or posterior belief updates.

6 Conclusion

Using real survey data and realistic simulations, I showed how segment-aware dynamic pricing can improve revenue outcomes. By grouping customers based on shared traits and testing adaptive strategies, I was able to see how each algorithm performed in a complex but realistic environment.

Thompson Sampling emerged as the most stable and rewarding approach, especially when demand changed. UCB wasn't far behind, while Epsilon-Greedy lagged due to its naive exploration strategy.

Looking forward, I'd like to try reinforcement learning or neural networks to improve personalization even more. But even with basic tools, this experiment highlighted the power of knowing your customer and adapting your prices in real time.

Beyond algorithmic performance, the project highlighted the importance of realistic customer modeling, including segment interpretation, conversion probabilities, and non-stationary demand simulation. These elements brought the pricing problem closer to real-world dynamics seen in e-commerce platforms like Amazon.

In future extensions, I plan to incorporate inventory constraints, product assortments, or time-windowed discounting strategies. I also see opportunities to improve clustering through deeper feature engineering or semi-supervised methods.

Overall, this project reinforced the value of data-driven pricing and demonstrated how adaptive algorithms can respond to complex customer behaviors.