OPTIMIZATION I

PROJECT 3 – NON-LINEAR PROGRAMMING

Newsvendor Optimization with Rush & Disposal Costs

$$\max_q E[p\min(q, D) - qc]$$

$$\max_q \frac{1}{n}\sum_{i=1}^{n}(p\min(q, D_i) - qc).$$

$$\max_{q,h} \frac{1}{n}\sum_{i=1}^{n} h_i$$

s.t.

$$h_i \le pD_i - qc$$
$$h_i \le pq - qc$$
$$h_i \ge -\infty$$

$$\max_q \frac{1}{n}\sum_{i=1}^{n}(pD_i - qc - g(D_i - q)^+ - t(q - D_i)^+)$$

Team G4

Abhay Puri, Abhiroop Kumar, Ethan Davenport, Liam Thompson

I. Executive Summary

This report applies optimization techniques to determine profit-maximizing printing and pricing decisions for a publishing company. Historically, our organization has relied on a **classic Newsvendor model** to set the print quantity for a title with uncertain demand. While effective in simple cases, this traditional NV approach:

- Assumes **price is fixed**
- Ignores **price elasticity of demand**
- Does not explicitly model **rush printing** or **disposal costs**

In this project, we have built a more realistic and powerful optimization framework:

1. A **regression-based demand model** that captures how price changes affect demand
2. A **linear program (LP)** for fixed-price decisions with rush/disposal
3. A **quadratic program (QP)** that jointly optimizes **price and quantity**
4. A **bootstrap analysis** with 4,000 data resamples
5. A comprehensive **managerial comparison** between the legacy NV model and the extended model

Key Outcomes

- **Standard NV (p = 1):**
  - $(q^* \approx 472)$
  - Expected Profit ≈ **231.48**
- **Joint Price–Quantity Optimization:**
  - $(p^* \approx 0.954)$
  - $(q^* \approx 535.29)$
  - Expected Profit ≈ **234.42**
- **Profit Improvement:**
  - $[234.42 - 231.48 = \mathbf{2.94} \text{ units}]$
    ≈ **1.2–1.5% improvement**
- **Bootstrap Analysis (4,000 replications):** The optimal pricing, quantity, and profit distributions are **tight and stable**, confirming high model robustness.

This document summarizes each model, interprets results, and visualizes key analyses to provide actionable managerial recommendations for a robust newsvendor optimization strategy.

II. Introduction and Problem Context

The publishing company needs to decide how many units of a title to print before demand is realized. Traditionally, managers use:

- Price fixed at **p = 1**
- A basic NV model that considers demand uncertainty but ***ignores price effects***

This project is motivated by several real-world realities:

a. Price affects Demand

Demand clearly changes with pricing - readers are sensitive to price variations, especially for non-essential content. Failing to incorporate elasticity may lead to under or over-production.

b. Operational Costs Are non-linear

- **Rush printing** is significantly more expensive
- **Excess inventory** gives rise to disposal costs
- Missed sales due to under-printing affects reputation and market share

c. Data-Driven Decisions

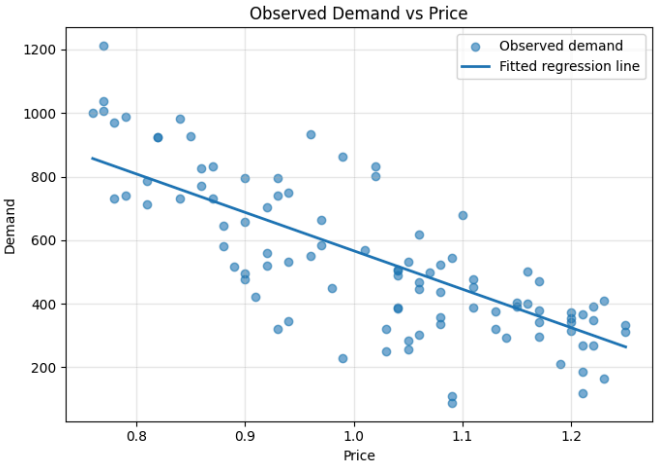
With access to historical data (price + demand), we can build:

- A more accurate statistical view of demand
- A more realistic profit function
- A better optimization model that respects all operational realities

This report builds such a model.

III. Data Description & Regression Model

Having established why the classic Newsvendor framework falls short in our price-sensitive environment, we now turn to the data itself.



Understanding how customers historically responded to price changes is the foundation for any improved decision model. This section begins our analytical journey by uncovering the relationship

between price and demand - setting the stage for everything that follows.

The dataset includes daily records of:

- Selling **price**
- Corresponding **demand**

We fit the regression:

$$[D_i = a + bp_i + \varepsilon_i]$$

Using the following code:

```
X = df[["price"]].to_numpy()
y = df["demand"].to_numpy()

ols = LinearRegression()
ols.fit(X, y)

a = float(ols.intercept_)
b = float(ols.coef_[0])
r2 = float(ols.score(X, y))
residuals = y - (a + b*X[:,0])
```

a. Regression Findings

- **Intercept (a)** ≈ 1851
- **Slope (b)** ≈ –895
- **Interpretation:** For every $1 increase in price, demand decreases by roughly **895 units**, indicating strong price elasticity.
- **R^2 ≈ 0.78**, meaning 78% of demand variation is explained by price alone.

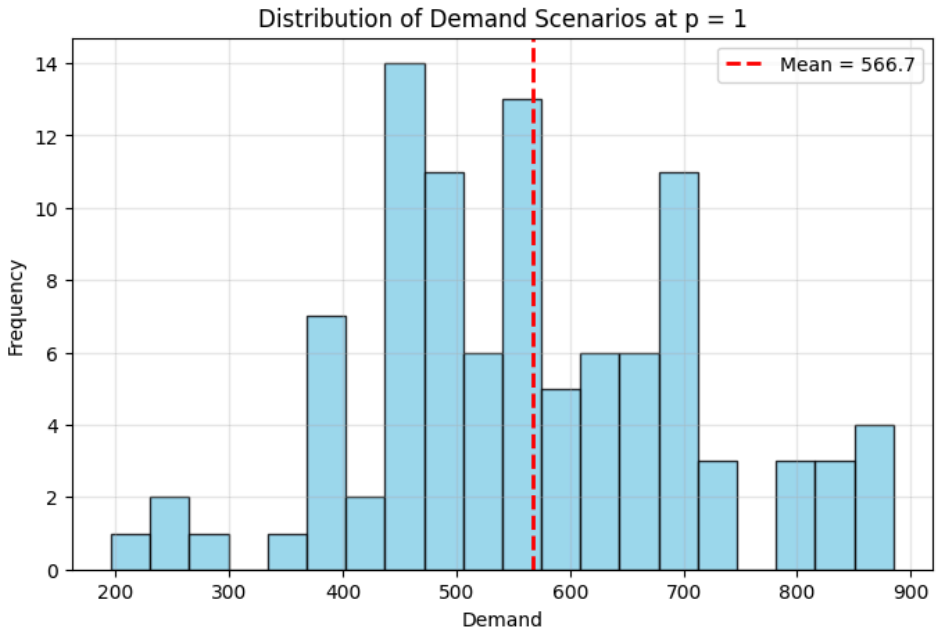
b. Residuals as Demand Scenarios

Residuals represent historical demand shocks and form the basis for our scenario-based optimization in later sections.

IV. Fixed Price Newsvendor with Rush & Disposal (LP)

With a clear view of how demand responds to price, we next explore how the firm currently operates: fixing price at p = 1 and adjusting only print quantities.

Building on the demand scenarios developed in the previous section, we evaluate how well the traditional Newsvendor approach performs once rush and disposal costs are included. This section provides a direct baseline against which our enhanced model will be compared.



At the fixed price (p = 1), demand scenarios become:

$$[D_i(1) = a + b \cdot 1 + \varepsilon_i]$$

Using this:

```
d1 = a + b*p_fixed + residuals
n = d1.size
```

Where:

- (c = 0.5): printing cost
- (g = 0.75): rush cost
- (t = 0.15): disposal cost

a. LP Model

```
m = gp.Model()
q = m.addMVar(1, lb=0.0, name="q")
s = m.addMVar(n, lb=0.0, name="s")
r = m.addMVar(n, lb=0.0, name="r")
d = m.addMVar(n, lb=0.0, name="d")

m.addConstr(s <= q[0])
m.addConstr(s <= d1)
m.addConstr(r >= d1 - q[0])
m.addConstr(d >= q[0] - d1)
```

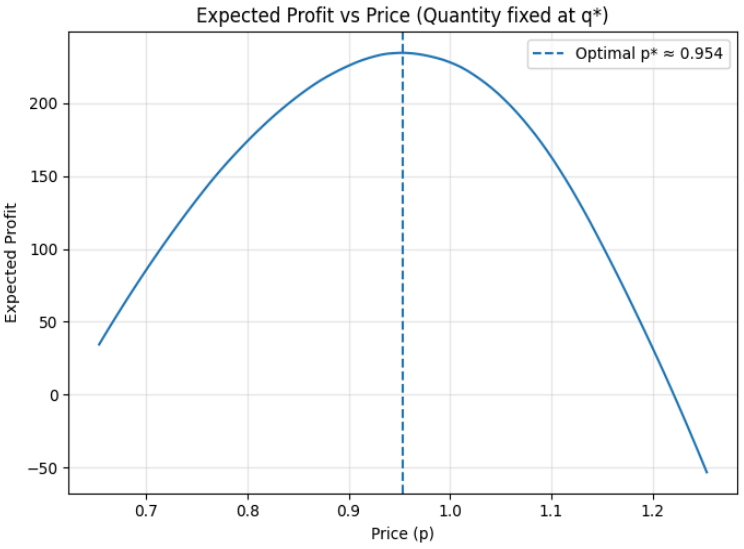
Objective:

```
total_revenue = p_fixed * np.sum(d1)
cost_expr = c * n * q[0] + g * quicksum(r) + t * quicksum(d)
obj = (total_revenue - cost_expr) / n
```

b. LP Results

Metric	Value
Optimal Quantity	**471.865**
Expected Profit	**231.484**

V. Joint Price–Quantity Optimization (QP)



While the fixed-price model reveals important operational tradeoffs, it still ignores the key driver of demand: **price**. This section moves the analysis forward by allowing price itself to be optimized.

Building upon the regression insights from Section 3 and the operational cost structure from Section 4, we now solve a full quadratic program that jointly determines both price and print quantity. This transition marks the moment where our model becomes truly data-driven.

The profit function becomes:

$$[\pi_i(p, q) = pD_i(p) - cq - g(D_i(p) - q)^+ - t(q - D_i(p))^+]$$

where:

$$[D_i(p) = a + bp + \varepsilon_i]$$

a. QP Structure

```
mq = gp.Model()

p = mq.addMVar(1, lb=0.0, name="p")
q = mq.addMVar(1, lb=0.0, name="q")

s = mq.addMVar(n, lb=0.0, name="s")
r = mq.addMVar(n, lb=0.0, name="r")
d = mq.addMVar(n, lb=0.0, name="d")

d2 = a + b * p[0] + residuals

mq.addConstr(s <= q[0])
mq.addConstr(s <= d2)
mq.addConstr(r >= d2 - q[0])
mq.addConstr(d >= q[0] - d2)

revenue_qp = p[0] * gp.quicksum(d2)
cost_qp = c * n * q[0] + g * quicksum(r) + t * quicksum(d)
obj_qp = (revenue_qp - cost_qp) / n
```

b. QP Results

Metric	Value
Optimal Price	**0.9536**
Optimal Quantity	**535.29**
Expected Profit	**234.42**

Interpretation

- Lowering price slightly increases demand significantly (due to steep slope *(b)*).
- Printing more reduces expected rush costs.
- Profit increases relative to the fixed-price model.

VI. Bootstrap Analysis (4000 Iterations)

Once we obtain optimal price and quantity decisions, the natural question becomes: **How reliable are they?** Optimization based on a single sample may overlook uncertainty in the data-generating process. Here, we build directly on the QP result

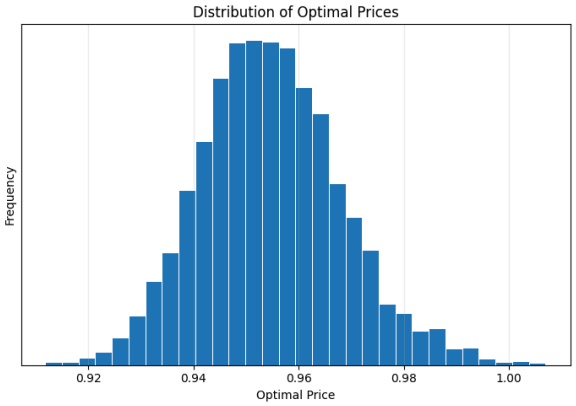
from Section 5 and use bootstrap resampling to evaluate the stability of our decisions under many plausible variations of the historical data. The goal is to quantify risk - not just return.

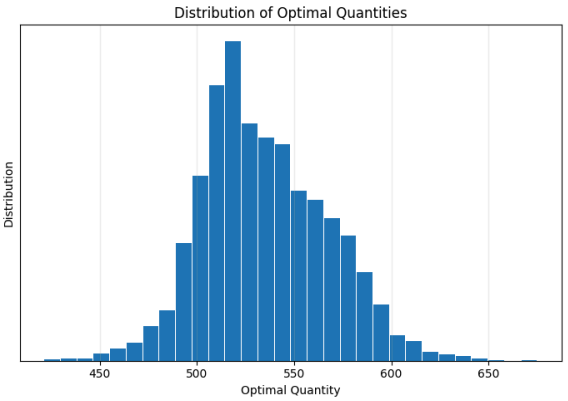
The bootstrap function:

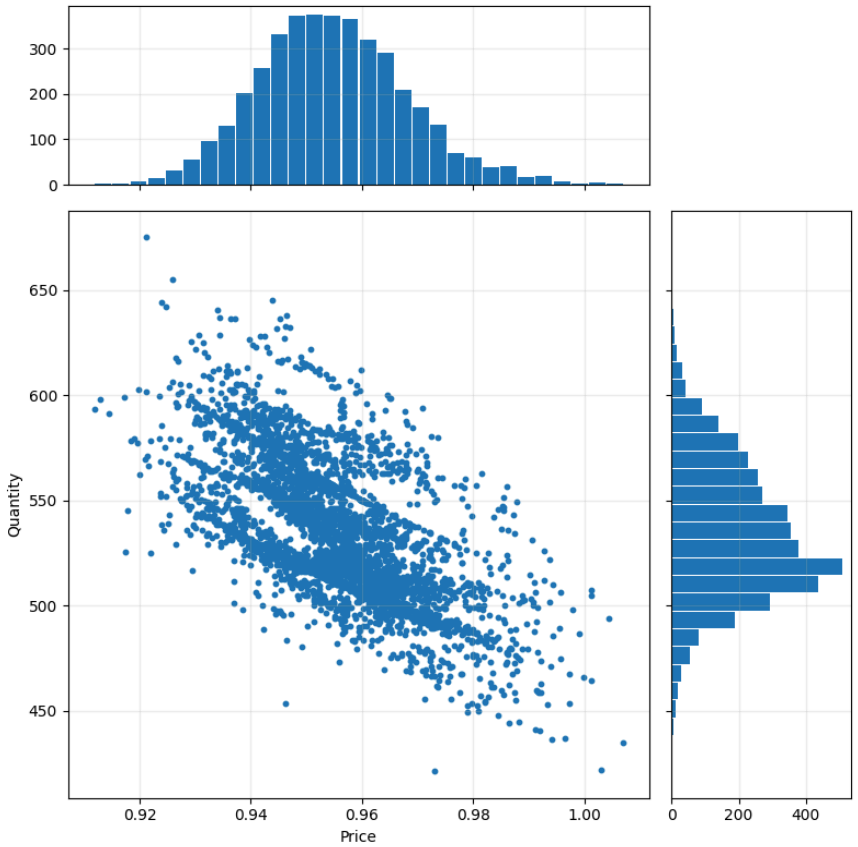
```
def boot(n_iter, df):
    prices = []
    quantities = []
    profits = []

    for i in range(n_iter):
        df_boot = resample(df, replace=True, n_samples=len(df))
        # df_boot.head()

        X = df_boot[["price"]].to_numpy()
        y = df_boot["demand"].to_numpy()

       ols = LinearRegression()
        ols.fit(X, y)

        a = float(ols.intercept_)
        b = float(ols.coef_[0])
        # r2 = float(ols.score(X, y))
        ......

    })

    print("Bootstrap complete!")
    print(f"Mean price:    {np.mean(prices):.4f}")
    print(f"Mean quantity: {np.mean(quantities):.4f}")
    print(f"Mean profit:   {np.mean(profits):.4f}")

    return results

  results = boot(4000, df)
```

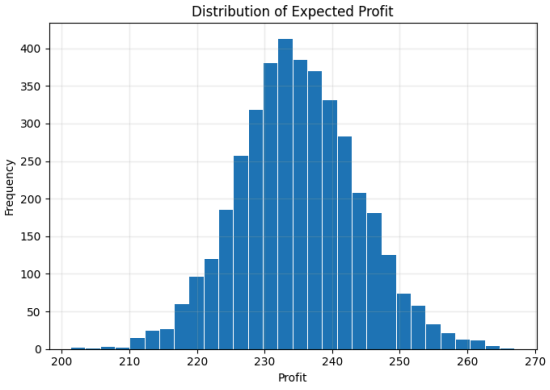
The function:

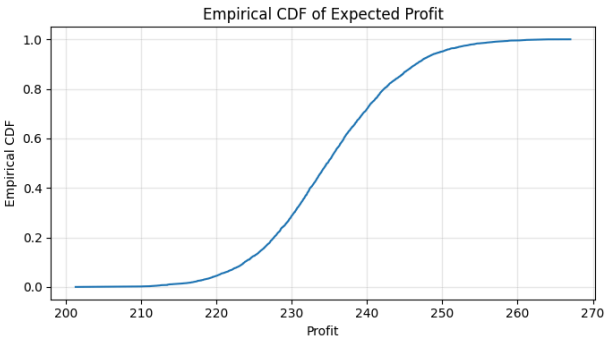
- Resamples the dataset
- Refits regression
- Rebuilds QP
- Re-solves
- Stores optimal p, q, profit


Distribution of Optimal Prices
Frequency
0.92
0.94
0.96
0.98
1.00
Optimal Price


Distribution of Optimal Quantities
Distribution
450
500
550
600
650
Optimal Quantity


300
200
100
0
650
600
550
500
450
Quantity
0.92
0.94
0.96
0.98
1.00
Price
0
200
400


Distribution of Expected Profit
Frequency
400
350
300
250
200
150
100
50
0
200
210
220
230
240
250
260
270
Profit


Empirical CDF of Expected Profit
Empirical CDF
1.0
0.8
0.6
0.4
0.2
0.0
200
210
220
230
240
250
260
270
Profit

a. Bootstrap Summary Statistics

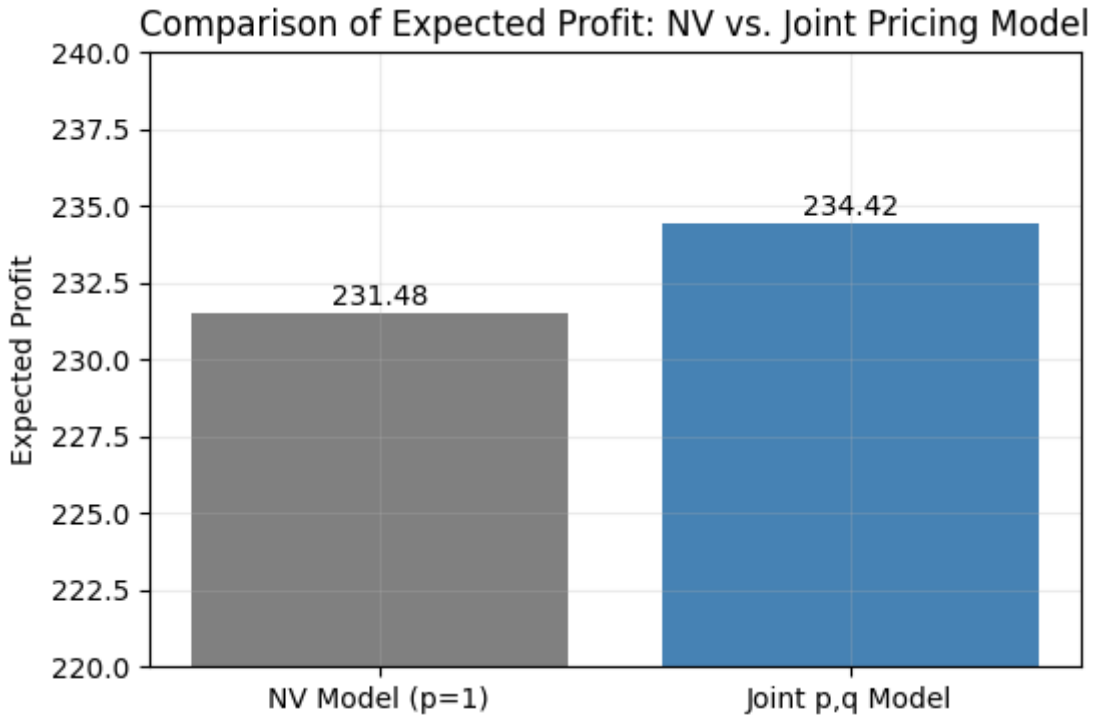
Metric	Mean	Comment
Price	**0.9547**	Very stable
Quantity	**534.49**	Very stable
Profit	**234.66**	Matches QP estimate

b. Risk & Robustness Insight

- The distributions of (p*) and (q*) are **tight**, meaning the model does not fluctuate wildly between bootstrap datasets.
- The profit distribution is similarly concentrated around ≈ 234–236.
- This ensures that the model recommendations will remain **consistent** even if underlying data vary.

VII. Managerial Comparison & Discussion

Armed with both optimized decisions and an understanding of their robustness, we now compare outcomes across the two competing approaches: the classic NV method from Section 4 and the data-driven joint model developed in Section 5. This comparison brings the analysis full circle, connecting statistical insight, optimization, and managerial implications into a unified narrative.



a. Is the NV Model Good Enough?

The NV model works when:

- Price is fixed
- Operational complexity is low
- Data is sparse

But here:

- Price elasticity is **strong**
- Rush and disposal costs matter
- We have enough data for statistical modeling

So the NV model leaves **profit on the table**.

b. Profit Improvement from Switching

$$[\text{QP Profit} - \text{NV Profit} = 234.42 - 231.48 = 2.94]$$

For a single title, this may seem small.

But across **hundreds** of titles and **multiple print cycles per year**, the improvement compounds substantially.

c. Advantages & Disadvantages

Standard NV Model

Pros:

- Simple
- Fast
- Easy to explain
- Works when price fixed

Cons:

- Ignores price elasticity
- Doesn't model rush/disposal inherently
- Lower profitability
- Less data-driven

Extended Price-Sensitive QP Model

Pros:

- Captures real demand behavior

- Models operational constraints
- Higher profits
- Robust to sampling variability
- Better strategic insight

Cons:

- More complex
- Requires regression + optimization
- Needs more computational time
- Requires ongoing data maintenance

VIII. Final Recommendations

Based on the details of both the models, it is suggested to adopt the **price-sensitive QP model** for **high-volume or revenue-critical titles** as a better alternative. It provides:

- Higher expected profit
- Better alignment with true demand behavior
- More realistic treatment of operational costs
- Strong empirical robustness through bootstrap testing

The NV model should be retained only for:

- Low-volume titles
- Cases where price must be externally fixed
- Quick heuristic decisions with incomplete data

Recommended Action:

Adopt the **price-sensitive model** as the default decision-making framework for high-volume titles. Although gains seem modest per title, they scale substantially over thousands of print runs annually.

The price-sensitive model is an overall **superior and future-proof decision tool**.

It reflects real market and operational dynamics, leads to higher profit, and maintains robust performance under uncertainty.

END OF REPORT