Optimization Project III: News Vendor Model

Advaith Shankar

Gayathree Gopi

Jason Antal

Sneha Sastry Rayadurgam

The University of Texas at Austin
Graduate School

Table of Contents

1. Introduction

Objective:

In this project, we explore extensions of the Newsvendor Model (NVM) to address production and pricing decisions in the publishing industry. The standard NVM provides a framework for optimizing production quantities when demand is uncertain to maximize expected profit. However, real-world applications often present additional complexities, such as varying production costs, which this project attempts to incorporate to obtain an optimal solution.

The project is divided into multiple stages, starting with the formulation and solution of the standard NVM using linear programming. Extensions to the model include incorporating rush-order costs, disposal costs, and the impact of pricing on demand using quadratic programming.

Using price-demand data, a linear regression model is fitted to understand the relationship between price and demand, and residuals from this model are utilized to generate new demand scenarios for optimization.

In summary, we begin by finding the optimal production quantity when price is fixed and considering additional costs, then optimizing both price and quantity when demand depends on price, then performing sensitivity analysis using bootstrap samples to test the reliability of our solutions, and finally comparing the extended NVM to the standard model and analyzing how these changes can improve profitability.

Newsvendor Model:

The Newsvendor Model (NVM) is an important approach in operations and supply chain management used to optimize production or inventory decisions under uncertain demand. It is particularly applicable to single-period planning horizons, such as producing daily newspapers or perishable goods. The goal is to determine the optimal production quantity that maximizes expected profit.

In standard NVM, demand is treated as a random variable, and the production decision q must be made before the actual demand DDD is revealed. The objective function is:

$$\max_{q} E\left[p \min(q, D) - qc\right]$$

Where:

p = Selling price per unit

c = Production cost per unit

D = Random demand represented by a probability distribution

min $f_0(q,D)$ = Quantity sold, limited by the minimum of demand and production

This equation reflects that revenue is only generated for the units sold, and excess production incurs costs without generating additional revenue. Conversely, underproduction leads to lost sales opportunities.

In practice, demand data collected over multiple periods is used to estimate expected profit from a demand distribution:

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

This is a non-linear formula that can be simplified into a linear program by introducing dummy variables to handle the min f_0(q,D) term., making it more computationally efficient and solvable using software like Gurobi.

The NVM provides a robust framework for balancing the trade-offs between overproduction and underproduction, making it a useful tool for decision-making under uncertainty.

Newsvendor Model Extension:

A basic extension of this model is the addition of the disposal costs, t and the rush order cost, g. If we don't print enough newspapers to satisfy demand, then we can send a rush order to the printers to print all that we need. These rushed newspapers cost more to print, we will call the cost per rushed newspaper g, where g>c. Additionally, if you print more than the demand you must pay a disposal fee of t dollars per newspaper. You could alternatively think about this as receiving money for taking excess newspapers to a recycling center if t<0. Putting this together, the objective function is:

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

2. Understanding the data

The first step involved loading the provided price-demand dataset. The dataset containing price and demand information was loaded using pandas into a data frame. The dataset had two columns: "price" and "demand", which represent the selling price of the product and its corresponding demand. The data was then reshaped to use with Scikit-learn.

Price vs Demand using Linear Regression:

To capture the relationship between price and demand, a linear regression model was fitted using the Scikit-learn library.

The model predicts demand (D) based on the price (P) with the formula:

$$D = \beta_1 * P + \beta_0$$

Where:

β_1 = slope (change in demand with price)
β_0 = intercept (baseline demand when price is zero)

The code reshaped the price data for compatibility with Scikit-learn, then trained the model. This was the result:

$$D = -1367.71 * P + 1924.72$$

Residuals (the differences between the actual demand and the predicted demand) were calculated to measure model accuracy. These capture the variability not explained by the linear regression.

<u>Calculating the new demand:</u>

To estimate demand for a specific price, the formula $D = \beta_1 * P + \beta_0 +$ Residuals was used. This model incorporates the variability captured in the residuals to simulate realistic demand.

```
X = df['price'].values.reshape(-1, 1)
y = df['demand'].values

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

slope = model.coef_[0]
intercept = model.intercept_
```

```
print(f"Linear Regression Model: demand = {slope:.2f} * price + {intercept:.2f}")
```

```
Linear Regression Model: demand = -1367.71 * price + 1924.72
```

```
df['predicted_demand'] = model.predict(X)
```

```
df['residuals'] = df['demand'] - df['predicted_demand']
```

```
given_price = 1
df['new_demand'] = slope * given_price + intercept + df['residuals']
```

```
df.head(1)
```

	price	demand	predicted_demand	residuals	new_demand
0	1.05	283	488.619393	-205.619393	351.385626

3. Model Development

Model Parameters

In the Newsvendor Model, we incorporated key parameters that reflect real-world considerations for production and inventory management. These include the cost of producing each unit (c), the rush order cost (g), and the disposal cost (t). The rush order cost (g=0.75) represents the additional expense incurred when demand exceeds production, requiring urgent replenishment. The disposal cost (t=0.15) reflects the expense of managing excess inventory when production exceeds demand. These parameters ensure that the model accounts for the penalties of underproduction and overproduction, balancing the trade-offs between them to maximize profitability.

We first solved the model for the plain vanilla scenario, where g=t=0, and calculated the optimal order quantity to be produced when price =1. This became our baseline model.

Extension 1

Decision Variables in the Model for Extension 1

To address the dynamic nature of demand and production, we defined the following decision variables:

Quantity to Produce (q): This decision variable determines the number of newspapers to produce, constrained by a non-negativity requirement, allowing flexibility to scale production as needed.

Daily Profit (h[i]): For each day of sales, h[i] represents the profit, capturing the difference between revenue and costs. This dummy variable helps effectively calculate daily profits without directly influencing decisions.

Auxiliary Variables (Shortage and Excess): These variables handle scenarios where production does not perfectly align with demand. A shortage variable is activated when demand exceeds production (Di >q), incurring rush order costs. Conversely, the excess variable captures overproduction scenarios (q>Di), incurring disposal costs. Both are constrained to be non-negative and mutually exclusive for the same day.

```
# Decision variable: Quantity to produce (q)
q = model.addVar(vtype=gp.GRB.CONTINUOUS, name="q", lb=0)

# Dummy variables for each day representing profit (h_i)
h = model.addVars(n, lb=-gp.GRB.INFINITY, vtype=gp.GRB.CONTINUOUS, name="h")

# Auxiliary variables for positive parts
shortage = model.addVars(n, lb=0, vtype=gp.GRB.CONTINUOUS, name="shortage")  # (D_i - q)+
excess = model.addVars(n, lb=0, vtype=gp.GRB.CONTINUOUS, name="excess")      # (q - D_i)+

# Set objective: Maximize the average profit
model.setObjective(gp.quicksum(h[i] for i in range(n)) / n, gp.GRB.MAXIMIZE)
```

Constraints in the Model for Extension 1

Shortage and Excess Constraints:

When Di>q, there is a shortage, and the excess must be zero.
When q>Di , there is excess inventory, and the shortage must be zero.

Profit Constraints:

Profit (h[i]) cannot exceed the revenue from selling the actual demand minus total costs .

Profit is also constrained by the revenue from selling the entire production quantity minus total costs. These constraints ensure that profit calculations reflect real-world limitations, such as not being able to sell more than what is produced or demanded.

```
# Add constraints for each observation
demands = df['new_demand'].values
for i in range(n):
    # shortage constraint = (D_i - q)+
    model.addConstr(shortage[i] >= demands[i] - q)
    model.addConstr(shortage[i] >= 0)

    # excess constraint = (q - D_i)+
    model.addConstr(excess[i] >= q - demands[i])
    model.addConstr(excess[i] >= 0)

    # Profit constraint
    # h[i] = pD_i - qc - g(D_i - q)+ - t(q - D_i)+
    model.addConstr(h[i] == p * demands[i] - q * c -
                    g * shortage[i] - t * excess[i])
```

Objective Function for Extension 1

Taking all of these into consideration, the objective function in direct mathematical terms can be expressed as

$$\text{Maximize: } \frac{1}{n}\sum_{i=1}^{n} h[i]$$

In simpler terms, we are maximizing the average daily profit, where each day's profit is calculated as:

- Revenue from selling newspapers (price × sold quantity)
- Minus regular production costs (cost × produced quantity)
- Minus rush order costs when demand exceeds production (rush cost × shortage)
- Minus disposal costs when production exceeds demand (disposal cost × excess)

Extension 2

Building on Extension 1, we introduced a dynamic relationship between price and demand, treating demand as a linear function of price with uncertainty modeled through residuals. This adjustment allows the model to capture the interaction between pricing strategies and production decisions, providing a more nuanced approach to maximizing profitability.

Decision Variables in the Model for Extension 2

In Extension 2, we expanded the decision variables to include optimal price (p) and residuals (r), representing the error term from the regression analysis to capture the uncertainty in demand forecasts.

```
# Decision variables
q = model.addVar(name="quantity", lb=0)
p = model.addVar(name="price", lb=0)

# Dummy variables for each day's profit
n = len(residuals)
h = model.addVars(n, lb=-GRB.INFINITY, name="daily_profits")

# auxiliary variables for shortages and excesses
shortage = model.addVars(n, lb=0, vtype=gp.GRB.CONTINUOUS, name="shortage")  # (D_i - q)+
excess = model.addVars(n, lb=0, vtype=gp.GRB.CONTINUOUS, name="excess")      # (q - D_i)+

# objective: maximize average profit
model.setObjective(gp.quicksum(h[i] for i in range(n))/n, GRB.MAXIMIZE)
```

Constraints in the Model for Extension 2

The model includes quadratic constraints to address the nonlinear nature of rush and disposal costs. Additionally, it incorporates:

Price-Demand Relationship: Demand is treated as a linear function of price, expressed as Di =β0 + β1 p + r, where β0 and β1 are regression coefficients, and r represents the residuals.

Profit Constraints: Constraints are updated to reflect the price-dependent demand, ensuring profitability calculations align with the dynamic pricing strategy.

```
# Add constraints for each day's profit
for i in range(n):
    # shortage = (D_i - q)+
    model.addQConstr(shortage[i] >= (beta0 + beta1*p + residuals[i]) - q)
    model.addConstr(shortage[i] >= 0)

    # excess = (q - D_i)+
    model.addQConstr(excess[i] >= q - (beta0 + beta1*p + residuals[i]))
    model.addConstr(excess[i] >= 0)

    # Profit constraint using shortage and excess variables
    model.addQConstr(h[i] == p*(beta0 + beta1*p + residuals[i]) - c*q -
                     g*shortage[i] - t*excess[i])
```

Objective Function for Extension 2

Taking all of these into consideration, the objective function in direct mathematical terms can be expressed as

$$\text{Maximize: } \frac{1}{n}\sum_{i=1}^{n}\left[p(\beta_0+\beta_1 p+\varepsilon_i)-cq-g(\beta_0+\beta_1 p+\varepsilon_i-q)^+-t\left(q-(\beta_0+\beta_1 p+\varepsilon_i)\right)^+\right]$$

This is a Quadratically Constrained problem(QCP) and in simpler terms, we are maximizing the average daily profit, but now with price as a decision variable that affects demand. Each day's profit includes:

- Revenue from selling newspapers (price × demand, where demand depends on the chosen price)
- Minus regular production costs (cost × produced quantity)
- Minus rush order costs when demand exceeds production (rush cost × shortage)
- Minus disposal costs when production exceeds demand (disposal cost × excess)

The key difference is that changing the price will affect the demand through our linear relationship ($D_i = \beta_0 + \beta_1 p + \varepsilon_i$), creating a more complex but realistic optimization problem

Solving this Quadratically Constrained Program (QCP), we identified an optimal price (p=0.9536) and production quantity (q=535), resulting in an expected profit of $234.44. This demonstrated the advantages of incorporating price-demand interactions, yielding more robust strategies than the standard Newsvendor Model.

To further validate the model's reliability, we implemented bootstrapping techniques. By resampling the data, we assessed the robustness of the model's outputs and quantified the uncertainty in predictions, ensuring confidence in the proposed production and pricing strategies. This step highlights the practical use of our approach in uncertain environments.

4. Bootstrapping

Following our initial price-impact model where we jointly solved for optimal price and quantity, we conducted a bootstrap analysis which aimed to explore the variability and robustness of the optimal price and quantity decisions in the context of a price and quantity optimization problem for a manufacturing company. The approach was based on resampling the original data and recalculating the optimal decisions repeatedly to understand how sensitive these decisions are to variations in the data.

Methodology for Bootstrapping:

Data Resampling:

The bootstrap analysis was carried out over 1000 iterations which means we generated 1000 bootstrap samples from the original price demand dataset. Each sample was created using **random sampling with replacement**, which is typical for bootstrapping. This allows data points to be selected multiple times in the same sample. Importantly, each bootstrap sample maintained the same size as our original dataset to preserve the statistical properties of our data.

```
bootstrap_iterations = 1000
optimal_prices = []
optimal_quantities = []
profits = []

for iteration in range(bootstrap_iterations):
    # Bootstrap sample from original price-demand dataset
    bootstrap_sample = df.sample(n=len(df), replace=True)
```

For Each Bootstrap Sample:

For each sample, a new linear regression model was fitted to capture the price-demand relationship. New β_0 (intercept) and β_1 (slope) coefficients were determined for each sample and new residuals were calculated to capture the uncertainty in demand.

The price-dependent demand relationship for each day i was modeled as:

$$D\_i = \beta_0 + \beta_1 p + \varepsilon\_i$$

where:

- D_i represents the demand for day i
- p is the price decision variable
- ε_i is the residual term capturing demand uncertainty

```
# Fit new regression on bootstrap sample
X_bootstrap = bootstrap_sample['price'].values.reshape(-1, 1)
y_bootstrap = bootstrap_sample['demand'].values
reg_bootstrap = LinearRegression()
reg_bootstrap.fit(X_bootstrap, y_bootstrap)

# Get new beta coefficients
beta0_bootstrap = reg_bootstrap.intercept_
beta1_bootstrap = reg_bootstrap.coef_[0]
residuals_bootstrap = y_bootstrap - reg_bootstrap.predict(X_bootstrap)
```

Optimization Process:

For each bootstrap sample, the optimization problem was formulated as a **Quadratically Constrained Program (QCP)**. This was necessary because the demand function includes a quadratic term involving the price (p), making the relationship between price, demand, and profit non-linear. The use of QCP allowed us to handle these quadratic relationships effectively and find the optimal price and quantity that maximizes profit.

The optimization model maximized the average expected profit, where the profit for each day i is given by:

$$\text{profit\_i} = (pD_i - qc - g(D_i - q)^+ - t(q - D_i)^+)$$

where:

- p is the price decision variable
- q is the quantity decision variable
- c = 0.5 is the unit production cost
- g = 0.75 is the rush order cost per unit

- t = 0.15 is the disposal cost per unit
- (x)+ denotes max(x,0)

```
# Create Gurobi model for each bootstrap iteration
model = gp.Model("Optimal_Price_Quantity_Bootstrap")

# Decision variables: Price (p) and Quantity to produce (q)
p = model.addVar(vtype=gp.GRB.CONTINUOUS, name="p", lb=0)
q = model.addVar(vtype=gp.GRB.CONTINUOUS, name="q", lb=0)

# Dummy variables for each day representing profit (h_i)
n = len(residuals_bootstrap)
h = model.addVars(n, lb=-gp.GRB.INFINITY, vtype=gp.GRB.CONTINUOUS, name="h")

# Auxiliary variables for positive parts
shortage = model.addVars(n, lb=0, vtype=gp.GRB.CONTINUOUS, name="shortage")
excess = model.addVars(n, lb=0, vtype=gp.GRB.CONTINUOUS, name="excess")

# Set objective: Maximize the average profit
model.setObjective(gp.quicksum(h[i] for i in range(n)) / n, gp.GRB.MAXIMIZE)
```

For each bootstrap sample, a Gurobi optimization model was created to determine the optimal price and quantity that maximize the expected profit. The model involved:

- Decision variables: price (p) and quantity (q)
- Auxiliary variables for shortages (D_i - q)+ and excesses (q - D_i)+
- Profit calculation incorporating production costs, rush costs, and disposal costs

```
for i in range(n):
    # Demand at any price p will be: beta0 + beta1*p + residuals[i]
    model.addQConstr(h[i] <= p*(beta0_bootstrap + beta1_bootstrap*p + residuals_bootstrap[i]) -
                    c*q - g*shortage[i] - t*excess[i])

    # Define shortage = (D_i - q)+
    model.addQConstr(shortage[i] >= (beta0_bootstrap + beta1_bootstrap*p + residuals_bootstrap[i]) - q)
    model.addConstr(shortage[i] >= 0)

    # Define excess = (q - D_i)+
    model.addQConstr(excess[i] >= q - (beta0_bootstrap + beta1_bootstrap*p + residuals_bootstrap[i]))
    model.addConstr(excess[i] >= 0)


model.Params.OutputFlag = 0
model.optimize()
```

The objective function maximized the average profit across all days in the bootstrap sample, subject to constraints that properly captured the rush order and disposal scenarios. The bootstrap analysis generated distributions of optimal prices, quantities, and profits, allowing us to understand the sensitivity of these decisions to variations in our dataset.

These distributions were visualized through:

- Histograms for each decision variable
- A scatter plot with marginal distributions examining the relationship between optimal price and quantity decisions

These visualizations are provided in the following section and help us identify patterns and relationships between the optimal decisions, providing valuable insights into the stability of our solutions.

5. Results

Linear Regression results:

Linear Regression Model: demand = -1367.71 * price + 1924.72

R^2 value: 0.62147

Observations:

- The coefficient of price is negative, which implies that an increase in the price causes the demand to decrease, more specifically, a unit increase in price causes the demand to fall by approximately 137 units. This aligns with standard economic principles where higher prices typically reduce demand.
- The Goodness of fit: The R^2 value of 0.62147 means that 62% of the variance in the demand can be explained by price. So, it would be worthwhile to take note of the uncertainty caused by other factors beyond the scope (i.e., beyond price).
- It would be of utmost importance to find the right price given the demand or the optimal quantity to order for a given price to breakeven and maximize the expected profits by efficient inventory management.

Residual analysis, demand simulation and Optimal order quantity using Linear Programming (when p=1):

We generated new demand using residuals with the above linear relationship when the price is set to $1. It involved incorporating the residuals into the predicted values to simulate realistic demand scenarios.

We used the generated new demand and solved the linear programming problem on Gurobi to get to an optimal order quantity q, with price, p =1, c=0.5, g=t=0. The results are as follows:

Optimal order quantity to produce = 569.9 (570 newspapers)

Average profit expected in this scenario = $219.28.

We then solved the linear programming problem on Gurobi to get to an optimal order quantity q, with price, p =1, c=0.5, g=0.75, and t=0.1. The results are as follows:

Optimal order quantity to produce = 471.86 (472 newspapers)

Average profit expected in this scenario = $231.48.

This method provides the optimal order quantity and a baseline profit for a given combination of price p, cost c, rush order cost, g and the disposal fee, t.

Linear programming assumes a fixed price and does not take into the combined effects, or the interaction effects between price and demand. This led us to explore other ways to capture this interaction and make informed decisions in those cases.

Price Impact on Demand (QCP results):

We start exploring the combined effects of price and demand by adding price and quantity as two decision variables we wish to optimize. The demand generated for each case was a fixed number in our previous case but by making price as one of the decision variables, the value of demand would depend on the price chosen and the regression coefficients we calculated earlier.

Demand now becomes a function of price:

Demand = -1367.71 * Price + 1924.72 + residual [i]

This demand was then used in the linear programming equations, which in turn made the entire problem a Quadratic programming (QP) problem. Gurobi is able to handle QP well when framed properly. Solving this optimization problem, using the same c, g and t values, to maximize the expected profits, gave us the following results:

Optimal Price = $0.9536

Optimal Order Quantity = 535.306 (approximately 535 newspapers)

Expected Profits in this case = $234.4350

Comparing it with the previous solution, by slightly decreasing our price from $1 to $0.9536, the optimal quantity of newspapers to order increases and at the same time, the overall expected profits also increase from $219 to $234. This indicates that a strategic adjustment of price based on the price-demand interaction can lead to a significant improvement in profitability.

This solution makes sense for a given dataset but there's inherent randomness in terms of which scenarios are likely to occur more and how our profits vary when we look at large samples of data.

But collecting such large amounts of data through a lot of experimentation would be tedious and time consuming. Therefore, we turn to our promising mathematical companion, "Probability" and "Central Limit Theorem" to simulate this exercise.

Bootstrap Analysis:

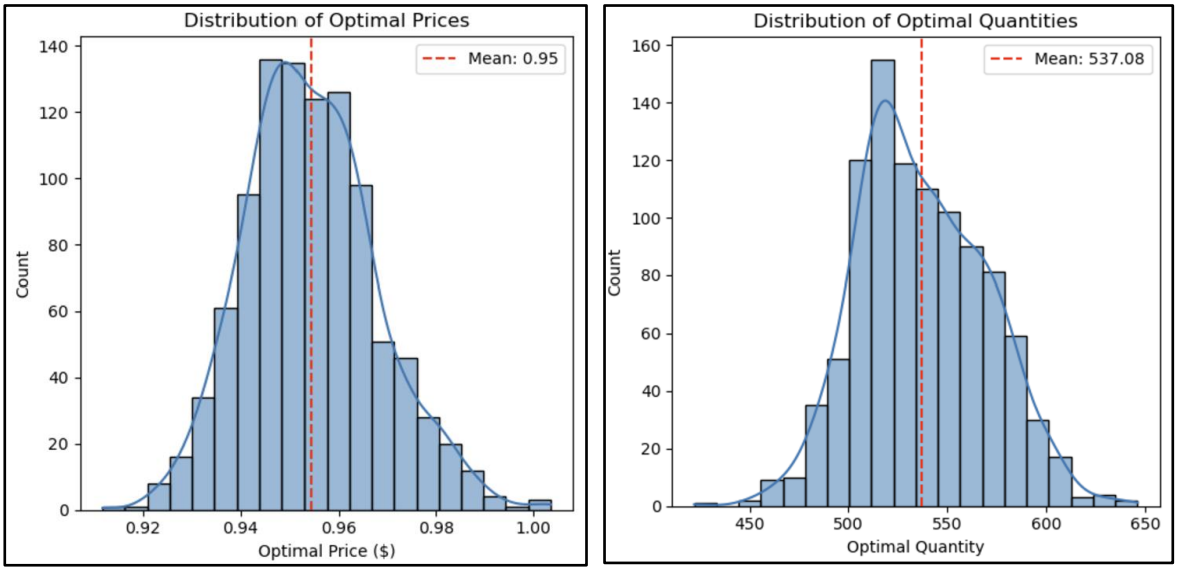
As discussed earlier, it would be very efficient to look at multiple datasets and then run the optimization problem. But the central limit theorem, bootstrapping and probability give a simple and effective answer to this problem. We bootstrapped 1000 samples from the given dataset (with replacement) and solved the QCP for each of them. We then plotted the three results – optimal order quantity, optimal price and expected profits on a histogram to look at the overall distribution for each of these. The overall expected value for each of the quantities is as follows:

The expected optimal price = $0.95

Expected optimal order quantity = 537.08

Expected average profit = $235.37

We observe that these numbers come really close to the values obtained in the previous section (QCP). While these expected values would work better in most of the cases, there are scenarios where optimal values would be slightly different. An interesting point to note is that 99% of the time, the optimal price is still between $0.9 to $1.00, and the optimal order quantity lies between 450 and 650 newspapers. The expected profits would, in these cases, would range from anywhere between $210 and $260. Using this approach, we can adjust the optimal order quantity based on the risk appetite of the vendor and look at the numbers along with a safety stock and a confidence interval as well as the expected profits in those cases.



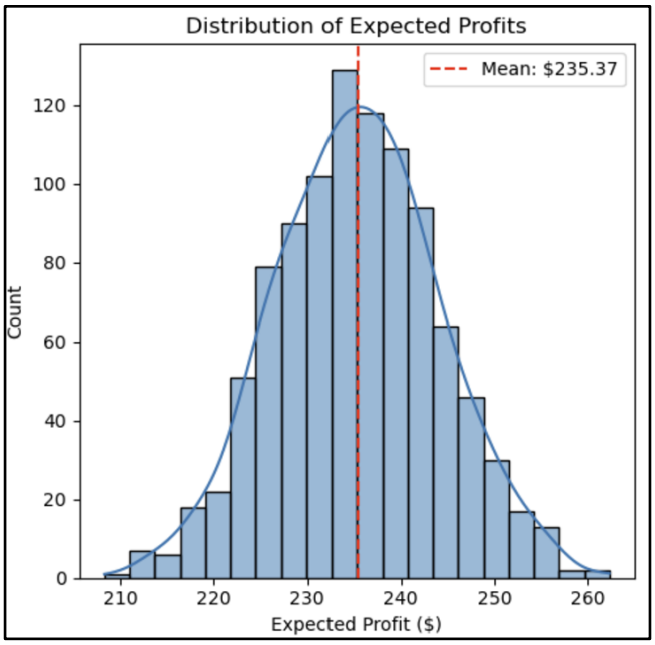


Fig: Distribution of the optimal prices, optimal quantities and the expected profits - They are all normally distributed

We then studied the combined effect of price and quantity and observed that they are negatively correlated- as the price of newspaper increases, the optimal order quantity decreases. This is in line with the earlier observation we made using the regression coefficients.

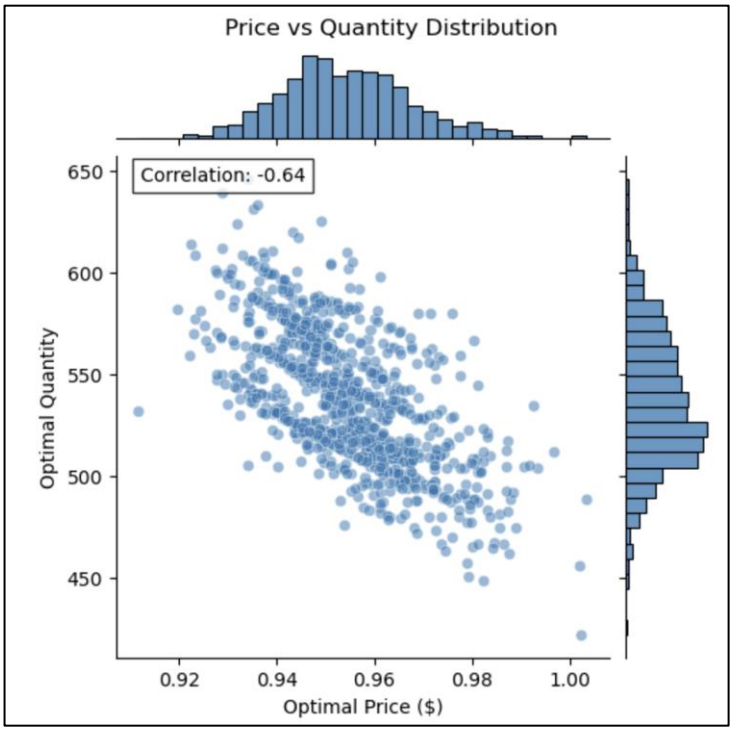


Fig: The histogram for optimal profits and quantity of newspapers to be ordered

Comparative study of models

We now compare the values obtained from different methods:

	Plain Vanilla model (price =1, and without considering g and t)	Model 1 (LP with price =1)	Model 2 (QCP without any bootstrapping)	Model 3 (QCP using 1000 bootstrapped samples)
Price/ Optimal Price	$1.00 (from the question)	$1.00 (from the question)	$0.95364	$0.95
Optimal Order Quantity	570	472	535	537
Expected Profit	$219.28	$231.48	$235	$235
Advantages	• Very Simple	• Simple • Good for short-term decisions where price fluctuations are not frequent	• Captures the price-demand interaction • Shows higher profit potential by strategically changing the price • Is more flexible compared to the simple model	• Assess the robustness of the previous model by adding uncertainty to the QCP solution - helps to assess risk and estimate profit variability • Understand other statistics like confidence intervals and make decisions as per risk appetite
Disadvantages	• Does not capture the disposal and rush order cost • Ignores price-demand interaction	• Fixed price limitation • Ignores the real time price-demand interaction effects	• Slightly more complex • Sensitive to the dataset used as an input to the model	• Computational challenges, in terms of time and complexity • Needs careful interpretation as you enter the realm of statistics

6. Conclusion & Future Scope:

We found that our approach in creating a more advanced model using bootstrapping and rush order/disposal cost variables showed greater ability to set optimal prices. The strengths of our advanced model allows us to forecast prices with greater accuracy and more flexibility, and the greater range of optimal prices allows for more risk management in decision making than a simple newsvendor model. It's not a perfect upgrade; the downside is that our model takes longer to run and loses the simplicity of a straightforward newsvendor model, but we believe the tradeoff is very much worth it.

We also believe that we could enhance this model further to maximize revenues by incorporating market dynamics. The explainability of price in motivating demand is just 62% - which implies that there are other features which should also be looked at while developing such a model. For example, developing a model that accounts for conditional probabilities would enable us to capture how one day's positive sales influence the next day's patterns of demand. A model that effectively optimizes this relationship would be able to not only minimize the negative effects of market swings, but even take advantage of them. Adding capacity, budget, and operational constraints would also reflect the real-world limitations of our business. Additionally, market segmentation could be integrated to better understand and respond to different regions' price sensitivities. These enhancements would take time to implement, but the end product would be an even more comprehensive decision-making tool than our advanced model that would improve results for our company.

An important extension of this project would involve capturing and optimizing the profit across the entire supply chain. In practice, various entities within the supply chain—such as manufacturers, wholesalers, and retailers—often enter into contractual agreements like profit-sharing or revenue-sharing models. These agreements help distribute risk more effectively across different stakeholders, promote collaboration, and ultimately enhance sales. By aligning incentives and leveraging these contracts, the overall profitability of the supply chain can be maximized, benefiting each individual participant as well as the system as a whole.

For now, our current advanced model's ability to capture more realistic outcomes shows great promise for improving demand forecasting operations. We believe it will boost revenues, and our recommendation is that the company switches to this model for all forecasting operations going forward.