I

**PRICE ELASTICITY MODELLING USING PYSPARK**

A diagram of a company's demand

Description automatically generated with medium confidence

**FINAL GROUP 1-3 - IST 718: BIG DATA ANALYTICS**

TEAM MEMBERS: Keerthi Krishna Aiyappan, Himanshu Hegde, Pranav Mahesh Mekal, Sheikh Mohammad Nawazish Khalandar

**Project Overview**

The demand for a product in the market is defined by multiple factors including its price, consumer income, availability of substitutes and complementary goods, consumer preferences, population demographics, and broader economic conditions. Marketing, seasonal trends, technological advancements, and government policies also play crucial roles.

Among these, price stands out as a key driver of demand, leading to the concept of **price elasticity**. Price elasticity measures how sensitive consumer demand is to changes in price, and understanding this sensitivity is vital for businesses to optimize pricing strategies and maximize revenue.

It’s a powerful and simple way to see how certain products are more sensitive to price changing than others. This can help retailers in the pricing process, to **maximize profit**.

One of the benefits of Price Elasticity Modelling (PED) is that it enables firms to understand price sensitivity of their target market better and optimize product prices to make sure that they retain their existing customer base and also bring in new customers.

Another key benefit of PED is firms are not able to accurately quantify changes in revenue for changes in price. The crucial aspect of such a model helps the companies know their target market more. This allows them to bring in new customers. After all, the best way to maximize revenue is to not just retain old customers but also bring in new ones, the right balance of which can be identified from PED.

A cartoon of a person holding a sign and a couple of bottles

Description automatically generatedA cartoon of a person shopping in a grocery store

Description automatically generated

PED measures the percentage change in quantity demanded in response to a 1% change in price. Mathematically, it is expressed as:

Price Elasticity of Demand=A black and white text

Description automatically generated ​

* **Elastic Demand (|E| > 1)**: If the absolute value of elasticity is greater than 1, demand is elastic, meaning consumers are highly responsive to price changes. Small price increases lead to large drops in demand, as elastic products are often replaceable or non-essential. For example, beef is elastic since it can be easily substituted with pork or chicken. Other examples include luxury goods and non-essential items like designer clothing and perfumes.
* **Inelastic Demand (|E| < 1)**: If the elasticity is less than 1, demand is inelastic, meaning that changes in price have a smaller effect on the quantity demanded. If a product has inelastic demand, even significant price changes are likely to have a minimal impact on demand. Some examples of products that behave this way are essential (or need-based) goods, such as medicine or tires.
* **Unitary Elasticity |E| = 1)**: If the elasticity is exactly 1, the percentage change in demand is proportional to the percentage change in price.

**Prediction, Inference and Dataset Description**

What we try to achieve through the project is developing a reproducible solution to maximize revenue gained from selling a product. Higher the quantity sold correlates to higher margins of revenue. To achieve just larger quantities of products sold, we can just sell it at low prices but for any stakeholder, maximizing revenue is the goal. So, at what price should we sell a product to ensure not just he maximizes revenue but also meets the demand of his customers (before it saturates). This leads us to optimizing the price. This is the primary goal of our project.

We are aiming to quantify the diminishing returns of increasing a product’s price and identify the optimal returns from such adjustments. The goal is to help businesses understand how price changes impact consumer demand so they can set prices that maximize revenue without losing customers. By focusing on demand behavior and market dynamics, we aim to support data-driven pricing decisions that drive profitability.

**What is new in our approach and why do we think it will be successful?**

We are leveraging Price Elasticity of Demand (PED), to clearly quantify how sensitive consumers are to price changes. By breaking down demand into elastic, inelastic, and unitary categories, we offer businesses a straightforward method to identify which products are volatile to price adjustments. Also, unlike most traditional PED models that rely on multi-linear regression, we plan to use non-linear methods to capture the more complex relationship between price and demand. Additionally, by incorporating competitor and promotional data, we will account for some of the main confounding variables that impact demand. This advanced approach will provide more accurate insights and help businesses optimize pricing strategies with greater precision.

**Who cares?**

Retailers, service providers, and any business that relies on pricing strategies to drive revenue will care about this approach. If successful, it will enable them to set prices more strategically, improve profitability, and maintain customer satisfaction. Additionally, it helps businesses stay competitive by understanding how price changes influence consumer behavior, leading to smarter, data-driven decisions across different product lines.

**METHODS**

The solution was built out in the following steps:

1. Data pre-processing
2. Exploratory Data Analysis
3. Feature Engineering
4. Modelling
5. Price Optimization

**EXPLORATORY DATA ANALYSIS**

1. **Average Price Distribution by Manufacturer**

**A graph with numbers and lines

Description automatically generated**

The above image shows the average price of shoes sold by each manufacturer. One of the reasons why we chose Manufacturer 320 as our imaginary customer is because of the number of competitors in the same price range, slightly above and slightly below range.

1. **Average Price Trends from 2002 to 2003**

**A graph showing a graph of a graph

Description automatically generated with medium confidence**

The above image depicts the average price of all shoes sold from 2002 to 2003. We can see it average out to somewhere between 37 and 38, especially from July 2002

1. **Cross metric correlation analysis**

**A screenshot of a color chart

Description automatically generated**

Coupon Amount has the highest correlation with quantity among the baseline price metrics. This makes sense as people are more likely buy when there is a discount or a promotion running.

1. **Market Share Fluctuation for Manufacturer 320**

A graph showing a graph of a market share

Description automatically generated with medium confidence

Daily Market Share (12%-14%) for Manufacturer 320 indicates the demand for the products sold by our manufacturer

1. **A graph of a number of different colored lines

   Description automatically generated with medium confidenceComparison between Manufacturer 320 and competitors**

The macro trends remain consistent for our manufacturers (320) and our competitors throughout the time. The micro trends vary slightly due to the possibility of marketing campaigns, celebrity endorsements etc.

**FEATURE ENGINEERING**

The goal of Feature Engineering is to extract as much information about what drives product sales as features, that we will feed into the model.

A diagram of a company

Description automatically generated with medium confidence

We start our feature engineering pipeline with raw data and transform it step by step. First, we build price positioning features to understand how each product's price fits within its category – things like price gaps and relative indices. We then look at competitor metrics to see how our products compare to others in the market.

Next, we add historical features by tracking price changes and calculating moving averages over different time periods (30, 60, and 90 days). To understand our promotion effectiveness, we create features that capture discounting patterns. Finally, we add regional features to account for different market conditions like store density and local income levels.

Each step of our pipeline builds on the previous one, giving us a rich set of features that capture all the important aspects of pricing, market dynamics, and customer behavior.

**Data Pre-Processing and Modelling**

We begin with our preprocessed data, which we further prepare for modeling by handling missing values and outliers using our preprocess\_for\_elasticity function. We then transform our categorical variables (like product categories, store IDs) into a format our models can understand using one-hot encoding. Since we're dealing with time-series data, we use a time-based split rather than random sampling to maintain the temporal nature of our data.

Once our data is ready, we train three different models - Random Forest, Decision Tree, and Gradient Boosting Trees. Each model goes through hyperparameter tuning where we test different combinations of parameters (like tree depth, number of estimators) to find the optimal setup. We use cross-validation during tuning to ensure our models generalize well to unseen data. Finally, we evaluate these models using multiple metrics (RMSE, MAE, MAPE) to understand both their absolute and relative performance, which helps us select the best model for our prediction task.

A diagram of a tree

Description automatically generated

**Modelling results**

All the models we ran gave us similar results, as seen below:

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Random Forest | Gradient Booster | Decision Tree |
| Median Percentage Error (Medape) | 68 | 66 | 67 |
| RMSE | 25 | 25 | 26 |
| MAE | 20 | 20 | 21 |

**Optimization**

Our next goal is to optimize product price, while maximizing revenue. This, however, cannot go unconstrained.

Let us consider two scenarios- unconstrained price increase and unconstrained price increase.

An unconstrained decrease can initially result in improved quantity for target products, but it may result in a decrease in quantity for other products, which may lead to a loss of revenue. It will also cut the profit margin of the target product. Parallelly, unconstrained increase can result in a decrease in quantity for target products and an increase in quantity for other products, which may lead to a decrease in profit margin of the target product.

We could drive away loyal customers with sudden price jumps or start a messy price war with competitors. Sometimes the highest-revenue price point could hurt our brand value or squeeze our profit margins too thin. Hence, constraints need to be set while optimizing for product price.

To implement these constraints in practice, we'll use Python's SciPy library, which offers powerful optimization tools.

Let me explain the two key optimization methods in SciPy that are relevant for price optimization:

**[https://docs.scipy.org/doc/scipy/tutorial/optimize.html]**

L-BFGS-B (Limited-memory BFGS with Bounds):

* Best for simple boundary constraints (like min/max prices)
* Works well when you just need to keep variables within ranges
* More efficient with memory, good for large-scale problems
* Faster convergence for simpler constraints
* Example: When you just need to keep prices between 70% and 130% of original

SLSQP (Sequential Least Squares Programming):

* Handles more complex constraints (like minimum quantity requirements)
* Can deal with both equality and inequality constraints
* Good for when you need multiple types of constraints
* Example: When you need to ensure:
* Prices stay within bounds
* Sales volume doesn't drop below certain level
* Price relationships between products stay logical

**Our implementation:**

The price optimization process follows a systematic flow, starting with creating a grid of potential price multipliers (ranging typically from 0.7 to 1.3 times current prices). This feeds into SciPy's minimize function, which begins an iterative optimization loop. In each iteration, the optimizer tests a price multiplier by creating a new price scenario. For each scenario, we recalculate price-dependent features (like price gaps and indices) and use our trained model to predict new quantities. These predictions help calculate the total revenue, which the optimizer uses to decide whether to try different multipliers or if it has found the optimal solution. Once the optimizer converges, we take that optimal multiplier, create a final price scenario, and calculate the final metrics to understand the impact. Throughout this process, optional constraints like minimum quantity requirements or dynamic price bounds can guide the optimization to ensure business-sensible results.

A diagram of a company

Description automatically generated

**Key Functions:**

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| create\_price\_grid & create\_price\_scenario | • Creates range of price multipliers (0.7 to 1.3) • Applies price changes only to target products • Tests different pricing scenarios |
| recalculate\_price\_features | • Updates price-dependent features after price changes • Recalculates category-level metrics • Handles price gaps and relative indices |
| predict\_quantities | • Makes predictions using trained model • Handles missing values in features • Assembles feature vector for prediction • Returns DataFrame with quantity predictions |
| calculate\_total\_quantity & quantity\_constraint | • Calculates total quantity for target products • Enforces minimum quantity increase constraint • Returns percentage change in quantities • Used in optimization constraints |
| get\_dynamic\_bounds | • Calculates price bounds based on category averages • Sets lower/upper limits for price changes • Includes fallback to default bounds • Ensures valid bound ranges |
| objective\_function & Optimization | • Maximizes revenue through price optimization • Uses scipy.optimize.minimize • Tests different price multipliers • Finds optimal price point • Includes constraints and bounds • Reports optimization results |

**Results:**

We are able to model the updated price, and increase in revenue/quantity for each product.

For example, we see that for product 54163 , decreasing product price from $11 to $8, we see a increase in quantity sold by 1.6% and a revenue change% of 32.1%

A black and white screen with white text

Description automatically generated

We are also able to select products and summarize the impact of our price changing decisions.For example, below we see that for a select range of products,we are able to achieve a 4.9% increase in product quantity sales over a specific time period.  


**Challenges:**

We chose this dataset primarily because of the number of features we could work with to do such a price elasticity model, but it came at the cost of data quality. There were a lot of data discrepancies, and underlying data issues. By this, we do not mean duplicate counts or nulls, but the data was not coherent. For example, there was not a lot of variation when we take the mean of all shoes sold for each manufacturer. The price of the product changed in the dataset daily, but the variation was lesser than we anticipated. This set a hard ceiling on the possibilities of the model.

Lack of optimizers in PySpark meant we had to rely on Scipy optimizer to work in tandem with our PySpark model for optimization. Additionally, the unavailability of model interpretation libraries like Shapley, which can be used to infer feature importances and dependencies from blackbox models was challenging.

We worked with a subset of data because our dataset could not be processed by PySpark and we ran into connection errors and memory problems frequently in the available infrastructure.

From a data point of view, there were consumer demographic metrics, for example, buying potential, credit rating, education status, but they had little variation with price or quantity, making it borderline uninterpretable.