Retail Analytics: Price Optimization Model

Price Optimization Definition

- Price optimization is identifying the optimal price point for any given product at any given location that will yield the highest profit.
- The right pricing can make or break a business, and copying your competitors might mean starting a price war, but making a guess could leave you balking at abysmal sales numbers.
- Hence, successful price optimization is a matter of a balance that can have a major impact on your sales, customer satisfaction, profit, and achievable growth goals.

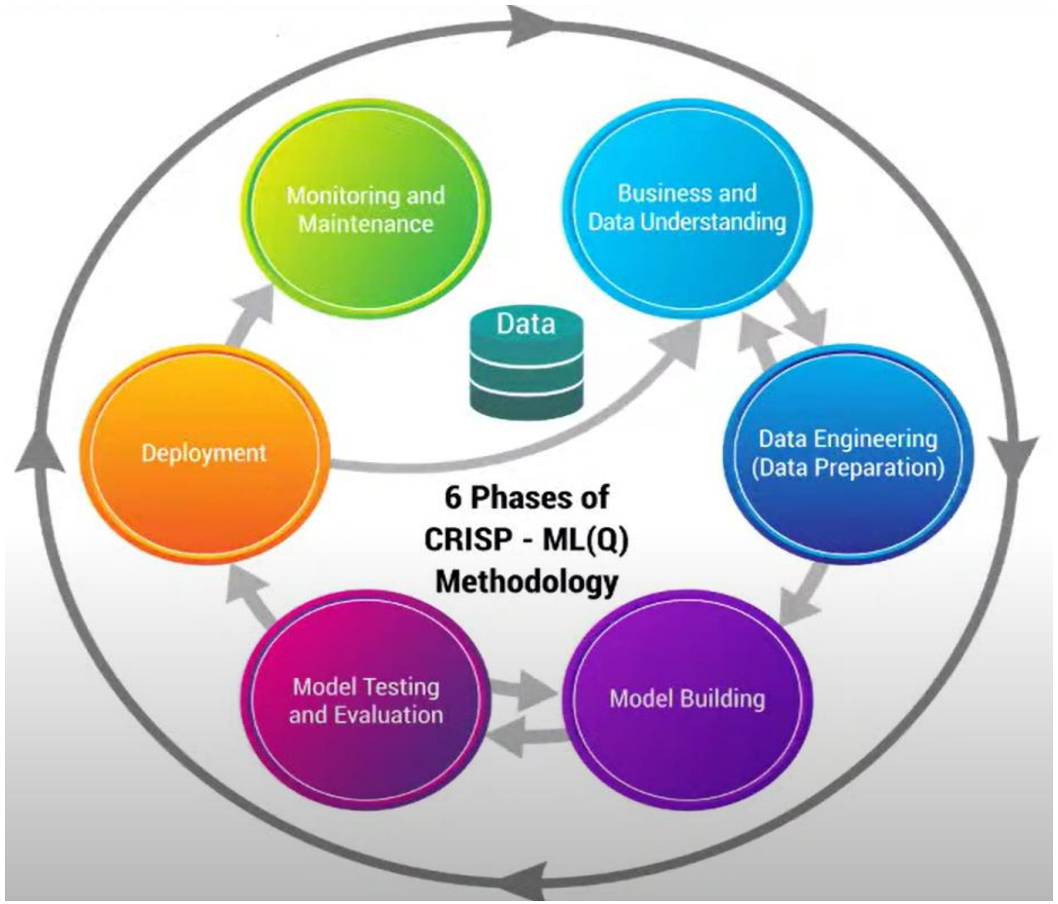
Project Overview and Scope

- This project focuses on predicting an optimal price for a product that will yield maximum profit by increasing the product's sales.
- This prediction system will help the client to get a suggested price based on the selling cost of a product and the number of products sold.
- The project will have a connection between the server where the sales data will be stored, and the model will fetch the data to predict the optimized price. The deployment will be using Streamlit.
- We are building this tool to minimize time and manpower and provide as accurate and thorough results as possible

Project Goals

Objectives	Constraints
• To maximize the profitability and sales, minimize the churning rate of customers to other vendors. • To identify the best price by understanding the pricing policy of each product in the retail market based on price optimization	• A change in market demand for retail products may affect the optimized price. • Price optimization for a product family- Any changes in the pricing of one product, may trigger a chain reaction across a product family/ Hence, the pricing of product family becomes a daunting task.

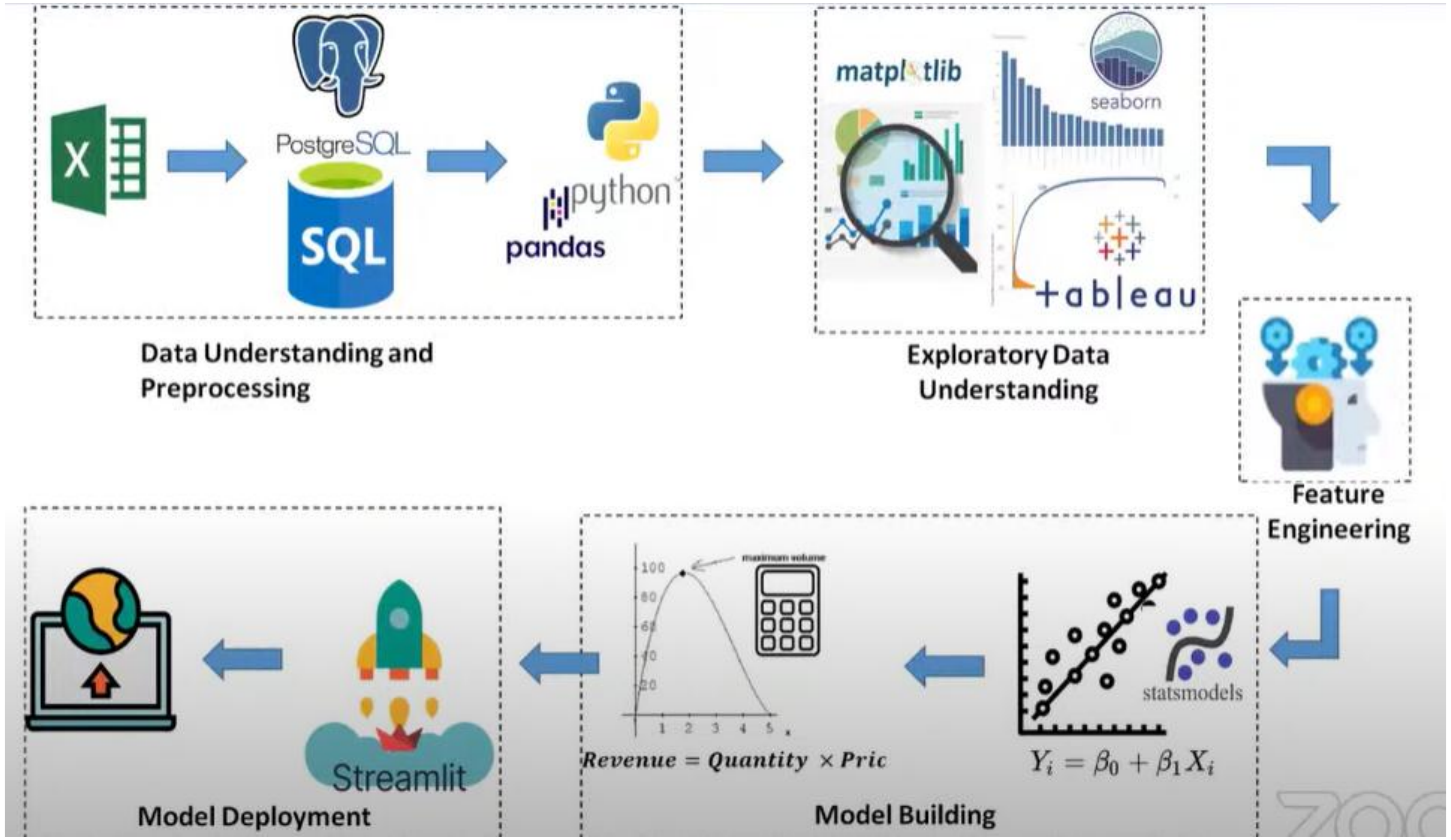
CRISP-ML(Q) Methodology



Technical Stacks

Libraries	NumPy	**Numpy** can be used to perform a wide variety of mathematical operations on arrays.
	pandas	**Pandas** is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.
	matplotlib	**Matplotlib** is a comprehensive library for creating static, animated, and interactive visualizations in Python.
Deployment	Streamlit	**Streamlit** is an open-source python library for creating and sharing web apps for data science & machine learning projects.

Project Architecture



Data Understanding

Particulars	Description
UID	Unique ID for a Material
NAME	Name of the Material
ZONE	Name of the zone of Business
Brand	Brand of the material
MC	Material Category: Category of the material
Fdate	Month of Sale
NSU	Net Sale Unit: Total units sold in a month
NSV	Net Sale Value: Total sale value in a month
GST Value	GST Value: GST on the NSV
(NSV-GST)	Calculation only
Sales at Cost	Cost to company

Data Pre-Processing

Steps in Data Pre-processing:

- **Data Cleaning**

Fill in missing values, smooth noisy data, identify or remove outliers, and resolve inconsistencies.

- **Data Integration**

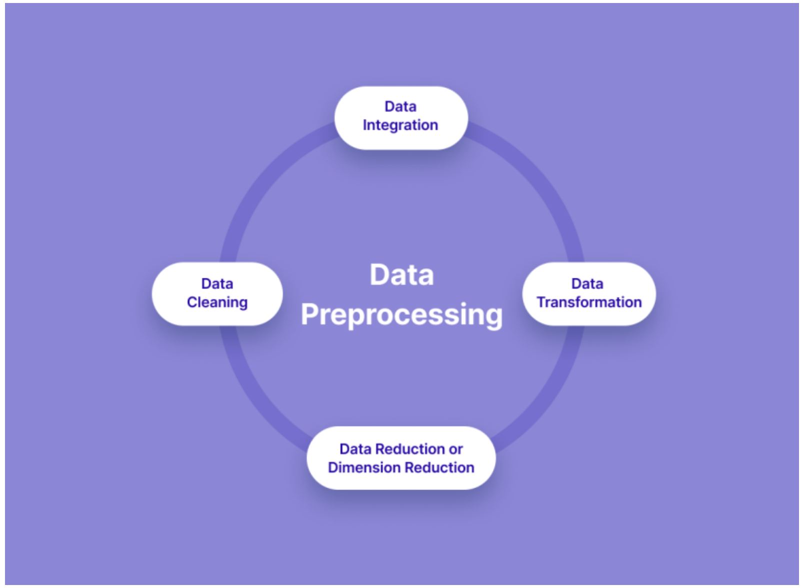
Integration of multiple databases, data cubes, files, or notes.

- **Data Transformation**

Normalization (scaling to a specific range).

- **Data Reduction**

Obtains reduced representation in volume but produces the same or similar analytical results



EDA Description

Measures of central tendency-

- Mean
- Median
- Mode

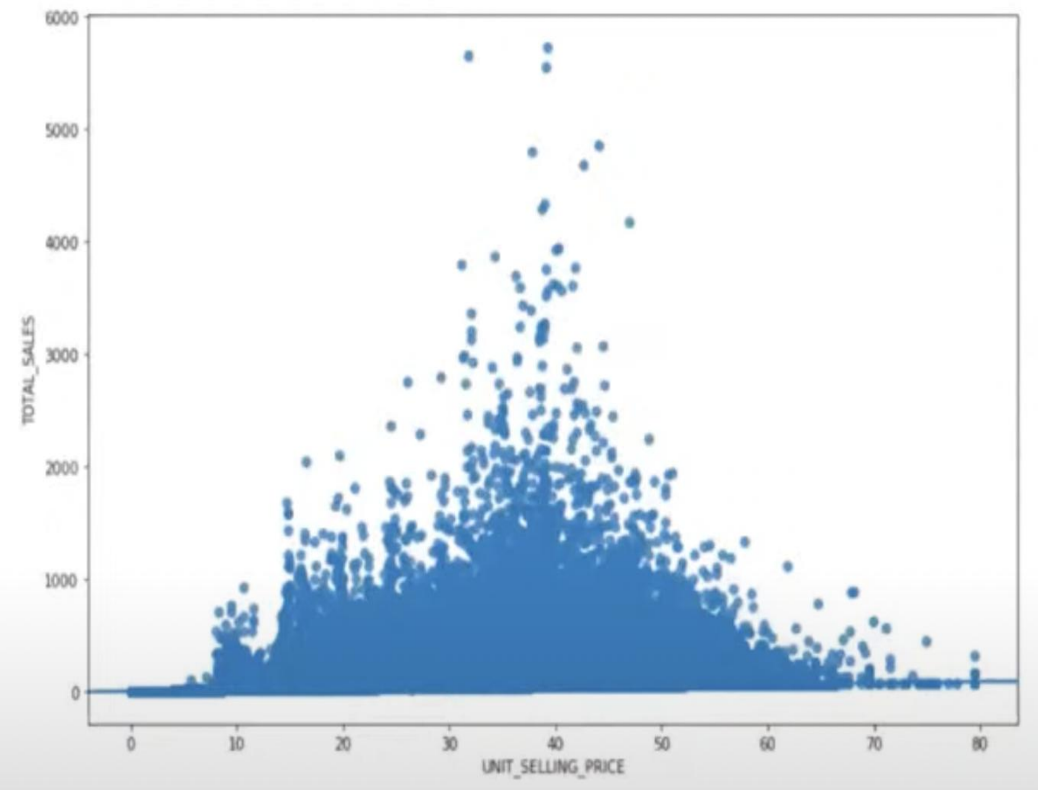
Measures of dispersion-

- Standard Deviation
- Variation

Third and Fourth Business Moment Decisions-

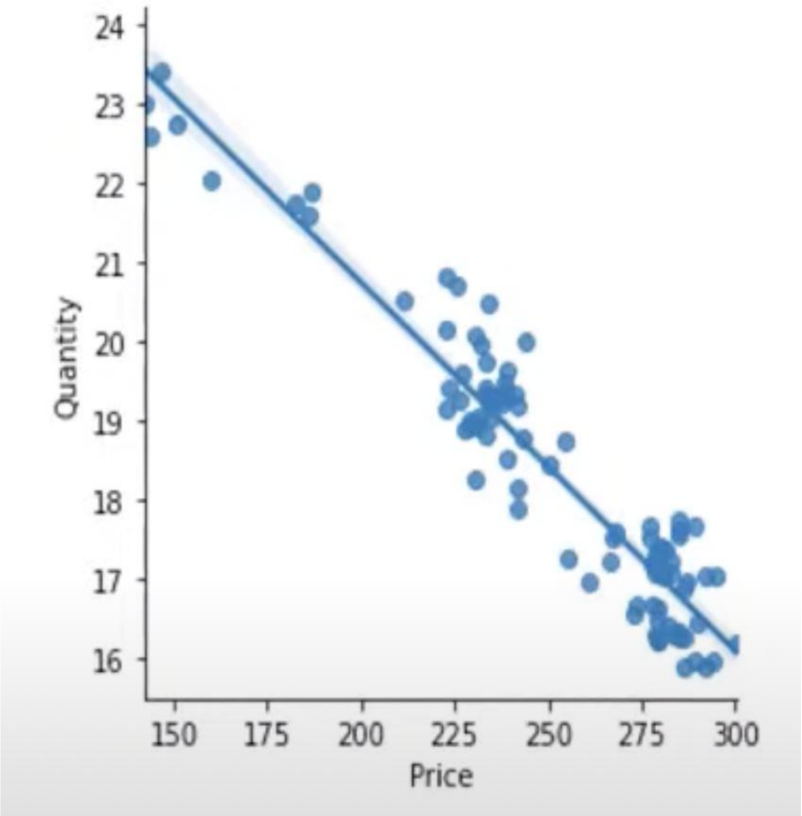
- **Skewness**
- **Kurtosis**

Exploratory Data Analysis



- Sales is the least corresponding to the lowest price of a unit.
- After certain threshold , higher selling price does not result in higher profit.

Model Building



- Optimize selling price of the product to maximize profit and sales.
- Price elasticity ideally needs data on how customers react to price variation.
- A demand curve helps analyze the maximum price at which demand is not impacted.
- We will use existing data to draw demand curve to find range of potential sales price to optimize revenue or profit.
- Based on the range, existing sales data will be used to predict or estimate the selling price.

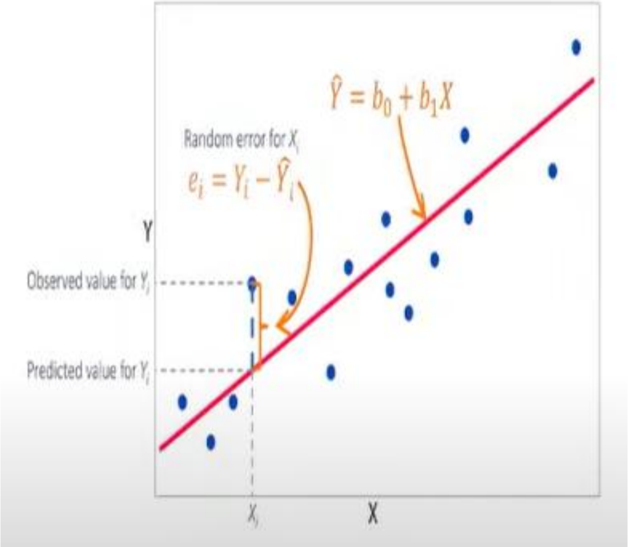
Model Building

Ordinary Least Squares regression (OLS) :

- OLS is a common technique for estimating coefficients of linear regression equations which describe the relationship between one or more independent quantitative variables and a dependent variable (simple or multiple linear regression).
- The simple linear regression is a model with a single regressor (independent variable) x that has a relationship with a response (dependent or target) y that is a
  $y = \beta 0 + \beta 1\ x + \varepsilon$

  Where
  - $\beta 0$: intercept
  - $\beta 1$: slope (unknown constant)
  - ε: random error component

- This is a line where y is the dependent variable we want to predict, x is the independent variable, and $\beta 0$ and $\beta 1$ are the coefficients that we need to estimate.



Model Building

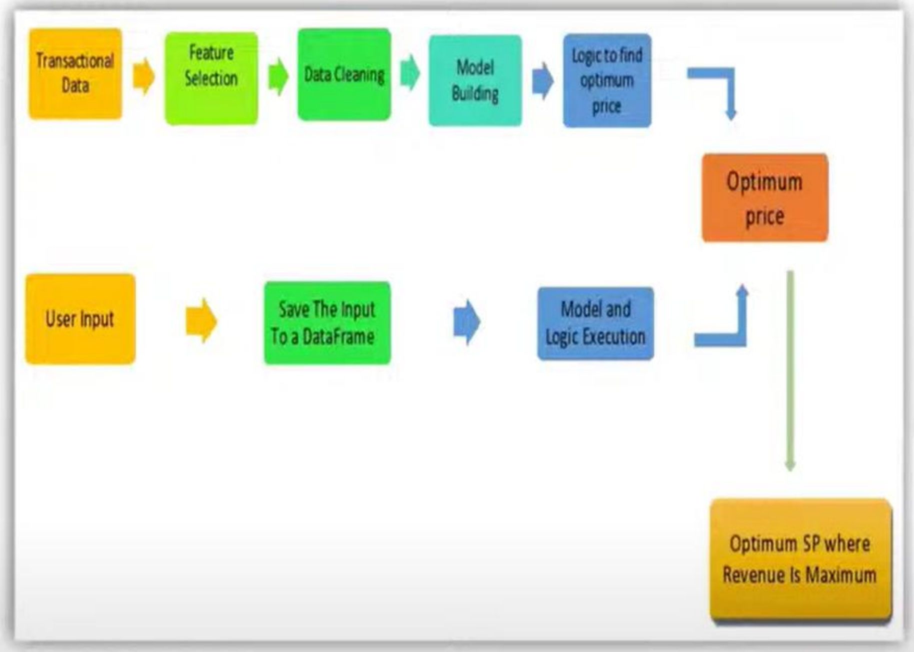


Fig : Model Building Process

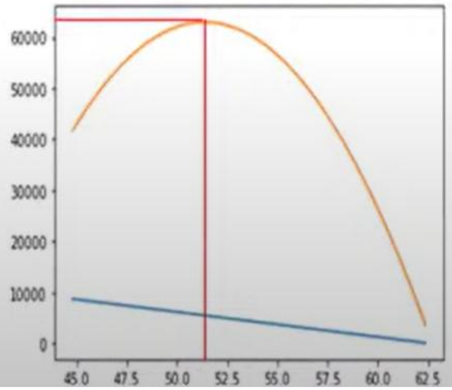
$\text{Quantity} = \beta_0 + (\beta_1 \times \text{Unit Selling Price})$

$\text{Revenue} = (\text{Quantity} \times \text{Unit Selling Price})$

Where,

Quantity is predicted using SLR

Unit selling price – range between minimum value at which company bought a single unit and maximum selling price from the past data.



At maximum revenue, the optimum price can be seen.

Model Building

Price Optimization For Retail To Maximize Profit

```
In [76]: import psycopg2
         conn=psycopg2.connect(dbname='Price_optimization',user='postgres',password='Bu6LYvqQw@123',host='127.0.0.1',port='5432')
         cur=conn.cursor()
         curs = conn.cursor()
         curs.execute("ROLLBACK")
         conn.commit()
         cur.execute('SELECT * FROM "project_price_optimization"')
```

```
In [77]: #cur.execute('SELECT * FROM dataoptimize ORDER BY zone, name, brand, mc')
         df = cur.fetchall()
```

```
In [78]: import pandas as pd
         import numpy as np
         import pickle
         import seaborn as sns
         import statsmodels.api as sm
         from statsmodels.formula.api import ols
         from scipy import stats
         import pylab
```

```
In [79]: df1 = pd.DataFrame(df)
```

Model Building

```
In [80]: df1=df1.rename( {0 : 'UID'},axis=1)
         df1=df1.rename({ 1 :  'NAME'},axis=1)
         df1=df1.rename({2 :'ZONE'},axis=1)
         df1=df1.rename( {3:'Brand'},axis=1)
         df1=df1.rename({4 :'MC'},axis=1)
         df1=df1.rename( {5:'Fdate'},axis=1)
         df1=df1.rename({6:'quantity'},axis=1)
         df1=df1.rename({7:'NSV'},axis=1)
         df1=df1.rename({8:'GST_Value'},axis=1)
         df1=df1.rename({9:'NSV-GST'},axis=1)
         df1=df1.rename({10:'sales_at _cost'},axis=1)
         df1=df1.rename({11:'SALES_AT_COST'},axis=1)
         df1=df1.rename({12:'MARGIN%'},axis=1)
         df1=df1.rename({13:'Gross_Sales'},axis=1)
         df1=df1.rename({14:'GrossRGM(P-L)'},axis=1)
         df1=df1.rename({15:'Gross_ Margin%(Q/P*100)'},axis=1)
         df1=df1.rename({16:'MRP'},axis=1)
         df1=df1.rename({17:'price'},axis=1)
         df1=df1.rename({18:'DIS'},axis=1)
         df1=df1.rename({19:'DIS%'},axis=1)
         df1[['quantity','NSV', 'GST_Value', 'NSV-GST', 'sales_at _cost', 'SALES_AT_COST', 'MARGIN%', 'Gross_Sales', 'GrossRGM(P-L)', 'Gro

         df1.columns

Out[80]: Index([                'UID',                     'NAME',
                               'ZONE',                    'Brand',
                                 'MC',                    'Fdate',
                           'quantity',                      'NSV',
                          'GST_Value',                  'NSV-GST',
                     'sales_at _cost',            'SALES_AT_COST',
                            'MARGIN%',              'Gross_Sales',
                      'GrossRGM(P-L)', 'Gross_ Margin%(Q/P*100)',
                                'MRP',                    'price',
                                'DIS',                     'DIS%',
                                   20],
      dtype='object')
```

Model Building

```
In [81]: # checking the Duplicated values present in the datasets
         df1[df1.duplicated()]
         data = df1.drop_duplicates()

In [82]: # Checking The null values present in th datasets
         data.isnull().sum()
         data = data.dropna()
         data.shape

Out[82]: (37421, 21)

In [83]: # Take the Items Whose Profit margin is greater than 0.
         data = data.loc[data['MARGIN%'] > 0,:]
         data.shape

Out[83]: (30808, 21)

In [84]: top_10_items  = data['NAME'].value_counts().head(10)
         print(top_10_items)
```

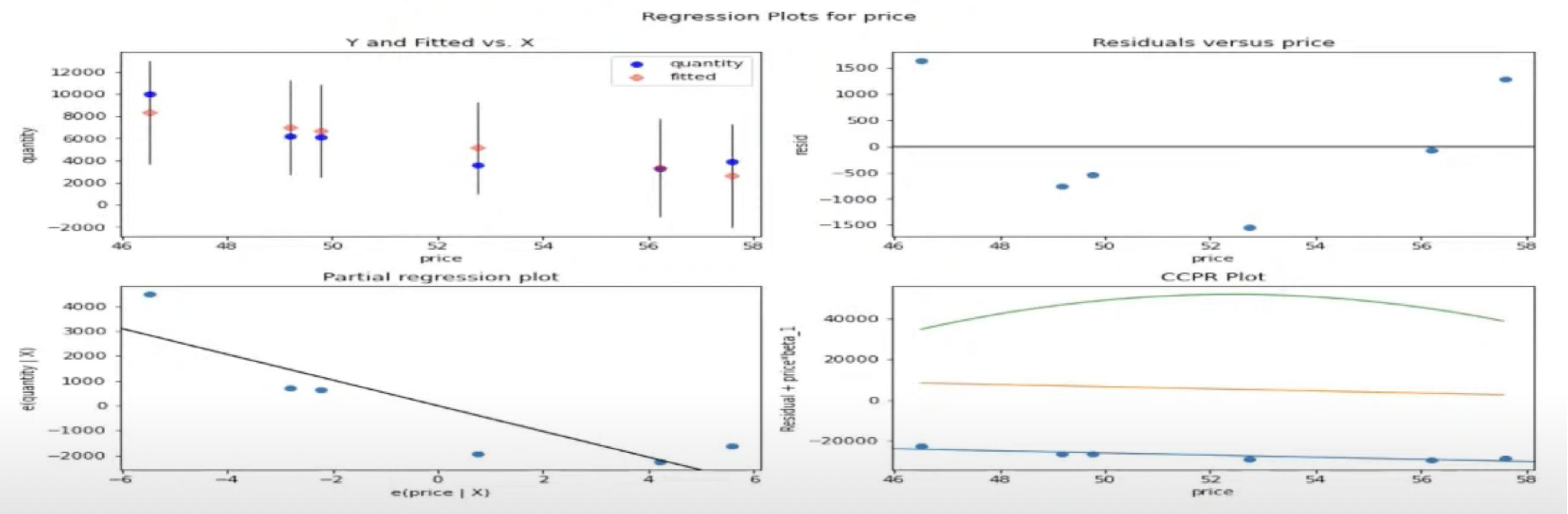
Model Building

In [93]:
```
# For GH TUR DAL PREM 500g
optimal_price = {}
```

In [94]:
```
optimal_price['For GH TUR DAL PREM 500g'] = find_optimal_price(data_new)
optimal_price['For GH TUR DAL PREM 500g']
```

```
Dep. Variable:             quantity   R-squared:                       0.765
Model:                          OLS   Adj. R-squared:                  0.706
Method:               Least Squares   F-statistic:                     13.02
Date:              Tue, 07 Jun 2022   Prob (F-statistic):             0.0226
Time:                      01:42:21   Log-Likelihood:                -50.667
No. Observations:                 6   AIC:                             105.3
Df Residuals:                     4   BIC:                             104.9
Df Model:                         1
Covariance Type:          nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept   3.241e+04   7472.630      4.337      0.012    1.17e+04    5.32e+04
price       -517.0619    143.292     -3.608      0.023    -914.903    -119.221
==============================================================================
Omnibus:                        nan   Durbin-Watson:                   2.409
Prob(Omnibus):                  nan   Jarque-Bera (JB):                0.506
Skew:                         0.257   Prob(JB):                        0.777
Kurtosis:                     1.674   Cond. No.                         693.
```

Model Building



Challenges

- Less data for certain products.
- Dealing with missing values and outlier treatment.
- Research on various prices optimization techniques and choosing the best fit.

Future Scope

- **Customer Behavior Analytics:**

Obtain customer data and categorize customers into regular, semi-regular, and infrequent. Use analytics and machine learning models to understand the purchase behaviors of these users and adjust prices and discounts accordingly.

- **Dynamic Pricing:**

Pricing can further be optimized by regularly adjusting the selling price, which can reflect commodity cost changes, supplier incentives, response to consumer behavior, response to market conditions, etc.