Monte Carlo Analysis on oil prices keeping supply demand balance with Periodic Review

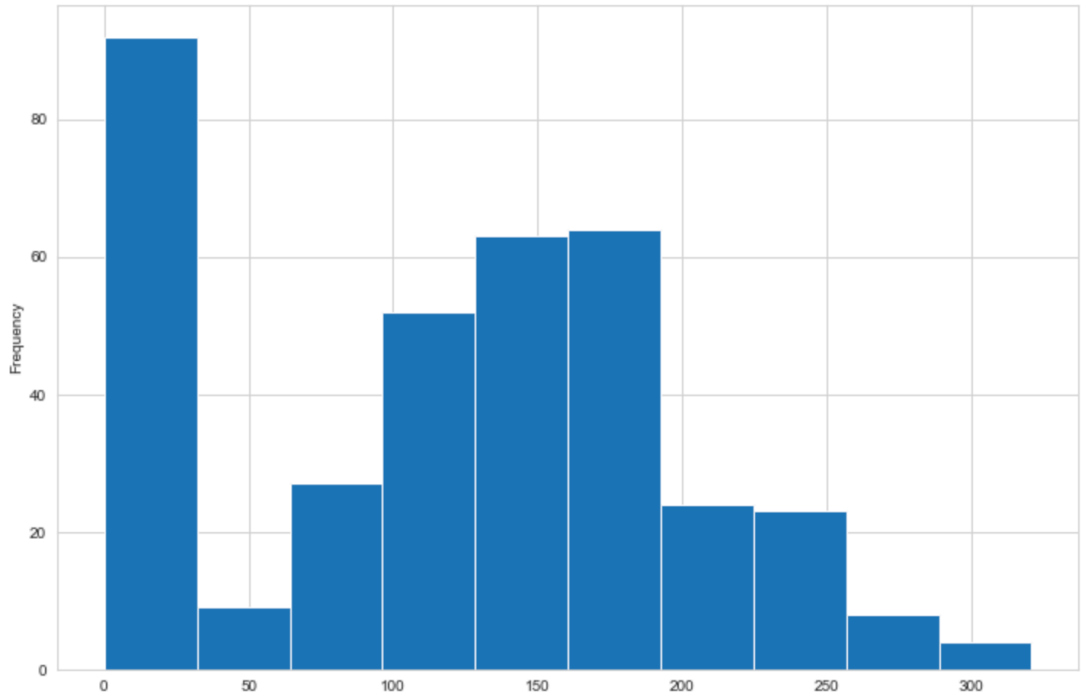
The problem describes the optimization of oil pricing keeping supply and demand balance. This case looks at the sale of a type oil product and looks to adopt a **periodic review** policy to manage its inventory with an objective to maximize profit. The data set consists of a randomly generated demand distribution of the product for one year.

The following information is assumed to be present for each oil product type.

Product	Volume (m^3)	Cost ($)	Selling Price ($)	Initial Inventory (Units)	Lead Time (days)	Demand Lead
Oil	0.52	12	15	3500	5	705

The contents of the table have been described below :-
Product-Type of product
Volume- Available volume of each type of the product.
Cost- Cost of Manufacturing
Selling Price- the price at which each unit of the commodity is sold.
Lead Time- It is the time from the moment the customer places an order to the moment it is ready for delivery.
Demand Lead- The lead demand is the total demand between now and the anticipated time for the delivery after the next one if a reorder is made now to replenish the inventory.

Based on the past year sales, given below is the histogram of the demand distribution of the product.



Periodic Review:-

In this process **it has been assumed** that the stock is replenished after certain fixed time frames which is dependent on the total period, lead time and demand lead of the product. For computation of the optimum profit keeping supply demand balance in check, **Monte Carlo technique** is used to simulate the daily demand of the product. Monte Carlo analysis simulates the behavior of demand and the calculation of profit for all possible scenarios of product demand thereby the result is closer to a global maximum. The description of the algorithm has been defined below.

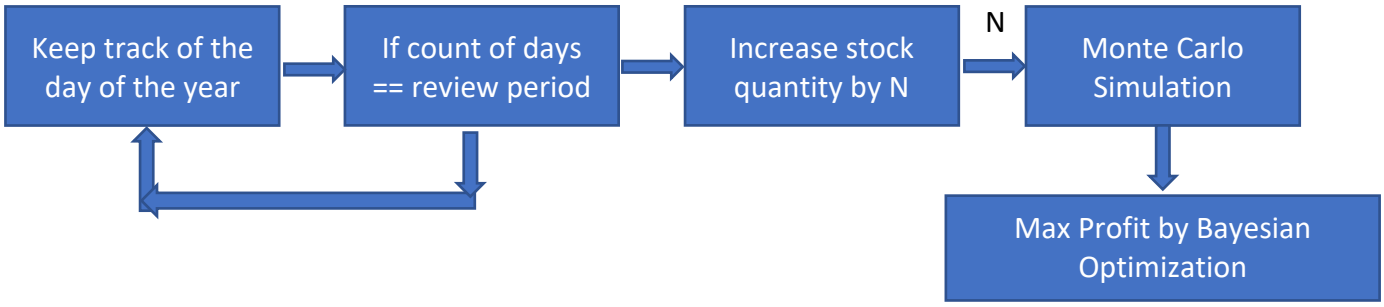
Monte Carlo Simulation Algorithm:-

In the simulation, the algorithm iterates through each day trying to capture the total level for the product thereby generating the product demand for that day.The product demand depends on the current supply level which is updated from the review period in the following two ways.

- If the demand can be completely met by the current supply level, the inventory level is reduced by the demand and number of units sold on that day are increased by the subsequent value.
- If the demand cannot be met completely by the supply level, the supply on hand would be the number of units sold on that day.

The simulation is run for a range of values of N with review period being equal to **30 days**.In the end, **Bayesian Optimization** is used to find the maximum value of the profit from the histogram generated by Monte Carlo Simulation.

The algorithm has been explained below in the following flowchart.



Profit Calculation Model :-

The inventory levels for one simulation of 365 days can be used to determine the profit made for that year. All the units that were sold are multiplied by the selling price to calculate revenue.
The costs are broken down into two components,

- **Product Costs:-**
  Product costs are calculated by multiplying the unit costs of each product to the total number of units ordered.
- **Ordering Costs:-**
  The ordering costs are calculated by multiplying the number of times an order was placed in a year to the individual cost of ordering for that product. The inventory levels for each day of the year were aggregated to indicate how much stock was held throughout the duration of the year.
- **Holding Costs:-**
  The holdings costs were then calculated by multiplying the amount of stock held with the unit volume of the product and the daily cost of holding a unit.

These costs were subtracted from the revenue to give the corresponding profit for that one realization of the year. The mathematical formula for the total profit is given below.

$$Total\ Profit = Revenue - Product\ Cost - Operating\ Cost - Holding\ Cost \quad (1)$$

Results:-

The maximum profit is computed to be **130638.81$.** The following is the maximum profit computed by the GyOpt library after 100 iterations in Python.

Appendix:-

Profit Model:-

```
def calculate_profit(data, product):
    unit_cost = product.unit_cost
    selling_price = product.selling_price
    holding_cost = product.holding_cost
    order_cost = product.ordering_cost
    size = product.size
    days = 365
    ## revenue
    revenue = sum(data['units_sold']) * selling_price
    ## operating cost
    Co = len(data['orders']) * order_cost
    ## holding cost
    Ch = sum(data['inv_level']) * holding_cost * size * (1 / days)
    ## product cost
    cost = sum(data['orders']) * unit_cost
    ## profit
    profit = revenue - cost - Co - Ch
    return profit
```

Code for Monte Carlo Simulation:-

```
def monte_carlo_ray(N, product, review_period=30):
    inventory = product.starting_stock
    mean = product.mean
    sd = product.sd
    lead_time = product.lead_time
    demand_lead = product.demand_lead
    q = 0
    stock_out = 0
    counter = 0
    order_placed = False
    # dictionary to store all the information
    data = {'inv_level': [], 'daily_demand': [], 'units_sold': [], 'units_lost': [], 'orders': []}
    for day in range(1, 365):
        day_demand = daily_demand(mean, sd, probability)
        data['daily_demand'].append(day_demand)
        if day % review_period == 0:
            # Update order
            q = N - inventory + demand_lead
            order_placed = True
            data['orders'].append(q)
        if order_placed:
            counter += 1
        if counter == lead_time:
```

```
            # Restocking day
            inventory += q
            order_placed = False
            counter = 0
        if inventory - day_demand >= 0:
            data['units_sold'].append(day_demand)
            inventory -= day_demand
        elif inventory - day_demand < 0:
            data['units_sold'].append(inventory)
            data['units_lost'].append(day_demand - inventory)
            inventory = 0
            stock_out += 1

        data['inv_level'].append(inventory)

    return data
```