Assignment on DAC

Amber Shukla

Role No: A1-42

Assignment: Real-Life Application of Maximum Sum Subarray Problem

```
price=[100, 110, 103, 93, 91, 88, 85, 79, 72, 79,
 71, 79, 82, 73, 63, 55, 51, 48, 54, 63,
 53, 60, 56, 66, 73, 76, 73, 77, 85, 83,
 73, 68, 71, 71, 69, 63, 59, 59, 52, 44,
 46, 39, 40, 41, 50, 48, 39, 43, 50, 43,
 45, 37, 44, 43, 53, 62, 63, 71, 67, 59,
 50, 47, 46, 38, 35, 28, 30, 28, 32, 42,
 43, 38, 39, 40, 36, 34, 44, 36, 45, 55,
 50, 57, 54, 49, 53, 55, 53, 63, 70, 67,
 67, 58, 55, 46, 46, 48, 46, 38, 34, 42]

len(price)
```

```
100
```

```
change=[]

for i in range(0,len(price)-1):
  change.append(price[i+1]-price[i])

print(change)
```

```
[10, -7, -10, -2, -3, -3, -6, -7, 7, -8, 8, 3, -9, -10, -8, -4, -3, 6, 9, -10, 7, -4, 10, 7, 3, -3, 4, 8, -2, -10, -5, 3, 0, -2, -6,
```

```
# Largest Sum Subarray (Kadane's Algorithm)
largest_sum = change[0]
current_sum = change[0]
optimal_start = 0
optimal_end = 0
current_start = 0

for i in range(1, len(change)):
    if current_sum + change[i] < change[i]:
        current_sum = change[i]
        current_start = i
    else:
        current_sum += change[i]

    if current_sum > largest_sum:
        largest_sum = current_sum
        optimal_start = current_start
        optimal_end = i+1

print(largest_sum, optimal_start, optimal_end)
```

```
42 65 88
```

```
print(f"Best Day to Buy: Day {optimal_start + 1} at Price {price[optimal_start]} , Best Day to Sell: Day {optimal_end + 1} at Price {pric
```

```
Best Day to Buy: Day 66 at Price 28 , Best Day to Sell: Day 89 at Price 70 , Net Profit: ₹42
```

```
import matplotlib.pyplot as plt

plt.plot(range(len(price)),price)
plt.title("Stock Price Over 100 Days")
plt.xlabel("Day")
plt.ylabel("Price (₹)")
plt.grid(True)
plt.scatter(optimal_start, price[optimal_start], color='green', s=100)
plt.scatter(optimal_end, price[optimal_end], color='red', s=100)
plt.show()
```

Report: Using Maximum Sum Subarray for Investment Insights

Objective: Apply Kadane's Algorithm to stock price changes to find the most profitable buy–sell window.

Process

1. Generate Daily Stock Prices.
2. Convert that Data for 100 days of daily gains/losses.
3. Run Kadane's Algorithm to detect the highest profit period.

Results

Buy Day: 67 (Price ₹28)

Sell Day: 88 (Price ₹70)

Profit: ₹42 per share

Pros:

Quickly spots the most profitable historical period.

Can aid in strategy backtesting.

Cons:

Works on past data only—no guarantee for future trends.

Ignores fees, taxes, and market conditions.

Conclusion

This Algorithm is useful historical analysis for maximum profit windows. However, in real investing, it is unable to predict prices meaningfullly.