H9ñññjjj////////

## SIMATS SCHOOL OF ENGINEERING

**SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES**

**CHENNAI-602105**

Maximizing Product Selection for an E-Commerce Platform

## A CAPSTONE PROJECT REPORT

*Submitted in the partial fulfillment for the award of the degree of*

# BACHELOR OF ENGINEERING

## IN COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

**Submitted by**

## Madhavan(192210580) Under the Supervision of

**Dr. T.Sangeetha**

# DECLARATION

I, Madhavan S**,** student of **Bachelor of Engineering in Computer Science Engineering and Artificial Intelligence and Data Science** at Saveetha Institute of Medical and Technical Sciences, Saveetha University, Chennai, hereby declare that the work presented in this Capstone Project Work entitled **"Title"** is the outcome of my own bonafide work. I affirm that it is correct to the best of my knowledge, and this work has been undertaken with due consideration of Engineering Ethics.

Madhavan S

192210580

Date:23-09-2024

Place: Saveetha School of Engineering, Thandalam.

# CERTIFICATE

This is to certify that the project entitled **Maximizing Product Selection for an E-Commerce Platform** submitted by

Madhavan S has been carried out under my supervision. The project has been submitted as per the requirements in the current semester of B.E Computer science engineering and B.Tech Artificial Intelligence in Data science.

Faculty-in-charge Dr. T.Sangeetha

## ABSTRACT

## This project addresses the challenge faced by e-commerce platforms, specifically ShopMax, during sales events. The problem is formulated as a Knapsack Problem, where the platform needs to feature a subset of products on its homepage to maximize customer engagement and revenue while considering space constraints. Given the impracticality of finding an exact solution, a greedy approximation algorithm is proposed, selecting products based on the highest value-to-size ratio. This report outlines the problem, the algorithm design, and the expected results through simulations on ShopMax’s product data.

## Keywords:

## Knapsack Problem

## Approximation Algorithm

## E-commerce

## Greedy Algorithm

## Revenue Maximization

## INTRODUCTION

## ShopMax, an e-commerce platform, faces the challenge of selecting which products to feature on its homepage during sales events. The homepage has limited space, and the company wants to maximize both customer engagement and revenue potential by carefully selecting a subset of products. This problem can be modeled as a variation of the Knapsack Problem, a well-known combinatorial optimization issue in computer science. In this context, products can be viewed as items, each associated with a value (representing potential revenue, engagement, or customer interest) and a size (representing the space required on the homepage). The goal is to select a set of products that maximizes the overall value while respecting the capacity constraint of the homepage.

## Problem Formulation: The Knapsack Problem Analogy

## In the classical Knapsack Problem, given a set of items, each with a weight and a value, the objective is to select items such that the total weight does not exceed the knapsack's capacity, while the total value is maximized. Similarly, ShopMax's problem can be formulated as follows:

## Products: Each product has a size sis\_isi​ (the space it occupies on the homepage) and a value viv\_ivi​ (the expected revenue or customer engagement it generates).

## Capacity: The homepage has a fixed space limit, denoted as CCC, which restricts the total space that can be allocated to selected products.

## Objective: The goal is to select a subset of products that maximizes the total value (engagement or revenue), subject to the constraint that the total space does not exceed the homepage capacity.

## CODING

## def knapsack\_approximation(products, values, sizes, capacity):

## n = len(products)

## # Calculate value to size ratio for each product

## ratios = [(values[i] / sizes[i], products[i], values[i], sizes[i]) for i in range(n)]

## 

## # Sort products by value-to-size ratio in descending order

## ratios.sort(reverse=True, key=lambda x: x[0])

## 

## total\_value = 0

## total\_size = 0

## selected\_products = []

## 

## for ratio, product, value, size in ratios:

## if total\_size + size <= capacity:

## selected\_products.append(product)

## total\_value += value

## total\_size += size

## 

## return selected\_products, total\_value, total\_size

## # Example usage

## products = ['p1', 'p2', 'p3', 'p4']

## values = [100, 300, 200, 150]

## sizes = [10, 20, 15, 25]

## capacity = 50

## selected, value, size = knapsack\_approximation(products, values, sizes, capacity)

## print("Selected Products:", selected)

## print("Total Value:", value)

## print("Total Size:", size)

## OUTPUT

## Example output from the algorithm:

## Selected Products: ['p2', 'p3', 'p1']

## Total Value: 600

## Total Size: 45

## COMPLEXITY ANALYSIS

## Best Case: The best-case scenario occurs when the algorithm can select the highest value-to-size ratio products without exceeding capacity early in the iteration. The complexity is primarily driven by the sorting step, which is \(O(n \log n)\).

## Worst Case: In the worst case, every product must be evaluated and sorted, with no ideal fit, resulting in the same complexity as the best case—\(O(n \log n)\).

## Average Case: On average, the time complexity remains \(O(n \log n)\), dominated by the sorting process. The greedy approach’s approximation ratio is guaranteed to be at least 50% of the optimal solution.

## CONCLUSION

## In conclusion, the greedy approximation algorithm provides an efficient solution to the product selection problem for e-commerce platforms like ShopMax. By selecting products based on their value-to-size ratio, the algorithm ensures near-optimal revenue maximization under space constraints. Although this solution does not guarantee an optimal result, it is computationally efficient and practical for real-world scenarios where exact solutions are infeasible.