Michael_Ghattas_Assignment3

Michael Ghattas

2025-03-08

Portfolio Optimization: Problem Statement, Implementation, and Analysis

Problem Statement

The goal of this assignment is to **maximize the return on investment** given a **fixed budget** by selecting a subset of investment options from a dataset. Each investment option consists of: - **Investment Name** (e.g., state name) - **Investment Cost** (e.g., average home price in that state) - **Estimated Return on Investment** (computed using the 10-year return ratio)

The problem is analogous to the **0/1 Knapsack problem**, where each state is an item with a cost and a profit, and we seek to maximize the profit while staying within the budget.

Dataset Description

The dataset (`state_zhvi_summary_allhomes.csv`) includes: - `RegionName`: The state name (Investment Name) - `Zhvi`: The average home price in that state (Investment Cost) - `10year`: The estimated 10-year return on investment (ROI) as a ratio. - The **Estimated Return on Investment** in dollars is computed as:

$$\text{Estimated ROI} = \text{Zhvi} \times \text{10year}/100$$

The first row contains column headers, and the second row is the **United States aggregate data**, which is **skipped** when reading the file.

Implementation Overview

The implementation consists of two primary functions: 1. **`loadInvestments(investmentFilename)`**: Loads the dataset, extracts relevant columns, and computes the estimated ROI for each investment option. 2. **`optimizeInvestments(investments, budget)`**: Implements a **dynamic programming (DP) solution** to maximize the total return within the budget.

Dynamic Programming Approach

Since this problem is a variation of the **0/1 Knapsack problem**, we use a **2D DP table**: - **Rows** represent **investment options**. - **Columns** represent **available budget values**. - **Base case**: If no investments are selected (row 0), the return is 0. - **Recursive case**: - If we **exclude** investment `i`, the optimal return is the same as without `i`. - If we **include** investment `i` (if budget allows), the optimal return is:

$$\max(\text{previous optimal return}, \text{current return} + \text{optimal return from remaining budget})$$

The **traceback step** identifies the selected investments by checking which choices contributed to the optimal return.

Algorithm Complexity

- **Time Complexity**: $O(nB)$ where n is the number of investments and B is the budget.
- **Space Complexity**: $O(nB)$ due to the DP table.

Results and Example Execution

When executed with a budget of **1,000,000**, the program outputs:

```
Maximum Return: 493
Selected Investments: ['Michigan', 'Tennessee', 'Colorado', 'Nevada']
```

This means that, given the budget constraint, these four states provide the highest possible return.

Key Enhancements

1. **Dynamic Column Detection**: Instead of hardcoding the `'10year'` column, the code dynamically identifies it using:

   ```python
   return_column = [col for col in df.columns if '10' in col.lower() and 'year' in col.lower()]
   ```

   This improves compatibility with different dataset formats.

2. **Efficient DP Table Construction**: The DP table ensures that no redundant computations occur, making the solution efficient even for large datasets.

Conclusion

This implementation successfully applies **dynamic programming** to a **real-world investment problem**. It efficiently selects the best set of investments given a budget constraint, leveraging **knapsack optimization** principles. The results provide valuable decision-making insights for portfolio optimization based on historical real estate returns.