Portfolio Optimization Simulator

Group Members

Muhammad Ali Younas (2024347)

Zuhair Hussain (2024692)

Supervisor

Amir Khan Maarofi

Course
AI Lab
Faculty
BS Artificial Intelligence(AI)

Document Approval

Superviosor:_____

Date:29 November 2025

1. Introduction

The **AI-Driven Portfolio Optimization & Risk Analysis System** is a Python-based project designed to help users analyze financial assets, simulate portfolios, and explore risk–return tradeoffs.
The project combines:

- **Python libraries** (Pandas, NumPy, Matplotlib)
- **Machine learning models**
- **Core data structures** (optional)
- **Financial analysis logic**

This project is designed for an AI Lab course and demonstrates how different disciplines—data analysis, algorithms, and ML—can be used to solve real-world financial problems.

---

2. Project Objectives

The main goals of this project are:

1. **Analyze historical asset price data** using Pandas and NumPy.
2. **Compute portfolio performance metrics** such as returns, volatility, Sharpe ratio, etc.
3. **Visualize financial trends** using Matplotlib.
4. **Use machine learning** to:
    - Predict asset price trends
    - Assist in portfolio optimization
5. **Simulate portfolio performance under multiple conditions**.
6. **(Optional) Integrate data structures** to store and manage financial data more efficiently.

---

3. System Overview

3.1 Input

- CSV files of stock/crypto prices

- User-selected portfolio weights
- Optional user preferences (risk tolerance, budget)

3.2 Output

- Graphs of asset price trends
- Portfolio performance summary
- ML-based predictions
- Recommended portfolio allocation

---

4. Technologies & Libraries Used

Core Python Libraries

- **Pandas** → data cleaning & analysis
- **NumPy** → numerical & matrix operations
- **Matplotlib** → graphs & visualizations

Machine Learning

- scikit-learn models such as:
    - Linear Regression
    - Random Forest Regressor (optional)
    - Train/Test Split
    - Scaling / Normalization

Optional Data Structures

(Used only if the student wants to show DSA knowledge)

- **Graph**
    - Represent relationships/correlations between assets
- **Binary Tree or AVL Tree**
    - Store assets sorted by volatility, returns, etc.
- **Linked List**

    - Store historical price nodes
- **HashMap / Dictionary**
    - Fast lookups of stock data

These are optional and can be included depending on the required project difficulty.

---

5. Core Project Features

5.1 Data Loading & Cleaning (Pandas)

- Import CSV files
- Remove missing values
- Normalize data
- Combine multiple asset datasets

5.2 Financial Metrics Calculation (NumPy)

- Daily returns
- Mean returns
- Covariance matrix
- Portfolio return formula
- Portfolio risk (volatility)

5.3 Portfolio Optimization

- Random weight generation
- Simulation of thousands of portfolios
- Identify best portfolio based on:
    - Highest Sharpe ratio
    - Lowest risk
    - Highest return

5.4 Machine Learning Module

ML is used to predict future price trends.

Models included:

- **Linear Regression** for predicting next-day price
- **Random Forest** for more accurate trend prediction (optional)

Outputs:

- Next-day price prediction
- Trend direction ("UP", "DOWN")
- Confidence score

---

6. Visualizations (Matplotlib)

The system provides the following charts:

- Price vs Time
- Daily return curves
- Predicted price vs actual price
- Portfolio risk-return scatter plot
- Sharpe ratio visualization

---

7. Optional DSA Extensions

These parts are optional and can be added depending on complexity required:

7.1 Graph Data Structure (Optional)

Use case:
Represent correlation between financial assets.

- Nodes → Stocks
- Edges → Correlation values
- Helps identify diversification opportunities

This feature is not required but adds depth.

7.2 Binary Search Tree / AVL Tree (Optional)

Use case:
Sort stocks based on one metric, such as:

- Volatility
- Returns
- Risk

BST/AVL ensures efficient searching and ranking.

7.3 Linked List (Optional)

Use case:
Store daily historical prices in nodes to simulate:

- Easy traversal
- Time-based access
- Simple backtesting

7.4 HashMap (Python Dictionary) (Optional)

Use case:
Store stock name → price data
Fast lookup in constant time.

These optional DSA components allow the project to be flexible and adjustable based on required difficulty.

---

8. System Workflow Diagram

High-Level Steps

1. Load dataset
2. Clean & preprocess

3. Calculate financial metrics
4. Predict future trends (ML)
5. Generate portfolio simulations
6. Visualize results
7. Output recommendations

---

9. Expected Results

The project produces:

- A predicted price graph
- A set of optimized portfolios
- A recommended best portfolio
- Financial insights in numeric and graphical form
- (Optional) DSA-based internal data representations

---

10. Conclusion

This project successfully demonstrates how **Python**, **Machine Learning**, and **Data Structures** can be combined to build a smart financial analysis system.

It is flexible enough to be scaled or simplified:

- If you want it easy → focus on Pandas + ML + graphs
- If you want it advanced → add trees, graphs, and linked lists