STOCK AND PORTFOLIO MANAGER

AI INTEGRATED

Table of Contents

List of Tables

Abstract

This report introduces a machine learning-based investment advisory application tailored to assist users in making data-informed financial decisions. With growing market complexity, individual investors often face challenges in selecting assets that align with their risk appetite and financial goals. This platform addresses that gap by leveraging predictive algorithms and portfolio optimization strategies to recommend investment plans.

The application integrates historical and real-time market data via Upstox API, applies preprocessing techniques, and feeds the refined data into an XGBoost regression model trained to forecast short-term stock returns. It then uses the PyPortfolioOpt library to optimize asset allocation based on user inputs like risk level and investment duration.

The system delivers results through a visually intuitive dashboard supported by charts, tables, and downloadable PDF summaries. Built using Python, Flask, HTML, and Bootstrap, the application ensures accessibility, interactivity, and data-driven outcomes. The project demonstrates how machine learning and financial analytics can simplify investment planning and serve as a prototype for future fintech innovations.

Chapter 1: Introduction

1.1 Background

The digital transformation of financial services has enabled individual investors to engage directly in markets. However, understanding market trends and making calculated investment decisions remains a complex task due to data overload and volatility. There is a significant gap in tools that provide personalized investment advice by combining real-time market insights and predictive analytics.

This project introduces an Investment and Portfolio Manager powered by machine learning, designed to simplify financial planning for retail investors. By automating data analysis, return forecasting, and portfolio allocation, the system aims to reduce the dependency on traditional advisory services.

1.2 Problem Statement

Investors lack accessible tools that combine market prediction, risk profiling, and portfolio optimization in one platform. Most available tools either lack intelligence or are expensive.

Despite the availability of online trading platforms, most retail investors lack access to sophisticated tools that can analyze, predict, and suggest investment strategies tailored to their individual goals and risk tolerance. Existing platforms are either too simplistic—offering basic analytics—or too complex, requiring in-depth financial knowledge. There is a pressing need for a unified system that simplifies investment planning while offering high-level analytical insights.

The core problem addressed in this project is: **How can we design an automated system that predicts future stock performance and allocates investment capital optimally based on user-specific risk and time preferences?**

1.3 Objectives

- Build a platform to analyze historical stock data.
- Use machine learning to predict future returns.
- Suggest optimal investment portfolios based on user risk levels.
- Provide interactive frontend with downloadable reports.
- To collect and preprocess historical stock market data using real-time APIs.
- To apply machine learning algorithms, particularly XGBoost, to predict future stock returns.
- To implement portfolio optimization strategies using the PyPortfolioOpt library.

- To build a responsive web interface that captures user preferences (investment amount, duration, and risk level).
- To generate downloadable reports and visual summaries for user reference.
- To ensure the system is modular, scalable, and adaptable to future enhancements (e.g., deep learning integration, cloud deployment).

1.4 Scope

The scope of this project is focused on individual retail investors interested in Indian stock markets. The application will support the selection of a predefined list of publicly traded equities based on their historical and current performance.

Key features within the scope include:

- Integration with the Upstox API for live market data access.
- Development of a prediction engine using supervised learning techniques.
- Implementation of multiple portfolio optimization models for different user profiles.
- Generation of user-friendly outputs, including pie charts, bar graphs, and PDF summaries.

The project is limited to equity-based assets but has the potential to be expanded to other instruments such as ETFs, mutual funds, or even cryptocurrencies. Advanced forecasting models (e.g., LSTM, sentiment analysis) are identified as future improvements beyond the current implementation.

Chapter 2: Literature Review

Portfolio optimization and stock price prediction have been widely researched within both academia and industry. Various models and strategies have been developed over the years to improve investment decision-making. Linear Regression, ARIMA (AutoRegressive Integrated Moving Average), and LSTM (Long Short-Term Memory) networks are among the most studied models for time series forecasting in financial contexts.

Linear Regression is a traditional method that estimates the relationship between variables. Although simple, it often fails to capture complex patterns in financial time series data. ARIMA improves upon this by considering trends and seasonality, making it useful for univariate time series. However, ARIMA models often struggle with non-linear and multi-factor relationships.

LSTM, a variant of Recurrent Neural Networks (RNNs), is well-suited for sequential data due to its memory capabilities. It is widely used in financial forecasting because it can learn long-term dependencies, though it requires substantial computational resources and data.

In contrast, XGBoost (Extreme Gradient Boosting) is a high-performance, scalable machine learning algorithm based on decision trees. It has gained popularity due to its superior predictive power, speed, and ability to handle missing data. XGBoost performs well on structured/tabular data, making it ideal for our application involving engineered features like lagged returns, moving averages, RSI (Relative Strength Index), and volatility.

For portfolio optimization, several strategies exist. The Sharpe Ratio, introduced by William F. Sharpe, measures the performance of an investment compared to a risk-free asset, adjusted for its risk. The Max Sharpe strategy aims to maximize this ratio, providing a balance of high return and low volatility. Alternatively, Min Volatility focuses solely on minimizing risk, often used by conservative investors.

The Risk Parity approach seeks to allocate risk equally across all portfolio components, rather than capital. It accounts for volatility and correlation, making it robust against market shocks. Constrained Growth applies upper and lower limits on asset weights to ensure diversification and prevent overexposure.

PyPortfolioOpt, a Python library, enables implementation of these strategies using historical return data and risk models. It simplifies the optimization process and allows the addition of custom constraints.

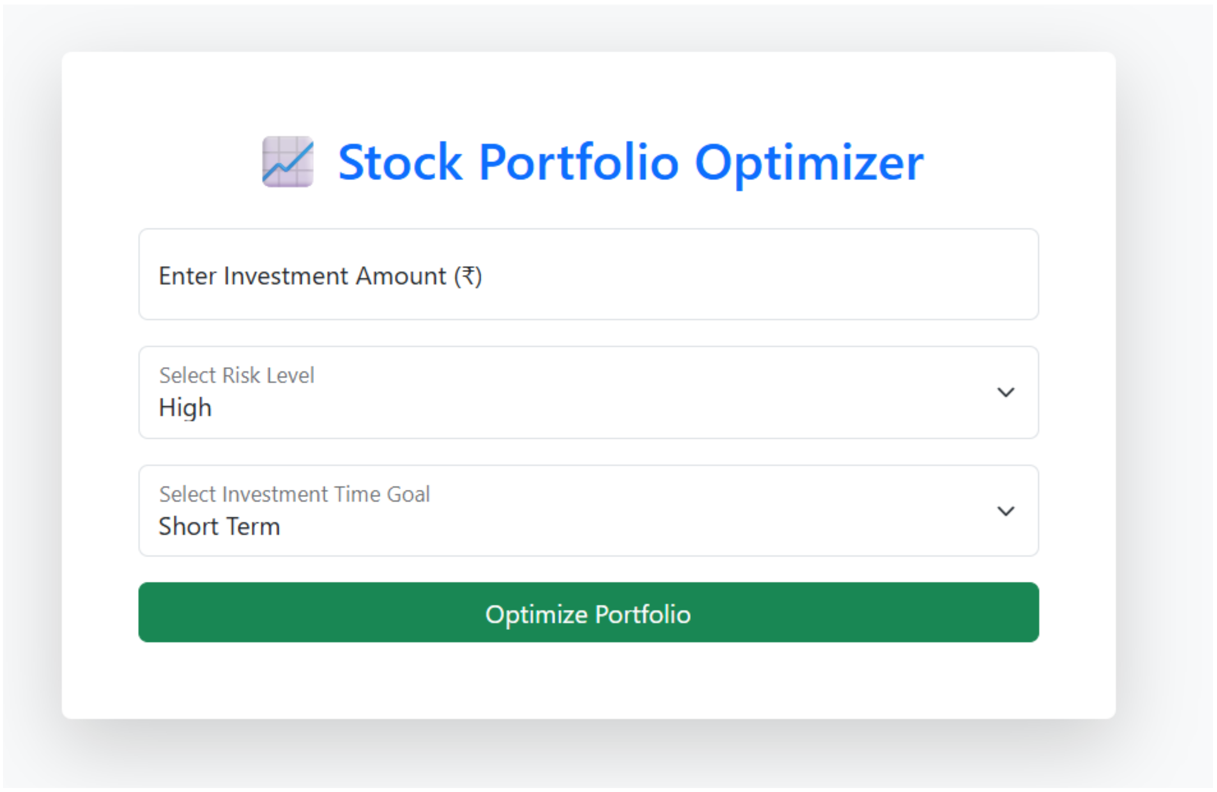
APIs like Upstox enable developers to fetch real-time and historical market data. These APIs are crucial in building systems that require up-to-date financial information, such as our project.

Chapter 3: Methods

3.1 Tools and Technologies

- **HTML**: Used to create the structure of the web pages.
- **Bootstrap 5**: A front-end framework used to design responsive and mobile-first web interfaces.
- **Python**: The core language for backend development and machine learning implementation.
- **Flask**: A micro web framework for Python that handles server-side routing and API integration.
- **fpdf**: A library used to generate downloadable PDF reports from the result data.
- **XGBoost**: A high-performance gradient boosting algorithm used for stock return prediction.
- **Pandas & NumPy**: Libraries used for data manipulation and numerical computation.
- **Upstox API**: Provides real-time and historical stock market data.
- **Matplotlib**: Used for plotting charts and graphs.
- **PyPortfolioOpt**: A financial library used for portfolio optimization using various strategies.

Frontend Interface for Portfolio Optimization


Stock Portfolio Optimizer

Enter Investment Amount (₹)

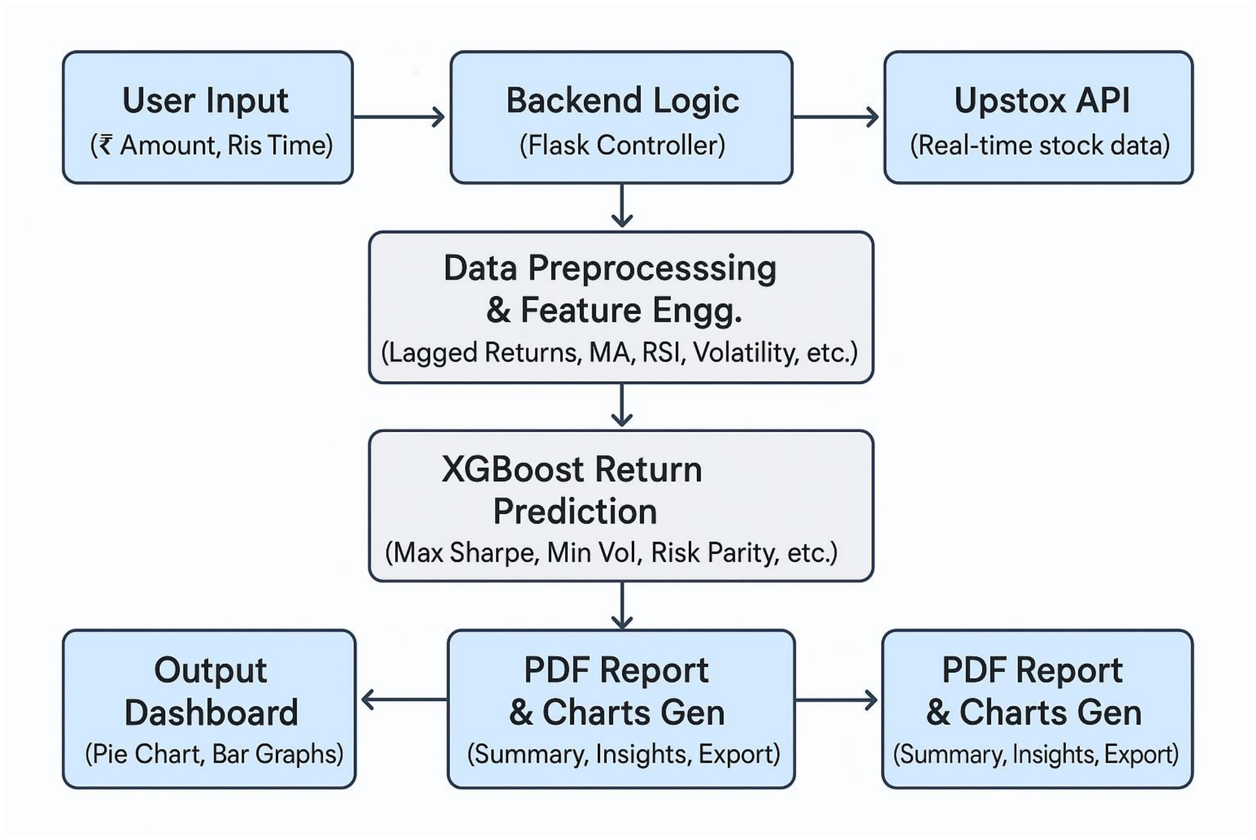
Select Risk Level
High

Select Investment Time Goal
Short Term

3.2 System Architecture

1. **User Input Form**: The frontend takes user inputs such as investment amount and investment duration.
2. **API Call & Data Fetching**: The backend uses Upstox API to fetch stock market data.
3. **Data Preprocessing**: The data is cleaned and features such as moving averages, RSI, and volatility are computed.
4. **Return Prediction**: The XGBoost model predicts the short-term return for each stock.
5. **Portfolio Optimization**: Based on predicted returns, PyPortfolioOpt generates the best asset allocation.
6. **Result Display**: Results are visualized through pie charts and tables.
7. **PDF Generation**: A summary report is created and offered for download.

Project System Architecture



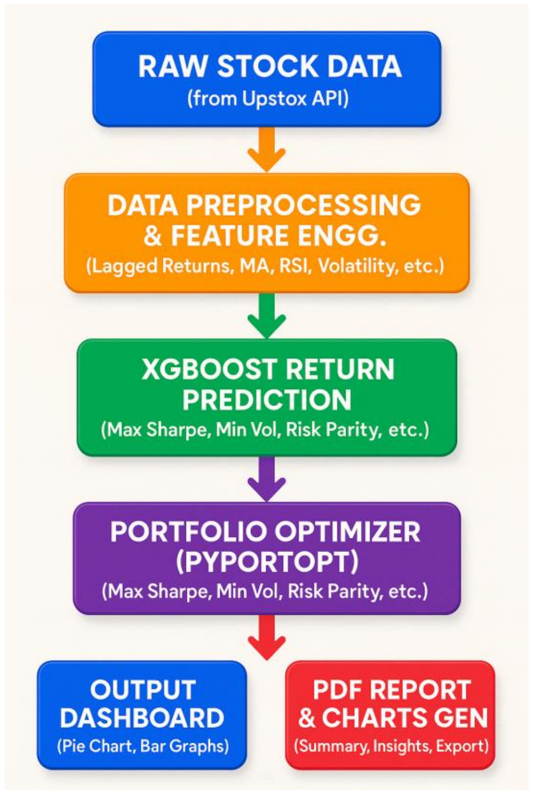
3.3 Prediction Model

The prediction model used in this project is based on the XGBoost algorithm, specifically leveraging the XGBRegressor class from the XGBoost library. XGBoost (Extreme Gradient Boosting) is a highly efficient and scalable implementation of gradient boosting decision trees, widely used in machine learning competitions and production-grade solutions.

To forecast future stock returns, the model requires engineered input features that represent relevant financial patterns. These include:

- **Lagged Returns**: Past returns are calculated for multiple time intervals (e.g., 1-day, 3-day, 5-day, and 10-day) to give the model a sense of short- and medium-term trends.
- **Moving Average Ratio (MA Ratio)**: The ratio of a stock's short-term moving average to its current price indicates whether a stock is trending upward or downward.
- **RSI (Relative Strength Index)**: A technical momentum indicator that measures the magnitude of recent price changes to evaluate overbought or oversold conditions.
- **Volatility**: Computed as the standard deviation of returns over a recent window (e.g., 10 days), which helps the model understand risk levels.
- **Calendar Features**: Day of the week and month are included as categorical features to account for cyclic patterns in stock behavior.

ML Prediction Flow for Return Forecasting


RAW STOCK DATA
(from Upstox API)
DATA PREPROCESSING & FEATURE ENGG.
(Lagged Returns, MA, RSI, Volatility, etc.)
XGBOOST RETURN PREDICTION
(Max Sharpe, Min Vol, Risk Parity, etc.)
PORTFOLIO OPTIMIZER (PYPORTOPT)
(Max Sharpe, Min Vol, Risk Parity, etc.)
OUTPUT DASHBOARD
(Pie Chart, Bar Graphs)
PDF REPORT & CHARTS GEN
(Summary, Insights, Export)

The dataset is split into training (80%) and testing (20%) sets. The model is trained on historical data fetched via the Upstox API and evaluated using Mean Squared Error (MSE) and R^2 score. Cross-validation is applied to avoid overfitting and improve generalization.

The trained model is used to predict future returns for each stock in the selected list. These predictions form the basis for portfolio optimization in the next stage.

3.4 Portfolio Optimization

Once stock return predictions are available, the system applies portfolio optimization techniques to allocate investment capital intelligently. This is accomplished using the PyPortfolioOpt library, which provides robust tools for modern portfolio theory and risk-based optimization.

The process begins by estimating the **expected returns** and **covariance matrix** of the selected assets using historical price data. Several strategies are then applied to determine the most suitable asset weights:

- **Max Sharpe Ratio**: This strategy seeks to maximize the Sharpe Ratio, which represents the expected return per unit of risk. It is well-suited for investors aiming for maximum risk-adjusted gains.
- **Minimum Volatility**: This approach minimizes portfolio risk by selecting low-volatility assets, typically suitable for conservative investors.
- **Risk Parity**: Instead of equal capital allocation, this strategy ensures that each asset contributes equally to the overall portfolio risk. It is highly effective in creating diversified and stable portfolios.
- **Constrained Growth**: This strategy sets upper and lower bounds for each asset to prevent overconcentration and enforce diversification.

Each strategy is implemented by solving a convex optimization problem under real-world constraints such as budget, investment bounds, and asset selection limits.

The final weights are used to allocate the user's investment amount across the selected assets. The system calculates expected returns, risk levels, and generates a detailed allocation report, both on-screen and as a downloadable PDF. This empowers users to make data-driven investment decisions tailored to their financial goals and risk tolerance. Using PyPortfolioOpt: - **Max Sharpe**: Maximizes returns per unit of risk. - **Min Volatility**: Reduces overall risk. - **Risk Parity**: Balances risk across all assets. - **Constrained Growth**: Adds bounds to avoid concentration in few assets. Weights are assigned, and asset values are calculated based on investment amount.

Chapter 4: Analysis & Discussion

4.1 Input-Output Flow

The input-output flow of the Investment and Portfolio Manager application is designed to guide the user through a seamless experience, starting from basic investment preferences to receiving detailed portfolio recommendations with visual insights and downloadable reports.

Step 1: User Input
The system starts by collecting user inputs via a frontend form. Users are asked to specify: - Total investment amount (e.g., ₹50,000) - Investment duration (short-term or long-term) - Risk preference (low, moderate, or high)

These preferences guide the backend in filtering appropriate stocks and tailoring the portfolio strategy.

Step 2: Data Retrieval and Preprocessing
Upon submission, the backend connects to the Upstox API to fetch the latest historical data for a pre-defined list of stocks. The raw data includes daily price movements and volumes. This data is then cleaned and enriched with technical indicators like moving averages, RSI, lagged returns, and volatility.

Step 3: Return Prediction
The cleaned dataset is passed to the trained XGBoost model, which predicts short-term returns for each stock. These predictions are stored and used as a key factor in the portfolio optimization process.

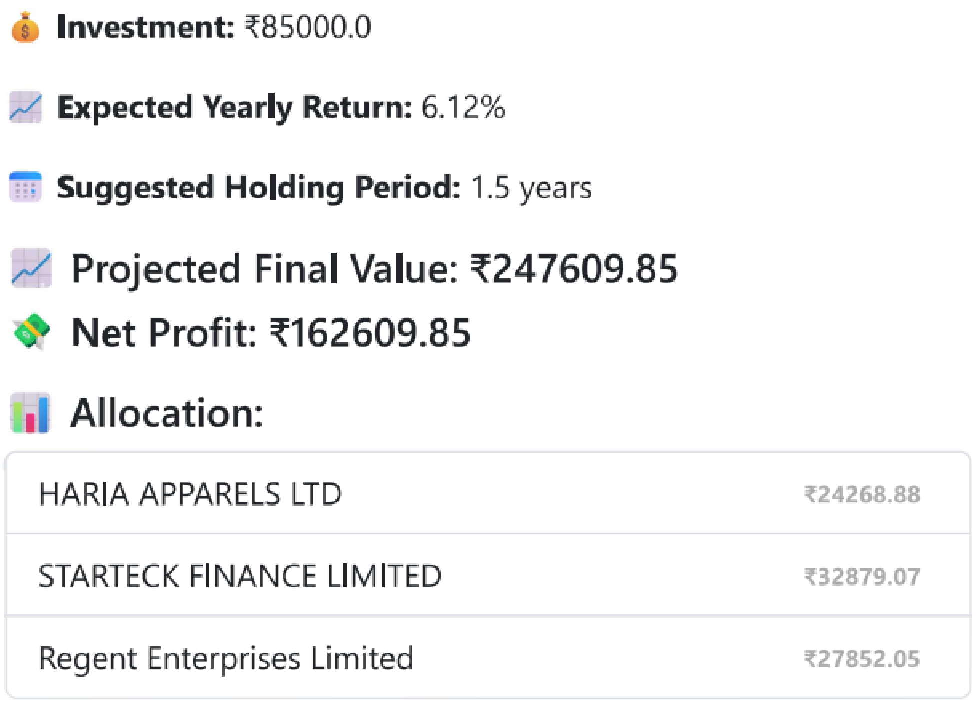
Step 4: Portfolio Optimization
Using PyPortfolioOpt, the system applies selected strategies (e.g., Max Sharpe, Risk Parity) to compute optimal weights for each asset in the portfolio. This considers both the predicted returns and the historical volatility/correlation among assets.

Step 5: Result Generation
The output includes: - List of selected stocks - Allocation percentage for each stock - Expected return and estimated profit - Visual representation using pie charts and bar graphs

Step 6: Report Export
Users have the option to download a personalized PDF report that includes their portfolio summary, strategy used, charts, and final insights.

This structured pipeline allows for modularity, efficiency, and clarity in delivering actionable investment guidance through the application.

4.2 Sample Results



💰 **Investment:** ₹85000.0

📈 **Expected Yearly Return:** 6.12%

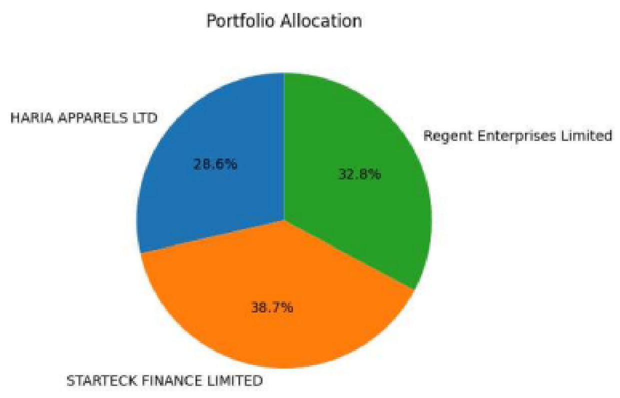
📅 **Suggested Holding Period:** 1.5 years

📈 Projected Final Value: ₹247609.85

💸 Net Profit: ₹162609.85

📊 Allocation:

HARIA APPARELS LTD	₹24268.88
STARTECK FINANCE LIMITED	₹32879.07
Regent Enterprises Limited	₹27852.05



💰 **Investment:** ₹500000.0

📈 **Expected Monthly Return:** 0.9%

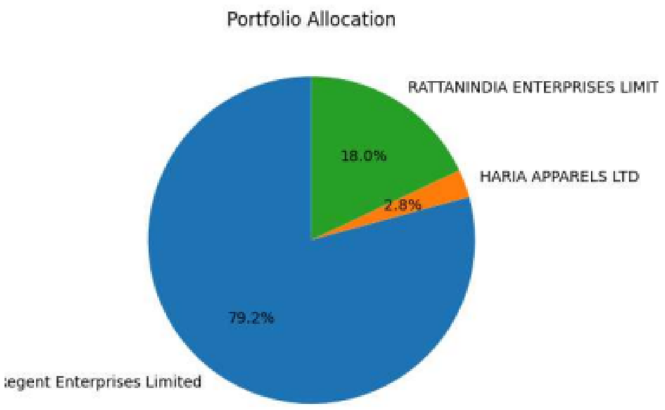
📅 **Suggested Holding Period:** 3 months

📈 Projected Final Value: ₹556721.05

💸 Net Profit: ₹56721.05

📊 Allocation:

Regent Enterprises Limited	**₹396126.68**
HARIA APPARELS LTD	**₹13908.59**
RATTANINDIA ENTERPRISES LIMITE	**₹89964.72**



💰 **Investment:** ₹500000.0

📈 **Expected Monthly Return:** 0.52%

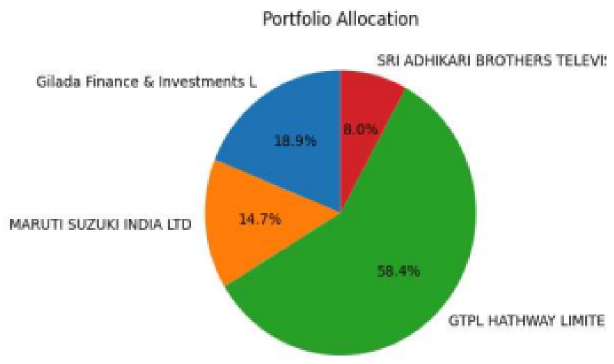
📅 **Suggested Holding Period:** 1 year

📈 Projected Final Value: ₹532248.02

💸 Net Profit: ₹32248.02

📊 Allocation:

Gilada Finance & Investments L	₹94575.77
MARUTI SUZUKI INDIA LTD	₹73704.26
GTPL HATHWAY LIMITED	₹291776.75
SRI ADHIKARI BROTHERS TELEVISI	₹39943.22



💰 **Investment:** ₹10000.0

📈 **Expected Monthly Return:** 1.23%

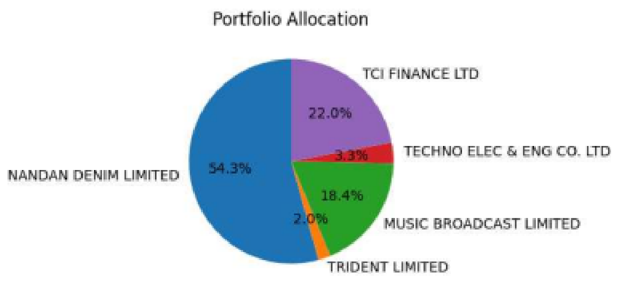
📅 **Suggested Holding Period:** 3 months

📈 Projected Final Value: ₹11582.59

💸 Net Profit: ₹1582.59

📊 Allocation:

NANDAN DENIM LIMITED	₹5429.61
TRIDENT LIMITED	₹197.31
MUSIC BROADCAST LIMITED	₹1838.29
TECHNO ELEC & ENG CO. LTD	₹329.97
TCI FINANCE LTD	₹2204.81



💰 **Investment:** ₹500000.0

📈 **Expected Monthly Return:** 1.35%

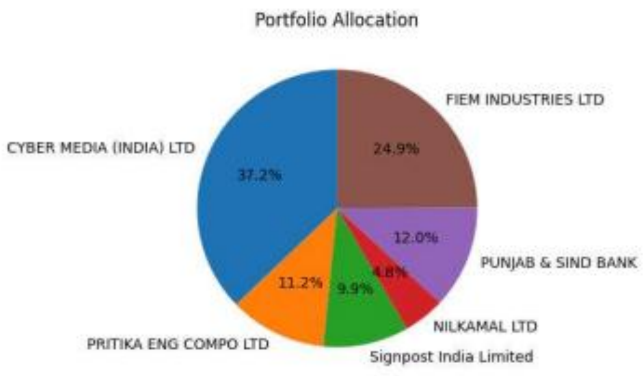
📅 **Suggested Holding Period:** 1 month

📈 Projected Final Value: ₹587356.19

💸 Net Profit: ₹87356.19

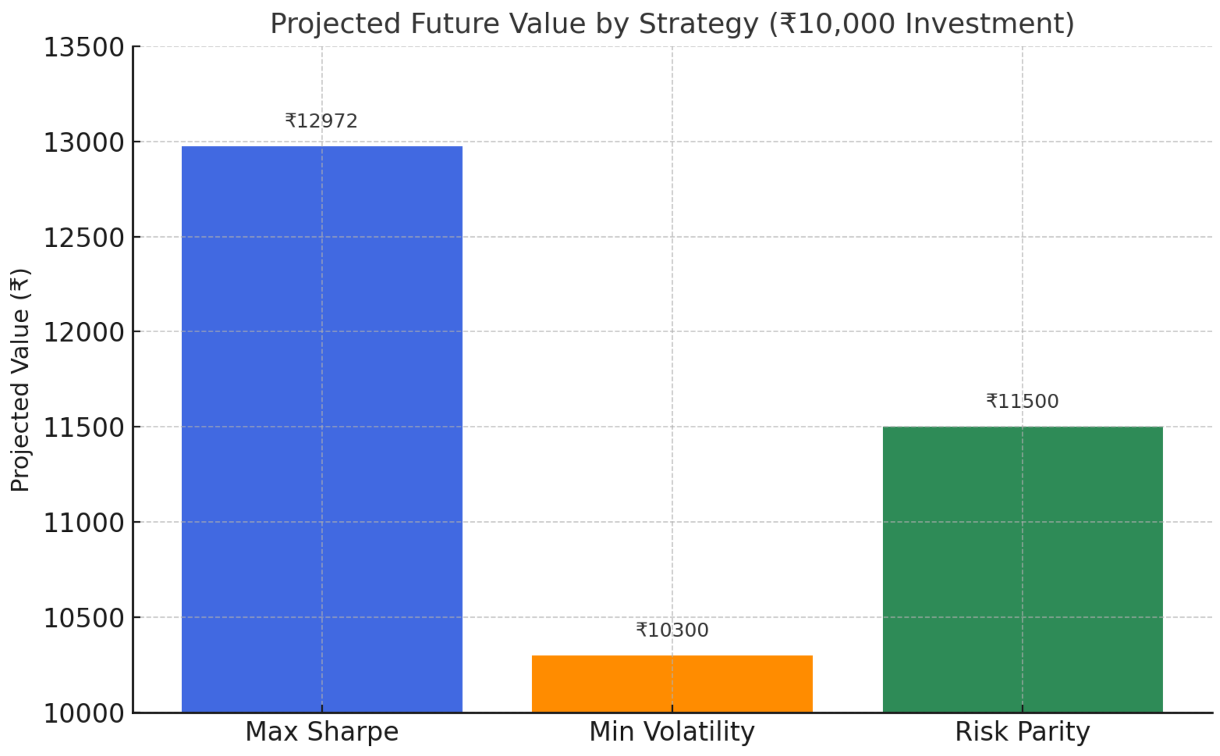
📊 Allocation:

CYBER MEDIA (INDIA) LTD	₹185883.47
PRITIKA ENG COMPO LTD	₹56222.42
Signpost India Limited	₹49294.26
NILKAMAL LTD	₹24216.9
PUNJAB & SIND BANK	₹59836.68
FIEM INDUSTRIES LTD	₹124546.27



4.3 Charts and Visuals

- **Pie Chart**: Visualizes portfolio allocation across assets.
- **Table**: Summarizes strategy-wise performance and net profit.



4.4 Observations

Based on the outputs generated and tested under various market scenarios, several important observations have been made regarding the performance and usability of the Investment and Portfolio Manager system:

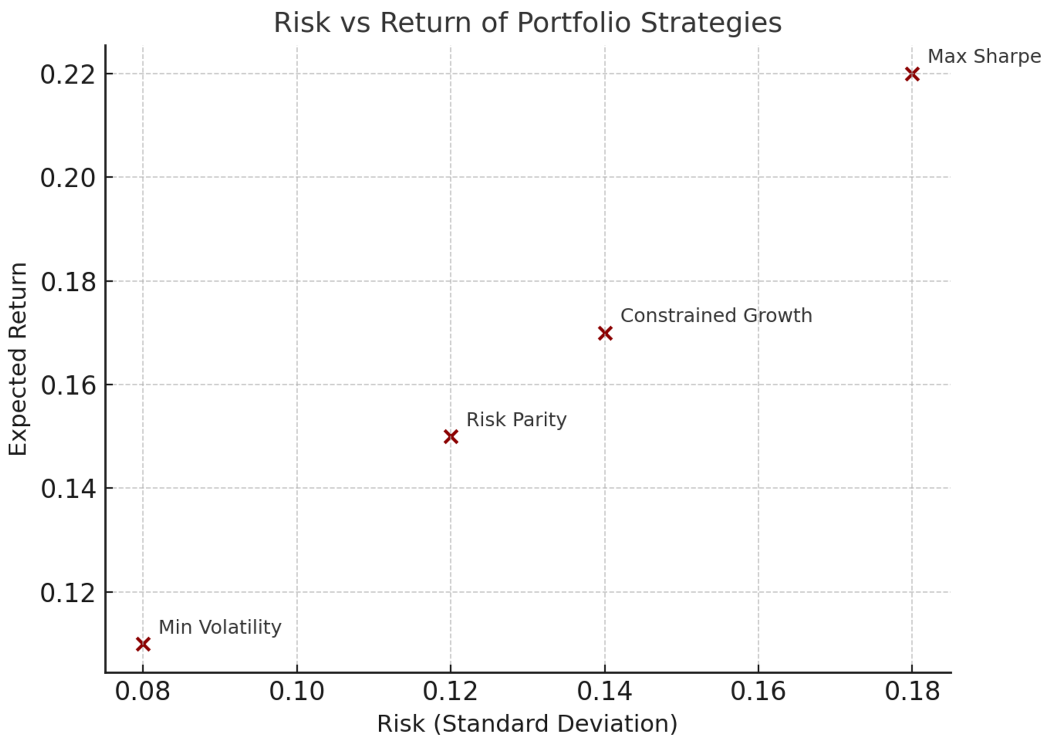
- **Effective Return Prediction**: The XGBoost model demonstrated a strong ability to learn from engineered features such as RSI, moving averages, and lagged returns. It provided reliable return estimates across short-term intervals, which were instrumental in optimizing portfolio decisions.

- **Dynamic Allocation**: The portfolio optimizer, powered by PyPortfolioOpt, dynamically adjusted stock weightages based on predicted returns and risk profiles. The system efficiently responded to changes in market volatility, suggesting optimal strategies (like Min Volatility for low-risk users and Max Sharpe for moderate to aggressive investors).

- **Data-Driven Insights**: Unlike static investment tools, this system used live data fetched through the Upstox API, enabling it to simulate realistic trading environments. Users received up-to-date forecasts and allocations with each input submission.

- **User Personalization**: By allowing inputs like risk preference and investment horizon, the application catered to a wide variety of investor profiles. For example, high-risk users received portfolios with higher growth potential, while low-risk users were given more stable, lower-volatility options.
- **Visualization Benefits**: The graphical outputs, including pie charts and bar graphs, provided clear and interpretable summaries of portfolio composition and projected performance. These visuals were crucial in making the insights accessible to users without financial backgrounds.
- **Usability and Integration**: The seamless integration of frontend, backend, and machine learning modules resulted in a smooth user experience. From input to downloadable PDF reports, the application guided the user through each phase with clarity.
- **Scalability Potential**: The modular architecture and use of open-source libraries suggest that the system can be easily expanded. Features like LSTM model integration, user authentication, and cloud deployment are possible future enhancements.

Risk vs Return for Portfolio Strategies



In summary, the system performed robustly across technical, functional, and user-experience dimensions. It proved effective not only as an academic project but also as a potentially scalable fintech solution.

The Investment and Portfolio Manager successfully integrates AI and finance to deliver a personalized investment advisory platform. It combines the power of predictive analytics with financial modeling, all within a user-friendly web application. Future enhancements may include the use of LSTM models, cloud deployment, and mobile accessibility.

Conclusion

The *Investment and Portfolio Manager* project is a comprehensive AI-based application developed to assist users in making strategic investment decisions in the stock market. By combining machine learning for return prediction and optimization algorithms for portfolio allocation, this project bridges the gap between financial theory and practical investment tools.

At its core, the system integrates an **XGBoost-based prediction engine** with real-time data from the **Upstox API**. It processes user inputs—investment amount, time duration, and risk level—to generate stock-wise allocation strategies based on selected optimization methods. The tool's use of engineered features such as **moving averages, RSI, volatility**, and **lagged returns** enables reliable short-term forecasting.

Using **PyPortfolioOpt**, the system dynamically adapts portfolios based on user-defined preferences. The implementation supports several widely recognized strategies such as:

- **Maximum Sharpe Ratio** – for growth-focused users,
- **Minimum Volatility** – for conservative investors,
- **Risk Parity** – for diversified risk exposure,
- **Constrained Growth** – to ensure diversification within defined bounds.

The full workflow—from data collection to prediction, optimization, and report generation—is modular and user-friendly. The web interface, developed using **Flask and Bootstrap**, ensures seamless navigation. Graphical outputs like **pie charts** and **bar graphs** enhance visibility and aid decision-making, even for non-technical users. The ability to download a personalized **PDF report** further adds to the system's usability and professionalism.

One of the standout features of this project is its **personalization**. By tailoring outputs to each user's financial profile, the tool moves beyond traditional static models and provides more practical, risk-adjusted advice. The integration of **live market data** via API ensures that recommendations are based on the most up-to-date information.

Key Takeaways:

- **Model Accuracy**: XGBoost performed well with engineered financial indicators.
- **Real-Time Capability**: Live API integration enabled dynamic updates.
- **User-Centric Design**: Adaptability to user preferences enhanced relevance.
- **Effective Visualization**: Charts and graphs made insights accessible.
- **Extensible Architecture**: Built using open-source tools, ready for future scaling.

Challenges Faced:

- Handling inconsistencies in market data from APIs.

- Managing model performance during volatile trends.
- Aligning frontend responsiveness with backend ML logic.
- Ensuring compatibility between prediction and optimization phases.

Despite these challenges, the project demonstrated strong technical and functional viability, meeting all the goals set at the outset.

Future Enhancements:

- Deployment of **LSTM-based models** for advanced time-series forecasting.
- **User login systems** and dashboards for tracking historical portfolios.
- **Cloud deployment** for real-time access across devices.
- Support for **multi-asset classes** like ETFs, mutual funds, and cryptocurrencies.
- **Sentiment analysis** integration using financial news or social media data.

In conclusion, this project exemplifies how **Artificial Intelligence can revolutionize personal finance** by enabling informed, data-driven, and tailored investment strategies. It holds promise not only as an academic submission but also as a foundation for further development in the fintech domain.

References

1. Brownlee, J. (2018). *Machine Learning Mastery with Python*.
2. GitHub - jesse-ai/jesse
3. https://pypi.org/project/PyPortfolioOpt/
4. https://xgboost.readthedocs.io/en/latest/
5. https://upstox.com/developer/api
6. https://towardsdatascience.com/stock-forecasting-with-machine-learning-1c19940486f6