ChatGPT

AI/ML-Driven Wealth Management Model for Indian Stocks

We aim to build a Python-based, CLI-driven "robo-advisor" that invests a lump sum (e.g. ₹1 Crore) in the Indian stock market according to user-specified goals and risk tolerance. In essence, the system takes inputs – initial capital, investment horizon (e.g. 5 years), and a risk/return preference (ranging from "very safe" to "very risky" with corresponding target returns) – and outputs a recommended portfolio of Indian stocks (blue-chip, mid/low-cap, sectors, indices, etc.) and their weightings. The highest priority is capital preservation: even aggressive portfolios should prioritize diversification and volatility control. This approach echoes modern robo-advisors, which "revolutionized wealth management by enhancing accessibility, cost-effectiveness, and efficiency" through algorithmic, automated advisory [1]. The user can select, say, a "Very Safe" profile (target ~5–8% annual return) or a "Risky" profile (~14–17% return), and the model will try to match that target while minimizing risk. By design, the model focuses on free data and tools, using open-source libraries and APIs throughout.

Data Acquisition and Sources

We gather data from free, reliable sources for Indian markets:

- **Price Data:** Use Python libraries like `yfinance` (free Yahoo Finance API) [2] and `nsepython` (NSE India API) [3]. For example, `yfinance` can download daily price series for Nifty, Sensex, or any stock ticker (e.g. `TCS.NS`, `RELIANCE.BO`) easily [2]. The `nsepython` library directly pulls data from NSE's public endpoints [3]. (Alpha Vantage's free API is another option: it offers global historical stock data and technical indicators via a free key [4] [5].) We can also use `pandas-datareader` or Selenium/BeautifulSoup to scrape historical data if needed.
- **Fundamentals & Indices:** Free fundamentals (P/E, P/B, etc.) are limited, but Alpha Vantage and Yahoo offer some financials. Official NSE/BSE sites or government filings can be parsed if available. We can track benchmark indices (Nifty 50, Sensex) via symbols `^NSEI`, `^BSESN` in `yfinance`.
- **Market News / Sentiment:** We fetch financial news headlines from free sources (e.g. Yahoo Finance News via `yfinance`, NewsAPI.org with a free tier, RSS feeds, or scraping news portals). These headlines feed into a sentiment analysis step. For example, we can apply a pretrained model like **FinBERT** (available on Hugging Face) to score news articles as positive/negative/neutral [6]. Academic studies show that "financial news is used along with historical stock price data to predict upcoming stock prices" improves accuracy [7]. Thus, by combining price data with news sentiment, our model captures both quantitative and qualitative signals.

Feature Engineering

After data collection, we compute features for modeling:

- **Technical Indicators:** Calculate moving averages, volatility (std of returns), RSI, MACD, Bollinger Bands, etc., using `pandas`, `numpy` or libraries like `TA-Lib` or `pandas_ta`. For each stock/ETF, we can use daily OHLCV data to compute short-term and long-term trends.
- **Fundamental Metrics:** If available, include P/E, market cap, dividend yield, etc. Even approximate values add context about valuation versus growth.
- **Time Features:** Include time-based features (month, quarter) if seasonality is relevant.
- **Sentiment Scores:** Using a model like FinBERT [6], convert news headlines into sentiment scores. For example, we could compute an average sentiment for each stock per day or week. These scores become features indicating market mood. Indeed, "the stock market is highly volatile…predicting stock price on historical data is hard" unless you incorporate news and other external factors [7].
- **Macro & Alternative Data (Optional):** Include free indicators like interest rates, CPI, or crude oil (all from free APIs) to capture broader trends. For diversification, we might also consider gold (though out of scope if strictly stocks).

All features are organized in a Pandas DataFrame indexed by date, with one column per feature/stock. We align data to the same timestamps and handle missing days (e.g. non-trading days).

Predictive Modeling

We use machine learning (ML) and deep learning (DL) to forecast returns or scores for each asset over the investment horizon:

- **Time-Series Forecasting:** Models like ARIMA/SARIMA (via `statsmodels`) or Facebook's Prophet can predict future prices or log-returns from past price series (capturing trends, seasonality). For non-linear patterns, an LSTM-based recurrent neural network (using `TensorFlow` or `PyTorch`) can learn temporal dependencies in the data.
- **Supervised ML:** We can train regressors (linear regression, Random Forests, Gradient Boosting like XGBoost/LightGBM) or even classification models to predict next-day/week returns or "up/down" signals. Features include recent technical indicators and sentiment scores. For example, a Random Forest might learn that a strong positive headline combined with an oversold RSI signals a jump. We split historical data into train/test (time-series CV) to avoid lookahead bias.
- **Ensemble/Stacking:** Combining multiple models often improves robustness. We might take a weighted average of ARIMA/LSTM regression outputs, or feed ML model residuals into another model. In practice, ML allows *dynamic* adaptation: unlike static models, ML "continuously adapts to real-time market conditions" and captures complex non-linear relationships [8]. CFA Research notes that ML methods can yield "superior risk-adjusted returns" and better diversification compared to traditional approaches [9]. We exploit this by tuning models on recent data and re-training periodically.

The model outputs an expected return (or score) for each asset (and the market index) over the chosen period (e.g. the next year or the entire horizon). These serve as inputs to portfolio optimization.

Portfolio Optimization and Risk Management

Once we have expected returns and risk estimates, we construct the portfolio. We use **Modern Portfolio Theory (MPT)** as our base. In particular, we leverage the PyPortfolioOpt library, which implements classical mean-variance optimization and related methods. The workflow is as follows:

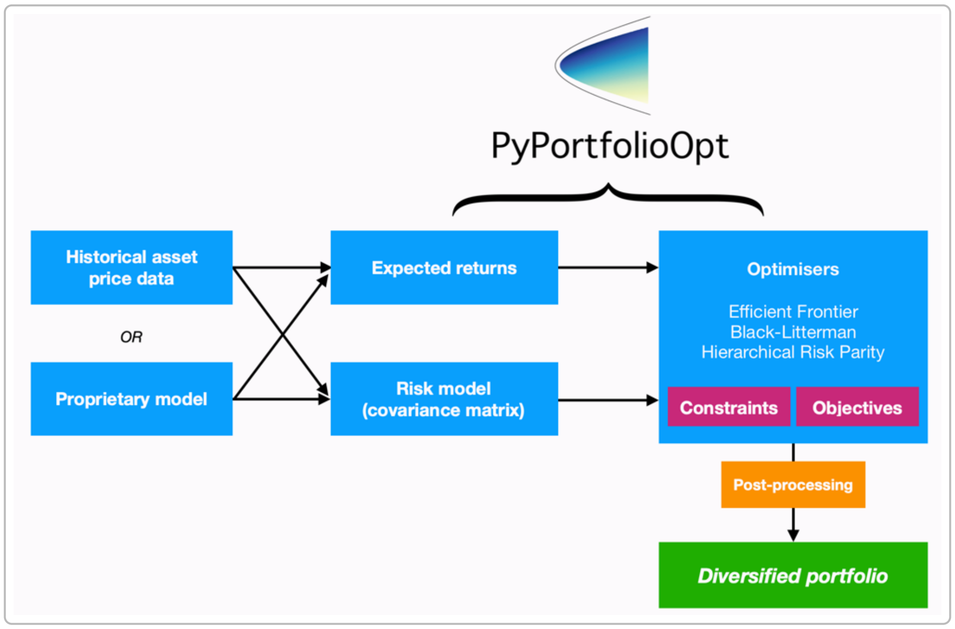

Figure: Conceptual flowchart of portfolio optimization (PyPortfolioOpt). Using historical prices or forecasts, we compute each asset's expected return vector (μ) and covariance matrix (Σ). Then we solve an optimization problem to find weights **w** that meet our objective. Typically, we target a specific expected return (per the user's risk level) and minimize portfolio variance, or equivalently maximize return for a given risk. This approach is grounded in Markowitz's classic mean-variance model [10] (which treats portfolio construction as a science rather than art).

For safety, we encode user risk profiles as constraints or targets in optimization. For example:

- **Low-risk ("Very Safe") profile:** Set a modest target return (e.g. 6%) and minimize variance. Constrain maximum weight in high-volatility stocks, and ensure minimum weight in stable large-caps (blue-chips) or index funds.
- **Medium-risk:** Target a higher return (e.g. 10%) with moderate variance allowed.
- **High-risk ("Risky"):** Allow up to, say, 15% target; permit more volatile assets (mid/small caps, cyclicals) but still diversify across sectors.

PyPortfolioOpt lets us specify either a *target return* or a *maximum volatility*, and it finds the efficient portfolio. We can also use advanced methods like **Hierarchical Risk Parity (HRP)** or Black-Litterman if desired. The resulting portfolio is diversified: it allocates weights across chosen stocks/sectors such that no single asset dominates risk. In practice, this might mean splitting the ₹1 Cr into, say, 40% in stable large-caps, 30% in midcaps, 10% in sectoral ETFs (tech, pharma, etc.), and 20% in index funds or bonds (to cap risk). By constraining weights and using the efficient frontier, the model seeks to "safely grow" the money for the period.

Python Libraries, Frameworks and Free Resources

We rely exclusively on free, open-source tools. Key Python libraries and resources include:

- **Data Fetching:** `yfinance` (free Yahoo Finance API) [2], `nsepython` for Indian market data [3], `pandas-datareader`, and web APIs (Alpha Vantage) [4].
- **Data Handling:** `pandas`, `numpy` for time-series manipulation.
- **Machine Learning:** `scikit-learn` (common algorithms), `xgboost` / `lightgbm` (gradient boosting), `statsmodels` (ARIMA), and Facebook's Prophet for forecasting. All are free to use.
- **Deep Learning:** `TensorFlow` / `Keras` and `PyTorch` (both open-source) for building LSTMs or MLP regressors.
- **Natural Language Processing:** `nltk` or `spaCy` for basic text processing, and Hugging Face's `transformers` library. We specifically use models like **FinBERT** (pretrained on financial text) for sentiment [6]. Hugging Face models are free to download and run locally.
- **Portfolio Optimization:** `PyPortfolioOpt` (implements Markowitz, CVaR, HRP, etc.), and `cvxpy` for custom convex optimization if needed. (PyPortfolioOpt's documentation even illustrates the above workflow【25†】.)
- **Backtesting & Metrics:** `backtrader` or `zipline` to simulate the strategy on historical data, and `empyrical` / `pyfolio` for performance metrics (Sharpe ratio, drawdown). These ensure our model has "high accuracy" in backtests.
- **Command-Line Interface:** Python's built-in `argparse` or the third-party `click` for a friendly CLI.
- **Other Tools:** `matplotlib` or `plotly` for any plotting; `pandas_ta` or `TA-Lib` for technical indicators; and free APIs like NewsAPI or even Twitter API (free tier) for additional data.

All the above libraries are open-source. For example, `yfinance` is "a free Python library" that interfaces with Yahoo Finance [2], and PyPortfolioOpt simplifies Markowitz MVO. Alpha Vantage explicitly advertises free access to its data with a free API key [4] [5]. No paid subscriptions are required.

System Workflow (CLI)

In practice, the CLI program will follow these steps (illustrative workflow):

1. **User Input:** Prompt the user to enter the investment amount (e.g. 10000000 for ₹1 Cr), the investment period (start and end dates or number of years), and a risk profile (e.g. "1" for Very Safe up to "5" for Very Risky).
2. **Data Retrieval:** Fetch historical price data for a chosen universe of assets (e.g. Nifty50 stocks, sector ETFs, and indices) over a look-back window (at least as long as the horizon). Also fetch recent financial news headlines.
3. **Feature Generation:** Compute all features – returns, indicators, sentiment scores – and assemble training data.
4. **Model Training/Prediction:** (Optionally) train the ML/DL models on past data, then use them to predict each asset's expected return over the investment period. Alternatively, run a pre-trained model or even skip modeling for very safe profiles (just use historical average returns).
5. **Portfolio Optimization:** Input the expected returns (μ) and risk (covariance Σ) into PyPortfolioOpt. Set up constraints according to the selected risk band (target return or max volatility). Solve for optimal weights.

6. **Output Recommendations:** Display the recommended portfolio: each stock/ticker name, its sector/category, and the percentage of the total capital to allocate. Optionally, show the projected return and portfolio volatility (Sharpe ratio, etc.).
7. **(Optional) Backtest Report:** Provide historical performance metrics (e.g. how this strategy would have performed in past 5 years) as validation.

The user can rerun the CLI anytime with updated data or a changed horizon. All components run sequentially in the script; no GUI is needed, satisfying the CLI requirement.

Accuracy, Risk Management and Backtesting

Ensuring high accuracy and safety involves extensive validation:

- **Backtesting:** Simulate the entire pipeline on past data. For each year in historical data, pretend it's the present, train the model on prior data, optimize the portfolio, and see the actual forward performance. This rolling backtest reveals real-world effectiveness and calibration.
- **Metrics:** Evaluate predictions with MAPE or RMSE, and portfolios with Sharpe ratio and max drawdown. Compare performance against benchmarks (e.g. Nifty50). High accuracy isn't guaranteed, but we aim for statistically significant outperformance of a naive benchmark.
- **Risk Controls:** Even for a "risky" profile, we can cap exposure to any single stock or sector to limit drawdowns. We may also impose a minimum allocation to cash or debt proxies (like bank deposits or gilt funds) if absolute safety is desired (though strictly out of "stock market" scope). Regular rebalancing (e.g. quarterly) can also manage drift.
- **Regular Updates:** The system should update data and re-run models periodically (monthly or quarterly) as markets evolve. ML models can be retrained with new data to adapt to changing regimes.
- **Safety Warning:** It must be emphasized that **no model can eliminate investment risk**. We can only minimize it through diversification and rigorous methodology. Even with "very safe" settings, short-term losses are possible. The model's design – borrowing from robo-advisors – prioritizes steady growth and capital preservation, but the user should understand market uncertainties.

Example Free Resources Summary

Below is a non-exhaustive list of the free tools and data feeds one can use:

- **Market Data:** `yfinance` [2], `nsepython` [3], AlphaVantage API [4] [5], official NSE historical data.
- **ML Libraries:** `scikit-learn`, `xgboost`, `lightgbm`, `statsmodels` (ARIMA), Prophet, `pandas`.
- **Deep Learning:** `TensorFlow/Keras`, `PyTorch`.
- **NLP:** `nltk`, `spaCy`, HuggingFace `transformers`. Use models like **ProsusAI/finbert** [6] for financial sentiment analysis.
- **Portfolio/Finance:** `PyPortfolioOpt`, `cvxpy`, `numpy`, `pandas`, `backtrader`.
- **CLI/Utilities:** Python's `argparse` or `click` for interface, `matplotlib` or `plotly` for any charts (optional).

All of these are open-source/free. For example, FinBERT on HuggingFace is a “pre-trained NLP model to analyze sentiment of financial text” [6]. AlphaVantage explicitly invites users to get a “free API key” for its data [4] [5]. By combining these libraries and APIs, we can implement the entire system at no monetary cost.

Conclusion

In summary, we propose a fully-functional, Python-based CLI tool that acts as an AI-powered wealth manager for Indian equities. It uses **machine learning and deep learning** to forecast market movements, sentiment analysis on news, and **modern portfolio theory** to allocate capital. The user picks an investment horizon and a risk/return target (e.g. 5–8% for very safe, up to 15%+ for very risky), and the program selects a diversified basket of stocks or indices to pursue that goal. This design mirrors contemporary robo-advisors [1] but is built entirely with free resources: data from Yahoo/NSE/AlphaVantage, ML libraries (scikit-learn, TensorFlow), NLP models (FinBERT), and portfolio tools (PyPortfolioOpt). By rigorously backtesting and regularly updating, the system strives for high accuracy and reliability.

Finally, it’s critical to note that **risk cannot be eliminated**, only managed. The model emphasizes conservative choices (blue-chip stocks, diversity, volatility control) especially for safer profiles. In all cases, the user’s specified target guides the optimization, but the model always seeks to maximize safety. With open-source tools and careful design, this CLI advisor can provide actionable investment advice (stock picks, sector allocations, indices) for any custom time frame, grounded in current market data and cutting-edge AI methods [10] 【25†】.

Sources: We rely on finance and ML literature and tools including Markowitz’s portfolio theory [10], industry research on AI-driven portfolios [8] [9], and practical guides to data and tools (e.g. `yfinance` [2], PyPortfolioOpt【25†】, FinBERT [6], etc.) to design this comprehensive model.

---

[1] AI-Driven Financial Advisory: The Rise of Robo-Advisors by Aditya Kashyap :: SSRN
https://papers.ssrn.com/sol3/papers.cfm?abstract_id=5268858

[2] Mastering yfinance: The Ultimate Guide to Analyzing Stocks Market Data in Python
https://www.marketcalls.in/python/mastering-yfinance-the-ultimate-guide-to-analyzing-stocks-market-data-in-python.html

[3] nsepython · PyPI
https://pypi.org/project/nsepython/

[4] [5] Free Stock APIs in JSON & Excel | Alpha Vantage
https://www.alphavantage.co/

[6] ProsusAI/finbert · Hugging Face
https://huggingface.co/ProsusAI/finbert

[7] Stock Prediction by Integrating Sentiment Scores of Financial News and MLP-Regressor: A Machine Learning Approach - ScienceDirect
https://www.sciencedirect.com/science/article/pii/S1877050923000868

[8] [9] It's Not Just What You Own, It's How Much: Machine Learning and the Portfolio Construction Imperative - CFA Institute Enterprising Investor
https://blogs.cfainstitute.org/investor/2025/08/13/its-not-just-what-you-own-its-how-much-ml-and-the-portfolio-construction-imperative/

[10] User Guide — PyPortfolioOpt 1.5.4 documentation
https://pyportfolioopt.readthedocs.io/en/latest/UserGuide.html