AI Project Report

Name:

Muhammad Umar

Roll No:

SU92-BDSFM-F24-003

Section:

3A

Project Report: Stock Trading Application Using SMA Cross Strategy

Introduction

This project implements a stock trading application that utilizes the Simple Moving Average (SMA) Cross strategy for backtesting trading strategies on stock data. The application is built using FastAPI for the backend, Streamlit for the frontend, and employs the Backtesting library to simulate trading strategies. The SMA Cross strategy is a widely used technique in financial markets, where two different SMAs (short-term and long-term) are used to generate buy and sell signals based on their crossing points.

Technologies Used

FastAPI: A modern, fast (high-performance) web framework for building APIs with Python.

Streamlit: An open-source app framework for Machine Learning and Data Science projects.

Backtesting: A Python library for backtesting trading strategies.

PyMuPDF (fitz): A library for extracting text from PDF files.

Pandas: A powerful data manipulation and analysis library for Python.

Requests: A simple HTTP library for Python to send requests to APIs.

Code Structure

The project consists of three main components:

Backend API (runtime.py**)**

Frontend Interface (streamlit_ui.py**)**

Data Extraction (simple.py**)**

Project Workflow

Loading Dataset

The initial step involves loading the dataset, which in this case includes historical stock prices. The data is typically extracted from a PDF or a CSV file containing stock information. This is crucial as it forms the foundation for all subsequent analyses and modeling.

In our project, the dataset is loaded using libraries like PyMuPDF to extract text from PDFs, which is then parsed and organized into a structured format (e.g., a Pandas DataFrame).

Data Exploration

Once the dataset is loaded, exploratory data analysis (EDA) is performed to understand the characteristics of the data. This includes examining statistical properties, identifying missing values, and visualizing trends.

EDA helps in gaining insights into the stock's historical performance, which is vital for making informed decisions regarding trading strategies.

Preprocessing

Data preprocessing involves cleaning and preparing the data for analysis. This may include handling missing values, normalizing data, and transforming features.

In our project, preprocessing ensures that the dataset is ready for analysis by removing any irrelevant information and structuring it appropriately for model training.

Splitting

The cleaned dataset is then split into training and testing sets. This division is essential to evaluate the performance of the trading strategy without overfitting to the training data.

A common approach is to use a time-based split where earlier data serves as the training set and later data as the test set, reflecting real-world trading scenarios.

Training / Applying Classifiers

In this phase, we apply classifiers or trading strategies to the training dataset. For our project, we implement the SMA Cross strategy, which involves calculating two SMAs (short-term and long-term) and generating buy/sell signals based on their crossover points.

The strategy is encapsulated within a class that defines how trades are executed based on these signals.

Testing & Processing Results

After applying the trading strategy to the test set, we analyze its performance by calculating various metrics such as total return, win rate, and maximum drawdown.

This step involves running simulations using backtesting frameworks to assess how well the strategy would have performed historically.

Displaying Accuracies

The results of the backtesting are summarized and displayed in a user-friendly format. This can include visualizations such as plots showing equity curves or summary statistics of trades executed.

In our project, we generate HTML reports that visualize these results, making it easier for users to interpret the effectiveness of their chosen strategies.

Application Phase

Finally, we deploy the application using FastAPI and Streamlit, allowing users to interact with the system through a web interface. Users can select stock tickers, input their parameters (such as initial balance and commission rates), and execute backtests in real-time.

This phase integrates all previous steps into a cohesive application that provides valuable insights into stock trading strategies.

Conclusion

The development of this stock trading application demonstrates an effective approach to utilizing historical data for backtesting trading strategies based on SMA crossovers. By following a structured workflow from data loading through to deployment, we can create an interactive tool that aids traders in making informed decisions based on quantitative analysis. Future enhancements could include integrating machine learning models for predictive analytics or expanding the range of available trading strategies within the application.