Development of an Automated Daily Trading System Using Python

Project Overview

Our project focuses on the development of an automated trading system that leverages data analytics and machine learning. This system integrates two key components:

1. **Data Analytics Model**: A machine learning model that predicts stock market movements for five highly correlated technology companies—Apple, Microsoft, Brown & Brown, Fastenal, and Old Dominion Freight Line.
2. **Web-Based Trading Application**: A Streamlit-powered interactive platform that allows users to visualize predictions, test trading strategies, and monitor trading performance in real time.

By combining these elements, we created a fully automated trading system capable of making data-driven decisions based on historical market data and predictive modeling.

Methodology

Data Processing (ETL)

- We collected historical stock price data from SimFin and Yahoo Finance APIs, focusing on daily stock prices over five years.
- A correlation analysis was performed to identify the most correlated stocks with Apple, ensuring the model's robustness.
- Feature engineering was conducted to prepare the dataset, utilizing key features like open, close, high, and low prices for the selected tickers.
- The final processed dataset was stored and utilized for model training.

Machine Learning Model

- Several iterations and experimentation were conducted using **Scikit-learn** and **XGBoost** models.
- The target variable was binary: 1 if the stock price was expected to increase the next day, 0 if expected to decrease.
- Data was split into training and testing sets, with an additional 30-day unseen data window for validation.
- The final **XGBoost model** achieved a **62% accuracy on the test set** and **60% accuracy on unseen data**, with a **precision of 57% for downward movement predictions and 64% for upward movement predictions**.

Backtesting and Simulation

- To evaluate model performance, we implemented a **backtesting strategy** using the backtest.py library in Python.
- The strategy followed a buy-only approach:
    - A position was entered when the model predicted a price increase (1).
    - A position was exited when the model predicted a price decrease (0) or when either the **take profit** or **stop loss** was reached.
- Users could customize parameters such as stock ticker, backtesting period, initial cash balance, and risk management levels.
- The backtest generated:
    - **Equity curve charts** showing portfolio growth.
    - **Profit/loss charts** per trade.
    - **Detailed candlestick charts** highlighting entry and exit points.
    - **Statistical summaries** featuring total trades, Sharpe ratio, and win percentage.

Web-Based Trading Application

- The application, developed using **Streamlit**, integrates multiple APIs and machine learning models.
- Real-time stock data is fetched via the **Yahoo Finance API**, while historical data is retrieved from **SimFin**.
- The application provides users with:
    - **Company Overview Page**: Displays key company details and market insights.
    - **Prediction Page**: Shows stock movement forecasts using the trained model.
    - **Backtesting Page**: Allows users to simulate trading strategies and visualize results interactively.
- The app updates stock data **every 20 seconds**, enabling users to track market movements and compare predictions in real-time.

Challenges Faced

- **Data Quality & Processing**: Handling missing data and ensuring accurate feature engineering were critical steps in developing a robust model.
- **Model Optimization**: Fine-tuning hyperparameters for the XGBoost model required multiple iterations to achieve the best accuracy.
- **API Integration**: Ensuring seamless integration between **SimFin, Yahoo Finance**, and our predictive model for real-time updates posed technical challenges.

- **Backtesting Complexity**: Designing a realistic yet computationally efficient backtesting framework was a key focus area.

Key Takeaways & Conclusion

- The project successfully demonstrated the feasibility of **machine learning-based stock prediction** and **automated trading strategies**.
- The final model achieved **reasonable predictive performance**, with **interactive visualization tools** enabling users to make informed decisions.
- Our **Streamlit-based trading app** serves as a **proof-of-concept** for real-time stock market analysis and strategy testing.
- Future improvements could include **incorporating additional financial indicators**, **enhancing model accuracy**, and **expanding the trading strategy beyond a simple buy-only approach**. Also, many improvements can be made within the UI/UX level, like integration with actual Stock Market Calendars.

This project showcases the potential of **data-driven trading automation**, empowering users with actionable insights to optimize their investment decisions. Thank you for reviewing our work!