UOBAM Coding Assessment

Question 1:

The `spx_analyzer.py` framework is designed to analyze S&P 500 (SPX) index data and calculate maximum returns under two different scenarios.

Class Structure

1. `SPXAnalyzer`: The main class that handles data loading, analysis, and return calculations.

Key Methods

1. `__init__(self, csv_file)`: Initializes the analyzer with data from the provided CSV file.
2. `_load_data(self, csv_file)`: Reads the CSV file and stores the data as a list of tuples (date, price).
3. `_calculate_returns(self)`: Computes daily returns based on the price data.
4. `_identify_trends(self, ...)`: Identifies market trends using various technical indicators (EMA, RSI, MACD).
5. `calculate_max_return_scenario1(self, ...)`: Calculates maximum return for Scenario 1.
6. `calculate_max_return_scenario2(self, ...)`: Calculates maximum return for Scenario 2.
7. `generate_report(self)`: Generates a comprehensive report comparing both scenarios and market performance.

Pipeline Flow

1. The script starts by loading the SPX data from a CSV file.
2. It then calculates daily returns and identifies trends using technical indicators.
3. For each scenario, it simulates trading based on the identified trends and calculates the maximum possible return.
4. Finally, it generates a report comparing the performance of both scenarios to the overall market performance.

Scenario 1: 2% Transaction Fee, No Turnover Limit

In the `calculate_max_return_scenario1` method:

1. The method iterates through the identified trends day by day.
2. It buys when a strong uptrend is detected (if not already in the market) and sells when a strong downtrend is detected (if in the market).
3. Each transaction incurs a 2% fee, applied to the portfolio value.
4. There's no limit on the number of transactions.
5. The method also implements safety checks to prevent unrealistic returns due to data anomalies.

Scenario 2: 0% Transaction Fee, Maximum 10 Transactions

In the `calculate_max_return_scenario2` method:

1. Similar to Scenario 1, it iterates through the trends day by day.
2. It uses a more stringent criteria for buying and selling, considering both trend strength and recent price changes.
3. No transaction fee is applied.
4. The method keeps track of the number of transactions and stops trading after reaching the maximum of 10 transactions.
5. It also implements a maximum holding period to prevent indefinite holding of positions.

Both scenarios aim to maximize returns by timing the market based on identified trends, but they operate under different constraints. Scenario 1 allows for more frequent trading but incurs transaction costs, while Scenario 2 has no transaction costs but limits the number of trades, requiring more selective entry and exit points.

Question 2:

- Analyzes and predicts S&P 500 (SPX) performance using machine learning models
- Implements a portfolio management strategy based on predictions

Key Components

1. **DataCollector**
    - Fetches SPX data and economic indicators
    - Uses yfinance and pandas_datareader libraries
2. **CustomDataLoader**

    - Loads data from CSV file
    - Filters data from 1989 onwards
3. **FeatureEngineer**
    - Creates features like returns, moving averages, volatility, RSI, MACD
    - Prepares data for model training
4. **ModelTrainer**
    - Implements multiple models:
        - OLS (Ordinary Least Squares)
        - ElasticNet
        - Random Forest
        - LSTM (Long Short-Term Memory neural network)
        - XGBoost
    - Trains models on historical data
5. **PortfolioManager**
    - Makes buy/sell decisions based on model predictions
    - Tracks portfolio performance and transactions
6. **BacktestEngine**
    - Runs backtests on historical data
    - Evaluates model performance

Main Workflow

1. Load and preprocess SPX data
2. Engineer features
3. Split data into training (before 2019) and test (2019-2023) sets
4. Perform rolling window backtest on training data
5. Train models on full training data
6. Evaluate models on test data (2019-2023)
7. Compare model performance to benchmark (buy-and-hold SPX)

Key Features

- Uses multiple machine learning models for prediction
- Implements a rolling window backtest for robust evaluation
- Compares performance against a simple buy-and-hold strategy

Molde Comparison Study

1. **ElasticNet Performance**:
    - Best performer in both scenarios
    - Significant drop in performance from rolling window (51.60%) to final test (40.65%)
    - More conservative in transactions during rolling window (17.36) compared to final test (20)
2. **LSTM Performance**:
    - Second-best in rolling window, but dropped to last in final test
    - Largest performance decrease from rolling window (45.67%) to final test (29.21%)
    - Suggests potential overfitting or sensitivity to specific market conditions
3. **OLS, RandomForest, and XGBoost**:
    - Remarkably consistent performance across both scenarios
    - All three models show identical or near-identical returns in both cases
    - Consistently used maximum allowed transactions (20)
4. **Transaction Behavior**:
    - Most models used all 20 transactions in both scenarios
    - ElasticNet and LSTM were more conservative in the rolling window, but used all transactions in the final test
5. **Consistency vs Volatility**:
    - OLS, RandomForest, and XGBoost showed high consistency but lower returns
    - ElasticNet and LSTM showed higher but more volatile returns
6. **Generalization**:
    - ElasticNet appears to generalize best from rolling window to final test
    - LSTM shows signs of overfitting, with poor generalization to the final test period
    - Other models show strong generalization, but at a lower performance level

Key Takeaways

1. **ElasticNet Superiority**: ElasticNet consistently outperformed other models, suggesting it captured market dynamics more effectively. Its ability to handle multicollinearity in financial data may contribute to its success.
2. **LSTM Volatility**: LSTM's significant performance drop highlights the challenges of using complex models in financial prediction. It may be capturing noise rather than signal in the rolling window tests.
3. **Consistency of Simpler Models**: OLS, RandomForest, and XGBoost showed remarkable consistency, suggesting they may be capturing fundamental, persistent market behaviors.

4. **Transaction Utilization**: The tendency for models to use all available transactions in the final test suggests they might be overly sensitive to short-term fluctuations.
5. **Rolling Window vs Final Test**: The discrepancy between rolling window and final test performances, especially for ElasticNet and LSTM, underscores the importance of robust out-of-sample testing in financial modeling.

This analysis reveals a trade-off between performance and consistency. While ElasticNet offers the highest returns, it also shows more volatility between scenarios. In contrast, simpler models offer more consistent, albeit lower, returns. The challenge lies in balancing the potential for higher returns with the need for consistent, reliable performance in varying market conditions.