Trading Project - Milestone 2

This project implements a quantitative trading strategy focusing on order execution optimization using machine learning and market microstructure analysis.

Project Structure

Core Components

- `main.py`: Main execution script that orchestrates the entire pipeline
- `data_loader.py`: Handles data loading and preprocessing
- `data_preprocessor.py`: Processes raw market data into structured format
- `factors.py`: Implements various market microstructure factors
- `targets.py`: Defines target variables for prediction
- `scoring_model.py`: Machine learning model for generating trading signals
- `strategy.py`: Core trading strategy implementation
- `model_evaluation.py`: Evaluates model performance
- `factor_analysis.py`: Analyzes factor performance and relationships

Data Flow

1. **Data Processing**
   - Raw market data is processed through `data_preprocessor.py`
   - Processed data is stored in `processed_data/` directory
   - Data includes order book, trades, and market microstructure features
2. **Feature Engineering**
   - `factors.py` calculates various market microstructure factors
   - Factors include spread, volatility, order flow imbalance, etc.
3. **Model Training**
   - `scoring_model.py` trains LightGBM models for each target
   - Models are saved in `models/` directory
   - Training uses top 50 features for each target
4. **Strategy Execution**
   - `strategy.py` implements the core trading logic
   - Uses dynamic thresholds based on market conditions
   - Records execution quality metrics

Key Outputs

1. **Model Outputs**
   - Model files: `models/{symbol}_{target}_model.joblib`
   - Score files: `results/{symbol}_{target}_scores.csv`
2. **Strategy Results**
   - Execution records in `strategy_results/` directory
   - Performance metrics and analysis reports
3. **Analysis Outputs**
   - Factor analysis results in `analysis/` directory

- Score distribution plots in `figure/` directory

Dependencies

Key dependencies are listed in `requirements.txt`: - pandas - numpy - lightgbm - matplotlib - scikit-learn

Reproduction Guide

1. Environment Setup

1. **Clone the repository**:

   ```
   git clone https://github.com/ZiyangLiuQuant/Algo-HFT-project.git
   cd Algo-HFT-project
   ```

2. **Create and activate a virtual environment**:

   ```
   # For Mac/Linux
   python -m venv venv
   source venv/bin/activate

   # For Windows
   python -m venv venv
   venv\Scripts\activate
   ```

3. **Install dependencies**:

   ```
   pip install -r requirements.txt
   ```

2. Data Preparation

1. **Download training data**:

   - The training data files are available in the `data/` directory:
     - `AMZN_train_data.csv`
     - `GOOG_train_data.csv`
     - `INTC_train_data.csv`
     - `MSFT_train_data.csv`
     - `AAPL_train_data.csv`

2. **Create necessary directories**:

   ```
   mkdir -p processed_data models results strategy_results figure logs
   ```

3. Running the Pipeline

1. **Data Preprocessing**:

   ```
   python data_preprocessor.py
   ```

This will process the raw data and create processed files in the `processed_data/` directory.

2. **Feature Generation and Model Training**:

   ```
   python main.py
   ```

   This will:

   - Generate features
   - Train models for each symbol
   - Save models and scores

3. **Strategy Testing**:

   ```
   python test_strategy.py
   ```

   This will test the trading strategy and generate results.

4. **Model Evaluation**:

   ```
   python model_evaluation.py
   ```

   This will evaluate model performance and generate analysis reports.

4. Expected Outputs

After running the pipeline, you should see:

1. **Processed Data**:
   - `processed_data/{symbol}_processed.feather` files
2. **Models**:
   - `models/{symbol}_{target}_model.joblib` files
3. **Results**:
   - `results/{symbol}_{target}_scores.csv` files
   - `strategy_results/` directory with execution records
4. **Analysis**:
   - `figure/` directory with various plots
   - Analysis reports in the root directory

5. Troubleshooting

1. **Memory Issues**:
   - If you encounter memory errors, try processing one symbol at a time
   - Modify `main.py` to process specific symbols
2. **Dependency Issues**:
   - Make sure all dependencies are correctly installed
   - Check Python version (recommended: Python 3.8+)
3. **Data Format Issues**:
   - Ensure data files are in the correct format
   - Check column names match the expected format

Notes

- All processed data is stored in feather format for efficient I/O
- Logs are maintained in the `logs/` directory
- Results are organized by symbol and target type