**Sprint-2**

**Introduction**

In this Sprint, the purpose was to deliver the first three components of the forecasting pipeline—DataAgent for fetching, imputing, normalizing, windowing, and splitting OHLCV data; TransformerAgent for encoding windowed inputs with a self-attention Transformer and training a next-day movement classifier; and EmbeddingAgent for freezing the trained model, extracting per-window embeddings, saving them with labels to CSV, and validating their integrity. The following sections contain the user stories I worked on with a detailed description of the tasks I worked on:

**User Stories**

I worked on the following User Stories:

[**TRF: Transformer-based Model with XGBoost for Stock Price Prediction #720**](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/720)

**Conditions of Satisfiability:**

1. Data Processing  
   o Condition: The system fetches, cleans, normalizes, and segments OHLCV data into windows without errors.  
   o Test: Fetch OHLCV for a ticker (e.g., AAPL), normalize features (z-score), and create train/validation/test splits with 20-day windows.  
   o Satisfaction: Data is correctly processed and usable for model training.
2. Transformer Model  
   o Condition: The Transformer model trains end-to-end on OHLCV windows, converges on validation data, and produces valid embeddings.  
   o Test: Train the model on sample data and verify that validation loss decreases and embeddings have the expected shape.  
   o Satisfaction: Model trains without crashing and generates meaningful embeddings.
3. Embedding Extraction  
   o Condition: The system extracts Transformer embeddings and pairs them with true labels for XGBoost training.  
   o Test: Forward test windows through the Transformer and save embeddings to disk (e.g., CSV).  
   o Satisfaction: Embeddings are correctly saved and aligned with labels.
4. XGBoost Training  
   o Condition: XGBoost trains on Transformer embeddings and improves classification accuracy over the standalone Transformer.  
   o Test: Compare hold-out accuracy of XGBoost vs. Transformer alone on test data.  
   o Satisfaction: XGBoost achieves higher accuracy than the Transformer baseline.
5. Inference Pipeline  
   o Condition: The inference pipeline processes a new OHLCV window and outputs the required JSON within 5 seconds.  
   o Test: Run inference on a sample window and measure latency.  
   o Satisfaction: JSON output is correct, and inference completes within 5 seconds.
6. CrewAI Decision  
   o Condition: The CrewAI agent correctly maps xgb\_prob to BUY/SELL/HOLD recommendations.  
   o Test: Input sample JSONs with varying xgb\_prob values and verify recommendation logic.  
   o Satisfaction: Recommendations are consistent with probability thresholds.
7. Backtesting with Backtrader  
   o Condition: Backtrader evaluates the model’s predictions over historical data, producing metrics like accuracy, return, and drawdown.  
   o Test: Run backtesting on a year of historical data and verify metrics align with expected outcomes.  
   o Satisfaction: Backtesting results are accurate and include visualizations of performance.
8. Visualization  
   o Condition: The system generates clear charts showing predicted vs. actual price movements and backtesting performance.  
   o Test: Generate charts for a test ticker and verify they accurately reflect predictions and backtesting results.  
   o Satisfaction: Charts are interpretable and correctly represent model outputs.
9. Performance  
   o Condition: The system handles large datasets and multiple predictions without significant latency.  
   o Test: Run inference on 100 windows and verify average latency is under 5 seconds.  
   o Satisfaction: System performs efficiently under load.
10. Security  
    o Condition: OHLCV data fetching and storage comply with secure practices.  
    o Test: Verify that yfinance API calls use secure connections and data is stored securely.  
    o Satisfaction: No vulnerabilities in data handling are detected.

**Definition of Done**

1. Functional Requirements  
   o OHLCV data is fetched, cleaned, normalized, and segmented into windows.  
   o Transformer model trains and produces embeddings.  
   o XGBoost trains on embeddings and predicts next-day price movements.  
   o Inference pipeline outputs JSON with transformer\_score and xgb\_prob.  
   o CrewAI agent delivers BUY/SELL/HOLD recommendations based on xgb\_prob.  
   o Backtrader backtests predictions and generates performance metrics.  
   o Charts visualize predictions and backtesting results.
2. Non-functional Requirements  
   o Inference completes in ≤ 5 seconds per window.  
   o System handles large datasets without performance degradation.  
   o Data fetching and storage are secure and compliant with best practices.
3. Testing and Validation  
   o Unit tests cover data processing, model training, embedding extraction, and inference.  
   o Integration tests verify the end-to-end pipeline (data → Transformer → XGBoost → CrewAI → Backtrader).  
   o Backtesting results are validated against historical data.
4. Integration  
   o Code is integrated into the existing AI-Agent-Stock-Prediction codebase on GitHub.  
   o Pull request is submitted with all tasks implemented and tested.
5. Documentation  
   o Documentation covers system architecture, data pipeline, model details, and backtesting setup.  
   o Instructions for running the pipeline and interpreting outputs are provided.
6. Deployment  
   o Pipeline is executable within the existing codebase environment (no Jupyter/Kaggle).  
   o All critical bugs are resolved, and no high-priority defects remain.

**Tasks**

[TRF.1: Data Ingestion & Windowing (20 ph) #721](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/721)

[TRF.1.1: Fetch OHLCV data for target tickers via yfinance (4 ph). #722](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/722)

[TRF.1.2: Impute missing data and align timestamps (4 ph). #723](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/723)

[TRF.1.3: Normalize features using z-score (4 ph). #724](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/724)

[TRF.1.4: Slice data into 20-day windows with train/validation/test splits (8 ph). #725](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/725)

[TRF.2: Transformer Model Implementation (24 ph) #726](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/726)

[TRF.2.1: Implement Transformer architecture with self-attention in PyTorch (10 ph). #727](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/727)

[TRF.2.2: Add classification head for price movement prediction (6 ph). #728](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/728)

[TRF.2.3: Write training loop with loss, optimizer, and early stopping (6 ph). #729](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/729)

[TRF.2.4: Log metrics and save best model checkpoint (2 ph). #730](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/730)

[TRF.3: Embedding Extraction (12 ph) #731](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/731)

[TRF.3.1: Freeze Transformer model and extract embeddings for all windows (4 ph). #732](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/732)

[TRF.3.2: Save embeddings and labels to CSV (4 ph). #733](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/733)

[TRF.3.3: Validate embedding dimensions and label alignment (4 ph). #734](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/734)

[TRF.4: XGBoost Training (16 ph) #735](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/735)

[TRF.4.1: Load embeddings and labels into pandas DataFrame (2 ph). #736](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/736)

[TRF.4.2: Perform train/validation split (2 ph). #737](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/737)

[TRF.4.3: Grid-search XGBoost hyperparameters (n\_estimators, max\_depth, learning\_rate) (8 ph). #738](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/738)

[TRF.4.4: Train and save best XGBoost classifier (4 ph). #739](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/739)

[TRF.5: Inference Pipeline (12 ph) #740](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/740)

[TRF.5.1: Write script to load Transformer and XGBoost models (4 ph). #741](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/741)

[TRF.5.2: Compute transformer\_score and xgb\_prob for a new window (4 ph). #742](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/742)

[TRF.5.3: Emit JSON with date, scores, and probabilities (4 ph). #743](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/743)

[TRF.6: CrewAI Decision Agent (10 ph) #744](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/744)

[TRF.6.1: Implement logic to map xgb\_prob to BUY/SELL/HOLD (4 ph). #745](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/745)

[TRF.6.2: Validate recommendation logic with test cases (4 ph). #746](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/746)

[TRF.6.3: Output JSON with recommendation (2 ph). #747](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/747)

[TRF.7: Backtesting with Backtrader (20 ph) #748](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/748)

[TRF.7.1: Implement Backtrader strategy using model predictions (8 ph). #749](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/749)

[TRF.7.2: Run backtesting on historical data (6 ph). #750](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/750)

[TRF.7.3: Calculate performance metrics (accuracy, return, drawdown) (6 ph). #751](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/751)

[TRF.8: Visualization of Results (16 ph) #752](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/752)

[TRF.8.1: Create charts for predicted vs. actual price movements (6 ph). #753](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/753)

[TRF.8.2: Generate equity curve and performance metrics visualizations (6 ph). #754](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/754)

[TRF.8.3: Ensure charts are clear and interactive (4 ph). #755](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/755)

[TRF.9: Integration Testing & Performance (16 ph) #756](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/756)

[TRF.9.1: Write integration tests for data → Transformer → XGBoost → CrewAI pipeline (6 ph). #757](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/757)

[TRF.9.2: Benchmark end-to-end latency to meet ≤ 5 s target (6 ph). #758](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/758)

[TRF.9.3: Validate backtesting results against historical data (4 ph). #759](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/759)

**Tasks I Worked On**

[TRF.1: Data Ingestion & Windowing (20 ph) #721](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/721)

I fetched OHLCV data for target tickers via yfinance, imputed missing values and aligned timestamps, normalized features using z-score, and sliced the data into 20-day windows with train, validation, and test splits. The task was estimated at 20 person hours but it took me 26 person hours to complete.

[TRF.2: Transformer Model Implementation (24 ph) #726](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/726)

I implemented a self-attention Transformer architecture in PyTorch, added a classification head for price movement prediction, wrote a training loop with loss calculation, optimizer steps, and early stopping, and logged metrics while saving the best model checkpoint. The task was estimated at 24 person hours but it took me 25 person hours to complete.

[TRF.3: Embedding Extraction (12 ph) #731](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/731)

I froze the trained Transformer model to extract embeddings for all windows, saved those embeddings and their labels to a CSV file, and validated that the embedding dimensions and label alignment were correct. The task was estimated at 12 person hours but it took me 15 person hours to complete.

**Summary Table of Work**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| UserStory GitHub Issue ID | User Story | Story Points | Task GitHub Issue ID | Task | Task Hours | Status | Actual Hours |
| [TRF](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/720) | [Transformer-based Model with XGBoost for Stock Price Prediction #720](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/720) |  | [TRF.1](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/721) | Data Ingestion & Windowing (20 ph) #721 | 20 | Complete | 26 |
| [TRF](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/720) | [Transformer-based Model with XGBoost for Stock Price Prediction #720](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/720) |  | [TRF.2](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/726) | Transformer Model Implementation (24 ph) #726 | 24 | Complete | 25 |
| [TRF](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/720) | [Transformer-based Model with XGBoost for Stock Price Prediction #720](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/720) |  | [TRF.3](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/731) | Embedding Extraction (12 ph) #731 | 12 | Complete | 15 |

**Summary Table of Commits**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Date | Commit Number | Commit Description (exactly as in github) | User Story | Task |
| June 22nd, 2025 | a273bab868b6cad3bb731115ee041b77371e7996 | [Adding Transformer model implementation and Embedding extraction](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/pull/766/commits/a273bab868b6cad3bb731115ee041b77371e7996) | [TRF](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/720) | [TRF.1](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/721)  [TRF.2](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/726) |
| June 22nd, 2025 | 1e0cd4847e7f8f6a48ea333986438718ef3763f0 | [Adding embeddings](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/pull/766/commits/1e0cd4847e7f8f6a48ea333986438718ef3763f0) | [TRF](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/720) | [TRF.3](https://github.com/Rivier-Computer-Science/AI-Agent-Stock-Prediction/issues/731) |