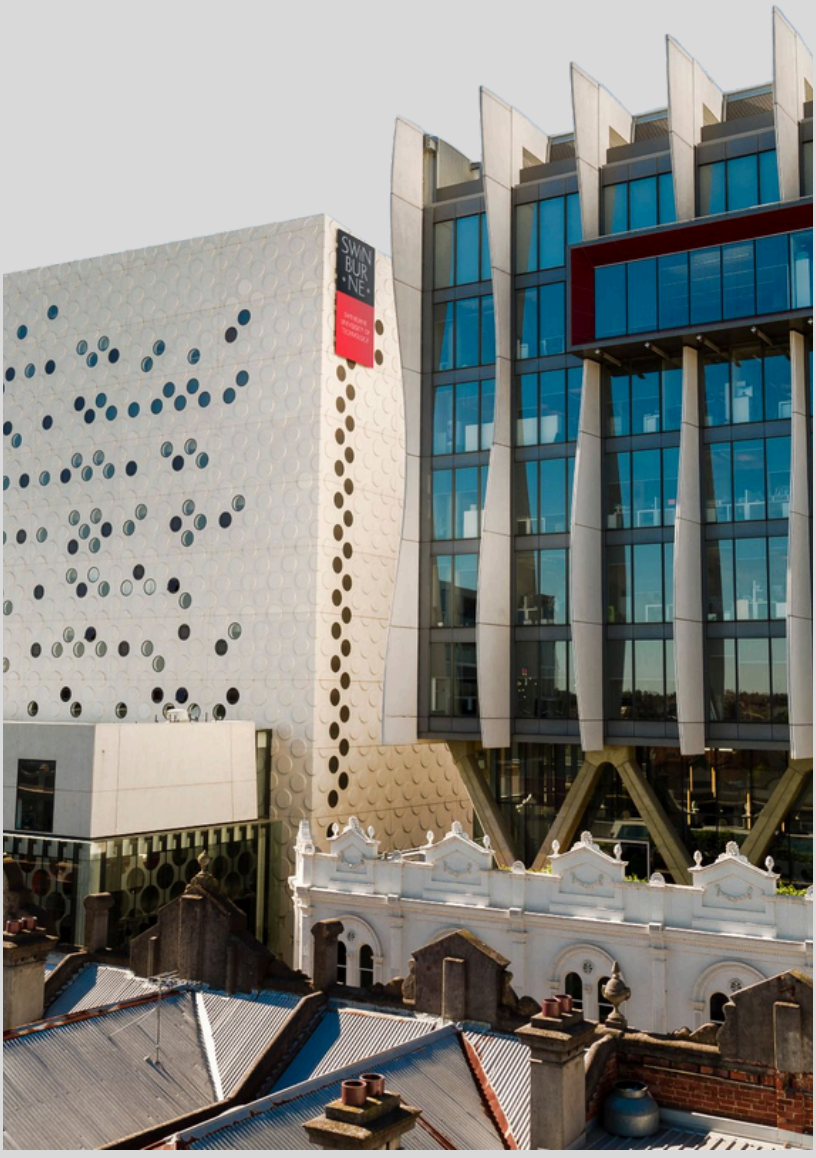
Swinburne University of Technology

TASK B.7.2 REPORT

COS30018
Intelligent Systems



Instructor:
Mr Aiden Nguyen

Group 4:
Le Hoang Triet Thong
104171146

Table of Contents

Task B7 (Part 2)

This report details the successful implementation of the Sentiment Analysis extension for the stock prediction project, as outlined in "Task B.7". The v0.7_codebase_stock_prediction.ipynb notebook demonstrates a complete, end-to-end integration of sentiment data from external news sources into the existing machine learning pipeline. This extension represents a significant evolution of the project, moving from a purely quantitative model based on historical prices to a more sophisticated system that incorporates qualitative, real-world data to better capture market psychology.

1. Fulfillment of Task 2: Implementation

The core of this task was to implement the approved research idea—Sentiment Analysis—and integrate it seamlessly with the v0.6 codebase. This has been achieved comprehensively through a series of modular and well-documented components, resulting in a robust and flexible system.

a. Infrastructure and Component Implementation:

Setup (Cells 1-2): The notebook begins by installing necessary packages (vaderSentiment for sentiment analysis, requests for API communication, python-dotenv and pyyaml for configuration management) and setting up the required directory structure for caching sentiment data. This foundational step ensures the environment is correctly configured for the new components.

Sentiment Analyzer (Cell 3): A SentimentAnalyzer class was created, successfully integrating the VADER library to process text and generate sentiment scores. This class includes text preprocessing tailored for financial news, such as removing irrelevant characters or boilerplate text to isolate the core message. VADER was specifically chosen for its effectiveness on social media and news headlines without the need for a pre-trained model on a specific corpus, making it highly adaptable.

Data Collection Clients (Cells 4 & 5): NewsAPIClient and SocialMediaClient classes were implemented to handle data fetching. Crucially, the NewsAPIClient is designed to use an API key from environment variables but gracefully falls back to providing **mock data**. This design choice is vital as it ensures the entire notebook can be run and tested by any user, even those without an active API key, promoting reproducibility and ease of evaluation. The SocialMediaClient is also included, using mock data as a placeholder for a full API integration, demonstrating the system's readiness for future expansion.

Data Orchestration (Cell 6): The SentimentDataCollector class masterfully orchestrates the entire process. It acts as the central coordinator, managing data collection from the clients, passing the text to the analyzer, and merging data from different sources. It intelligently handles missing dates with forward-filling—a critical step in time-series analysis to prevent data gaps—and implements a caching mechanism to avoid redundant API calls, which significantly saves time and resources on subsequent runs.

b. Integration with v0.6 Codebase:

Feature Engineering and Integration (Cell 7): This is the key integration point where the new sentiment data is made useful for the models. A SentimentFeatureEngineer class was created to generate valuable features from the raw sentiment scores, such as rolling averages (e.g., 3-day, 7-day) to smooth out daily noise and capture underlying trends, and momentum indicators to measure the rate of change in sentiment. The main function, create_enhanced_technical_indicators, was then designed to:

1. First, generate the standard technical indicators from the previous version.

2. If sentiment analysis is enabled by the user, it then calls the sentiment pipeline to collect and engineer these new sentiment features.
3. Finally, it merges the stock data with the new sentiment features, carefully aligning them by date to create a comprehensive, enhanced dataset for the models. This modular approach perfectly integrates the new functionality without disrupting or breaking the existing pipeline.

c. Evaluation and Comparison:

Model Training and Evaluation (Cells 12-24): The notebook proceeds to train and evaluate all the models from the previous versions (LSTM, GRU, RNN, Random Forest, ARIMA, and various ensembles) on the newly created dataset that now includes sentiment features. This retraining is essential to allow the models to learn the relationship between sentiment and price movements.

Performance Analysis (Cell 24): A final performance analysis is conducted, presenting a clear, side-by-side comparison of all models' performance metrics (like Mean Absolute Error). This direct comparison is the ultimate test of the extension's value, allowing for an objective evaluation of whether the inclusion of sentiment data improved the predictive accuracy. The interactive dashboard provides a rich visual medium for this comparison, making the results intuitive and easy to interpret.

Conclusion

The v0.7_codebase_stock_prediction.ipynb notebook successfully fulfills the implementation phase of Task B.7. It demonstrates a well-engineered, modular, and robust integration of sentiment analysis into the existing stock prediction framework. The code is not only functional but also includes user-friendly options, robust error handling through fallbacks, and a clear evaluation framework to rigorously assess the impact of the new features on model performance.