Final Project Report

Group16

Title: Stock Price Prediction using Stacking Ensemble Learning

1. Abstract

Predicting stock price movements remains a significant challenge in financial machine learning due to the high noise, instability, and stochastic nature of market data. This project investigates the effect of various machine learning models in forecasting the price direction of TSMC (Taiwan Semiconductor Manufacturing Company). Initially, our research focused on regression-based price point forecasting; however, preliminary experiments revealed a persistent "lagging" problem, where model predictions trailed actual price movements.

To overcome this, we shifted our strategy to a binary classification approach, aiming to predict market direction (Up/Down) rather than absolute values. We implemented a three-layer system architecture: a Preprocessing Layer; a Heterogeneous Base Learners Layer comprising six models (KNN, LSTM, XGBoost, Random Forest, Naive Bayes, and NN); and a Stacking Ensemble Layer.

Our findings demonstrate that the Stacking Ensemble model successfully synthesized diverse predictive signals, achieving a Stacking Accuracy of 65.53% on the independent test set (2025 stock price) and accuracy of 55-65% on single model.

2. Introduction

2.1 Background:

Stock price prediction is one of the most challenging topics in finance and machine learning. The stock market is highly complex and influenced by many unpredictable factors, such as economic policies, global news, and investor sentiment. As a result, financial data is full

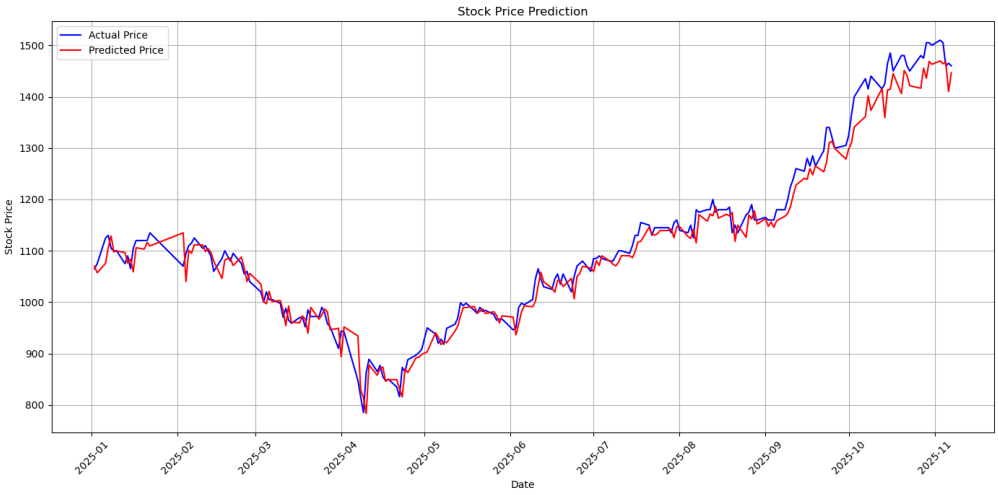
of noise and instability. Even with large amounts of historical data, it is very difficult to find clear patterns, making accurate forecasting a significant challenge.

2.2 Motivation:

In the past, researchers mainly used statistical models like ARIMA or GARCH. While these models are good at finding simple, linear relationships, they often fail to capture the complex, non-linear patterns hidden in modern financial markets. This limitation has led to the rise of Machine Learning and Deep Learning methods. These modern techniques are better suited for handling complex data interactions. In this project, we apply these advanced models to see if they can outperform traditional methods in predicting stock movements.

2.3 Core Findings:

At first, our goal was to predict the exact future stock price. However, after testing our basic models, we found a significant "lagging" problem. As shown in the Figure, the model's prediction was always following the trend but was one or two days behind the real price. This means the model was only learning from the past instead of predicting the future. We reviewed some studies and found that this is a widespread problem. Many researchers face this same issue, and currently, there appears to be no effective solution to completely fix it. Because of this, we decided to change our strategy. Instead of predicting the exact price, we switched to a classification approach to improve our results.



3. Related Work

3.1 Applications of Machine Learning in Stock Trend Prediction

Patel et al. (2015) [1]conducted a comparative study on various machine learning algorithms, including **Artificial Neural Networks (ANN)**, **Support Vector Machines (SVM)**, **Random Forest**, and **Naive Bayes**, specifically evaluating their performance in predicting price movement directions. Their research highlights that the effectiveness of financial forecasting models is highly contingent upon the **"representation of input data."** By transforming continuous technical indicators into discrete **"Trend Deterministic Data"** (such as +1 or -1), the researchers demonstrated a significant improvement in predictive accuracy. This preprocessing approach effectively filters out micro-noise within raw financial data, providing a more intuitive reflection of underlying market momentum.

3.2 Shifting Predictive Targets: From Stock Prices to Trading Signals

Beyond input data processing, the selection of the predictive target is equally critical. Research, such as that by Finlab [2], highlights that utilizing machine learning to forecast absolute stock price values is statistically

---

[1] Predicting stock and stock price index movement using Trend Deterministic Data Preparation and machine learning techniques https://www.sciencedirect.com/science/article/abs/pii/S0957417414004473

[2] 機器學習真的無法預測股價嗎？https://www.finlab.tw/ml-can-not-predict-price

challenging and frequently results in **predictive lagging**. In practice, a more effective approach is to position machine learning for **direct trading signal generation**. Instead of demanding precise price-point forecasts, models are trained to estimate the probability of an upward trend given the current feature set. This strategic shift from **"Regression-based forecasting"** to **"Classification-based decision making"** significantly mitigates the high-variance issues inherent in financial time-series data.

3.3 Limitations of Conventional and Single Machine Learning Models

The experiments by Patel et al. (2015) demonstrated that while discretized features can enhance performance, standalone models still struggle with non-stationary financial data, often leading to overfitting or performance collapse during Regime Shifts. Consequently, this project recognizes the necessity of a Stacking Ensemble Learning Architecture. By utilizing multiple Base Learners to capture diverse non-linear structures within the data and integrating their output probabilities, the model can maintain predictive robustness even in extremely volatile market environments.

4. Methodology

4.1 Data Pipeline:

To ensure a robust evaluation of our models, we established a standardized data pipeline encompassing data collection, feature construction, and preprocessing.

- **Data Source :**

  Daily trading data for TSMC were retrieved via the twstock API. To capture the impact of global markets, we also integrated key international indicators, specifically the Dow Jones Industrial Average (DJI) and TSMC's U.S.-listed American Depositary Receipt (ADR). Including these external variables allows the model to learn from cross-market dynamics and international trend influences.

- **Base Feature Pool:**

After initial data cleaning, the dataset consists of 36 core features. This feature pool includes: Basic Market Data(Daily Open, High, Low, Close prices...), Technical Indicators (Moving Averages, Relative Strength Index...) and US market Influence (DJI index, NASDAQ index, stock price of TSMC in the US market...).

- **Preprocessing:**

To guarantee fair and robust evaluations, we adopted a two-tier preprocessing strategy. While all models share a consistent Base Feature Pool to ensure they have access to the same fundamental market information, we applied tailored feature engineering for each specific model architecture. For instance, dimensionality reduction (PCA) was implemented for KNN to mitigate noise, while temporal sliding windows were constructed for LSTM and NN to capture sequential dependencies. This approach ensures that performance variations stem from each model's specialized processing capabilities rather than differences in the underlying data.

4.2 System Architecture:

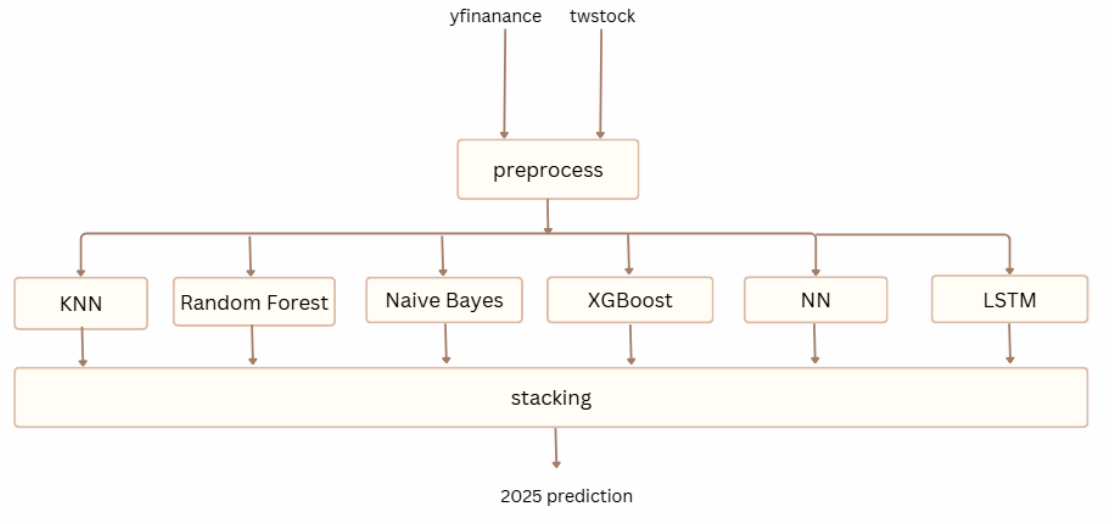
- As illustrated in **Figure 1**, the proposed system architecture is structured into three primary functional layers:
    - **1. Preprocessing Layer:** Historical price data of TSMC from 2010 to 2024 are used as the primary training dataset, while data from 2025 are reserved for evaluating the model's out-of-sample performance. This layer begins with feature engineering, where key technical indicators, such as the Moving Average Convergence Divergence (MACD), Relative Strength Index (RSI), and Bollinger Bands are computed to enhance the input features. Subsequently, data preprocessing is performed to ensure data quality and consistency. This includes handling missing values, scaling features using standardization techniques, and splitting the dataset into training and testing subsets.
    - **2. Heterogeneous Base Learners Layer:** Processed data is fed in parallel into six distinct core models. Each model is designed to capture specific non-linear structures within the financial time-series:
        1. **KNN** (K-Nearest Neighbors): Identifies spatial similarities in the feature space.

2. **LSTM** (Long Short-Term Memory): Capture long-term temporal dependencies and sequential trends that traditional static models often miss.
3. **XGBoost** (Extreme Gradient Boosting): Models complex non-linear interactions between different technical indicators.
4. **Random Forest:** Provides robust ensemble learning through bagging, effectively reducing overfitting by averaging multiple decision trees.
5. **Naive Bayes:** Calculates the conditional probability of market directions based on feature distributions, providing a probabilistic baseline that is computationally efficient.
6. **Neural Network (NN/MLP):** Employs a multi-layer perceptron structure with non-linear activation functions to learn high-level abstract representations of the input features.

- **3. Stacking Ensemble Layer:**

The outputs from the base learners—the predicted "probability of upward movement"—are aggregated and fed into a Meta-Learner (Stacking Model). In this framework, each base model is trained on historical data from previous years and generates predictions for the period **2020 to 2024**. These predictions serve as the input features for the stacking model, allowing it to synthesize the diverse predictive signals. The Meta-Learner then generates the final market direction forecast for the 2025 independent test set based on the 2025 predictions from each base model.

- Figure 1



4.3 Base Learners: ◦

- **Cluster-Augmented KNN with Dimensionality Reduction**

1. Advanced Feature Engineering (117 Dimensions)

Drawing inspiration from the "Trend Deterministic Data" concept proposed by Patel et al. (2015), the raw dataset was expanded into a 117-dimensional feature space. The selection logic is categorized into four distinct groups:

- Multi-dimensional Technical Indicators (with Trend Discretization): Beyond standard continuous metrics (MA, EMA, MACD, RSI), a wide array of binary trend signals were implemented (e.g., MA crossovers, RSI overbought/oversold triggers). This discretization mitigates the impact of high-frequency noise on KNN distance calculations, focusing the model on critical market momentum pivots.
- Global Market Interdependency Features: Integration of the four major U.S. indices (NASDAQ, S&P 500, DJI, SOX) and their daily returns. These features capture the spillover effects of the international semiconductor industry on the Taiwan stock market, providing the model with cross-market prior information.
- Temporal Lag Features: Calculation of price inertia and indicator states over a lookback period of 1 to 10 days. By incorporating these lags, the static KNN model gains the ability to capture dynamic changes within the temporal dimension of financial time-series.
- Candlestick Pattern Quantization: Mathematical representation of body size, shadow ratios, and open-close relative positioning. These metrics enable the model to identify technical support and resistance levels, reflecting extremes in market sentiment.

2. Logical Imputation Strategy

To maintain the physical integrity of financial data, the fill_na_safely function was executed with specific logic:

- **Binary Event Signals:** Missing values are filled with 0, signifying "no specific

event or crossover occurred."

- **Historical Lag Features:** Imputed with -1 to explicitly distinguish "data absence" from a "zero-value change."
- **Base Price Data:** Strictly processed using **Forward Fill (FFill)**, ensuring that the 2025 test set predictions remain free of look-ahead bias.

3. Feature Optimization: PCA and K-Means Enhancement

To overcome the "Curse of Dimensionality" inherent in a 117-dimensional space, a two-stage optimization was employed:

- **Principal Component Analysis (PCA):** The feature space was projected onto **40 orthogonal principal components** (threshold determined via validation set). This reduces high collinearity among technical indicators while retaining maximum explanatory variance.
- **K-Means Market Regime Identification:** Unsupervised learning was performed in the PCA-projected space to identify distinct **Market Regimes**. These cluster labels were concatenated as auxiliary features to guide the model's state recognition.

4. KNN Model Configuration and Validation

- **Distance Metric: Euclidean Distance** was utilized, with global scaling performed via StandardScaler.
- **Hyper-parameter Tuning:** Through a comprehensive parameter sweep, the optimal number of neighbors was determined to be **k=11(neighbors), n=7 (clusters).**
- **Output Format:** The model outputs the **probability of upward movement**, which serves as a weighted input for the subsequent Stacking ensemble layer.

- **Random Forest**

"At the beginning, we used absolute prices (actual numbers) as training features. However, when the stock price rose to a very high level, the model always predicted the price would 'fall.' This is because the price

was higher than anything the model had seen before.

To fix this, we changed absolute prices to relative prices (such as percentage changes). After this adjustment, the model can now correctly predict 'up' or 'down' even when the stock price is high."

- **XGBoost**

1. **Feature engineering**

   - **Feature Selection:** After extensive testing, we choose the following market indicators as the feature used in this model: Close, Turnover, High, Low, Transaction, Movement and the external market indicator NASDAQ index performance for T+5 prediction. It is shown in the test that this particular combination of features yields the best performance for this prediction task.
   - **High-Dimensional Temporal Lag Features:** To extend the model with the ability to capture long-term momentum, we also implemented extensive lag processing. For each of the 7 features, historical values from the past 1 to 50 days were calculated to construct a high-dimensional feature space with significant temporal depth.
   - **Target Definition:** The model aims to predict the (Movement) N days into the future. Experiments were conducted on both T+1 and T+5.

2. **Training Strategy and Parameter Optimization**

   - **Weak Learner Constraints:** Set max_depth = 2. By restricting the model with shallow decision trees, the model is forced to learn only the most explanatory feature interactions, preventing it from over-fitting to specific period noise.
   - **Randomness Enhancement:** Set subsample = 0.8 and colsample_bytree = 0.8. By randomly selecting a portion of samples and features during each iteration, the model's robustness against unseen data is increased.
   - **Training Scale:** Set n_estimators = 100000 to ensure sufficient iteration space for the gradient descent process.

- **Evaluation Metric:** logloss is utilized to measure the performance of the classification model where the prediction is a probability value between 0 and 1.

- **Naive Bayes**

1. **Configurations**

   - Data set: analysis datasets of TSMC's stock prices from 2021 to 2024
   - Test set: analysis data of TSMC's stock prices on 2025
   - Output: predicted possibility of an upward movement in tomorrow given today's features.

2. **Feature selection**

   Naive Bayes models are highly sensitive to a slight difference of features, including noises. To make our model more precise and also more robust, we shall focus on features which provide the most information and eliminate those who are noisy or misleading. During testing, we found out that after doing selectKBest, there is a significant surge in accuracy in model performance.

3. **Hyper-parameter fine tuning**

   Moreover, to furthermore improve the performance, we apply fine tuning to the hyper-parameters of GaussanNB() and selectKBest, the result shows the best combination is

   - K = 28
   - Selected features = Turnover', 'Open', 'High', 'Low', 'Close', 'Change', 'Transaction', 'MA5', 'MA10', 'MA5_Capacity', 'BR5', 'BR10', 'High-Low', 'Open-Close', 'EMA5', 'EMA10', 'K', 'D', 'MACD', 'MACD_signal', 'MACD_hist', 'RSI14', 'STD20', 'ADX', 'RSI', 'DJI', 'NASDAQ', 'SOX', 'SPX', 'ADR', 'twclose'
   - Score function of selectKBest = mutual info classif

4. **Handling Class Imbalance**

However, when we dive into the output generated by this model, we notice that the recall of 1s (upward movement)of this model is very low, meaning the model had a hard time finding when the price will go upward. Combing the fact that the model still remains a decent accuracy, we believe that this is because the stock price of the dataset is usually going upward, making the result seem pretty well even if the model blindly predicts 1s.

A naive solution to this is by deleting some of the data with result equals 1 in training set. Yet this might lower the accuracy of the model.

5. **Boosting**

Another solution to the problem just mentioned is by boosting, We can train a model first, then test it. After that, we form a new dataset by collecting the data the model had mis predicted and use them to train a new model. The result is generated by making a poll to all the models. Since we are trying to predict 0s, in this model, we include all data with label equals 0.

- **<u>Neural Network</u>**

1. **Input and output Representation**

We believe that predicting future stock price movements depends not only on the previous day's price, but rather on the stock prices over a preceding period. We adopted a sliding window approach with a window size set to 30 days. This means that for each instance, the model receives a complete multi-dimensional feature sequence covering the past 30 trading days as input. The output layer is designed for single-point prediction, aiming to forecast the stock price movement (rise or fall) for the next day (T+1) or a specific future timeframe (e.g., T+5). We will choose the model with better performance when doing stacking.

2. **Data preprocessing**

In the data processing phase, we aggregated daily stock metrics

(such as opening and closing prices) from every consecutive 30-day period into a single row. This allows the 30-day historical data to serve as the input features for model training.

Furthermore, we ensure that the class distribution is balanced in both the training and validation sets. This prevents the model from developing a bias toward the majority class, ensuring that it learns to identify features for both upward and downward trends effectively.

3. **Model Architecture and handling overfitting**

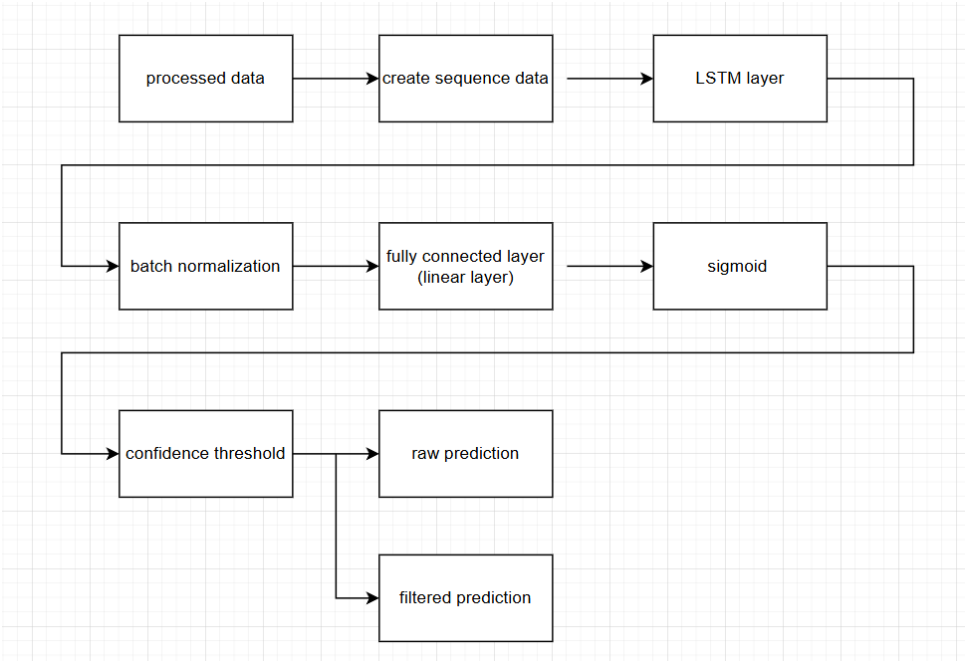
The neural network architecture consists of 6 layers, with each hidden layer containing 4,096 units and utilizing ReLU as the activation function. We employed nn. CrossEntropyLoss as the loss function.

However, we initially encountered severe overfitting during the training process. To address this, we incorporated Batch Normalization and Dropout into every hidden layer and applied Adam optimizer as L2 regularization.

4. **Feature selection**

During the training process, we recognized that using absolute values—such as opening and closing prices—as input features might cause the model to merely memorize specific price points rather than learning underlying patterns. For instance, the model might develop a bias to predict a decline solely because the price exceeds a certain fixed threshold. However, our goal is for the model to generalize effectively to unseen data rather than relying on rote memorization. Therefore, we switched to using relative features that are independent of absolute magnitude, such as the difference between opening and closing prices and MACD. This approach effectively resolves the issue.

- **LSTM**



To address the limitations of static models in handling time-series data, we implemented a Long Short-Term Memory (LSTM) network specifically designed to capture sequential patterns and long-term dependencies in stock price movements.

1. **Input Representation and Sequence Construction** Unlike traditional machine learning models that treat data points as independent observations, our LSTM model utilizes a **sliding window** approach to preserve temporal order.

- Feature Selection: We selected 9 key features as input variables.
- Sequence Length: The sequence length is set to 32, meaning the model processes a history of the past 32 trading days to predict the next day's price movement. This window size was chosen to allow the model to effectively learn both short-term fluctuations and medium-term market trends.

2. Model Architecture:

To prevent overfitting on noisy financial data and to improve convergence

stability, we designed the following architecture:

- LSTM Layer: Consists of one LSTM layer with a hidden dimension of 128. This layer is responsible for extracting temporal features from the input sequences and maintaining a memory of long-term dependencies.
- Batch Normalization (1D): A Batch Normalization layer is applied after the LSTM output. In the context of financial time series, this step is critical as it standardizes the distribution of hidden states, mitigates the vanishing gradient problem, and significantly accelerates training convergence.
- Output Layer: A final Fully Connected (FC) layer maps the 128-dimensional latent vector to a single scalar value, representing the unnormalized prediction score. During evaluation, a sigmoid activation function is applied to this value to map it into the range (0, 1), yielding the estimated probability of an upward price movement.

3. Training Strategy: Handling Imbalance and Regularization

- Class Imbalance Handling: To prevent the model from biasing towards the majority class, we calculated the imbalance ratio between positive (Up) and negative (Down) samples. We then incorporated a specific weight into the BCEWithLogitsLoss function to rebalance the loss contribution of each class.
- Regularization: We employed the Adam optimizer combined with Weight Decay (L2 Regularization). This constrains the magnitude of the model weights, reducing complexity, and preventing overfitting.
- Early Stopping: We monitored the F1 score and Accuracy on the validation set during training. We saved the model weights from the epoch with the best validation performance.

4. Confidence-Based Filtering

During the testing phase, we observed that a standard classification threshold of 0.5 was vulnerable to stochastic market noise. To enhance robustness, we introduced a Confidence Score mechanism:

$$\text{confidence} = |\ \text{probability} - 0.5\ | * 2$$

We applied a confidence threshold of 0.3. This implies that only

predictions with high certainty are considered valid trading signals. Predictions falling between these bounds are classified as neutral and filtered out.

Experimental results demonstrate that this filtering mechanism improves prediction accuracy from a baseline of 56% to 60% on high-confidence samples (consisting approximately 20% of the test set).

4.4 Stacking Strategy:

To further improve prediction stability and accuracy, we implemented a Stacking Ensemble technique. Instead of relying on a single model, Stacking combines the predictions from multiple "Base Learners" to generate a final, more robust decision.

Meta-Model Selection:

In this final stage, we employed a simple XGBoost Classifier as our Meta-Learner. Instead of using raw market data, this model takes the predicted probabilities from our base learners as its input features. This allows the model to learn which base learner is more reliable in different market situations.

Optimization and Constraints:

To combine these predictions effectively, we used Grid Search to find the best hyperparameters. A key part of our strategy was to strictly limit the max_depth to a low range (1 to 3). By keeping the tree depth shallow, we ensure that the Meta-Learner remains simple and generalizable, acting as a robust weighting mechanism rather than overfitting the training data.

4.5 Backtesting & Simulation Stategy

We set a maximum limit for buying, and the amount we buy depends on the confidence score. In our backtesting program, the maximum is set to 10 lots (which equals 2000 shares). For example, when the confidence score is 0.7, we buy 0.7 times the maximum lots (10 lots * 0.7). If the confidence is lower than 0.5, we also sell the stock based on this linear ratio. This means the strategy holds different amounts of stock at different times, but we can calculate the average number of lots held by the model.

To check how well the model performs, we compare it with a "Buy and Hold" strategy. To be fair, the Buy and Hold strategy is set to buy the same amount as the model's average position.

In this backtesting program, we did not set an initial capital amount. This is because the main point is to check the relative return rate, not the absolute amount of money. This ensures that limited funds do not become a bottleneck for trading, so we can see the true performance of the strategy.

We also did not include transaction fees. This is because taxes and fees in the futures market are very low. For a stock like 2330, these costs are small enough to ignore. This is also why we use "lots" as the trading unit instead of standard shares.

5. Experiments & Results

5.1 Base Models Performance

- **KNN**：After applying the **Cluster-Augmented KNN** model (configured with Clusters=7 and k=11) to the independent 2025 test dataset, the performance metrics are summarized as follows:

```
===== Final Cluster-KNN Test Evaluation =====
Accuracy     : 0.6390
Recall Up    : 0.5455
Recall Down  : 0.7738
Macro F1     : 0.6390

Classification Report:
              precision    recall  f1-score   support

           0     0.5417    0.7738    0.6373        84
           1     0.7765    0.5455    0.6408       121

    accuracy                         0.6390       205
   macro avg     0.6591    0.6596    0.6390       205
weighted avg     0.6803    0.6390    0.6393       205

Confusion Matrix:
[[65 19]
 [55 66]]
```

2025 test	basic	After PCA	After clustering

result	KNN		
accuracy	45.85%	58.54%	63.9%
macro-f1	45.67%	54.65%	63.9%

- **Discussion & Interpretation:** The experimental results indicate a transformative improvement in performance, with accuracy escalating from **45.85% to 63.90%**. This 18% surge is attributed to two key factors:
  - **Noise Mitigation via PCA:** Reducing the 117-feature space helped eliminate redundant signals and multicollinearity, focusing the KNN algorithm on the most impactful latent variables.
  - **Regime Awareness via Clustering:** The K-Means labels provided the model with the ability to distinguish between different market phases. This effectively addressed the **non-stationarity** of TSMC's stock data, allowing the KNN classifier to identify historical "neighbors" that are structurally similar to the 2025 AI-driven market regime.

  These findings suggest that **market environment recognition** is a critical prerequisite for successful nearest-neighbor classification in volatile financial markets.

- **Naive Bayes:** After all the fine-tuning and boosting being done, the final performance is show as follows:

- **Before optimizing**

```
Raw Model Evaluation:
Accuracy: 0.5394021739130435
Precision: 0.5400357772200566
F1 Score: 0.4707967192043625
Recall: 0.522204452736871

Classification Report:

              precision    recall  f1-score   support

           0       0.54      0.86      0.66       387
           1       0.54      0.19      0.28       349

    accuracy                           0.54       736
   macro avg       0.54      0.52      0.47       736
weighted avg       0.54      0.54      0.48       736
```

- **After optimizing**

```
Final Model Evaluation:
Accuracy: 0.5560975609756098
Precision: 0.5529100529100529
F1 Score: 0.5403021118257313
Recall: 0.5477281405116456

Classification Report:

              precision    recall  f1-score   support

           0       0.54      0.39      0.46        97
           1       0.56      0.70      0.63       108

    accuracy                           0.56       205
   macro avg       0.55      0.55      0.54       205
weighted avg       0.55      0.56      0.54       205
```

- **Discussion & Interpretation:** The result proves a noticeable improvement regarding recall, with an increase from 0.19 to 0.70. This serves as evidence of the problem mentioned in 4.3, and our solution to that problem, to some extent, does work.

- **XGBoost:** the performance after applying the feature engineering and the training strategy mention above is show as follows:
  - T+1

```
Classification Report:
              precision    recall  f1-score   support

         0.0       0.49      0.42      0.45        84
...
    accuracy                           0.59       205
   macro avg       0.56      0.56      0.56       205
weighted avg       0.58      0.59      0.58       205
```

- T+5

```
Classification Report:
              precision    recall  f1-score   support

         0.0       0.53      0.39      0.45        83
...
    accuracy                           0.61       201
   macro avg       0.59      0.57      0.57       201
weighted avg       0.59      0.61      0.59       201
```

- **Discussion & Interpretation: an interesting phenomenon is that this model has slightly better performance on T+5 compared to T+1. This suggests that XGBoost might be better for predicting medium term market movement instead of short-term result.**

- **NN: T**he performance after applying the feature selection and the overfit handling strategy mention above is show as follows:

```
------------------------------
Test Set Evaluation Results
------------------------------
Accuracy: 0.5561
F1-Score: 0.4485
------------------------------
Confusion Matrix:
[[77 29]
 [62 37]]
------------------------------

Detailed Classification Report:
              precision    recall  f1-score   support

           0       0.55      0.73      0.63       106
           1       0.56      0.37      0.45        99

    accuracy                           0.56       205
   macro avg       0.56      0.55      0.54       205
weighted avg       0.56      0.56      0.54       205
```

T+1

```
------------------------------
Test Set Evaluation Results
------------------------------
Accuracy: 0.5220
F1-Score: 0.3000
------------------------------
Confusion Matrix:
[[86 20]
 [78 21]]
------------------------------

Detailed Classification Report:
              precision    recall  f1-score   support

           0       0.52      0.81      0.64       106
           1       0.51      0.21      0.30        99

    accuracy                           0.52       205
   macro avg       0.52      0.51      0.47       205
weighted avg       0.52      0.52      0.47       205
```

T+5

Discussion & Interpretation: It is evident that despite extensive tuning, the performance for T+1 is better than predictions of T+5. We hypothesize that due to the inherent noise in stock market data, T+5 predictions are susceptible to the noise accumulated during the T+1 to T+5 interval, which negatively impacts performance.

- **LSTM:**

```
Accuracy : 0.6262
Precision: 0.6275
Recall   : 0.6273
F1-score : 0.6262
Classification Report:
              precision    recall  f1-score   support

           0       0.65      0.60      0.62       107
           1       0.60      0.66      0.63        99

    accuracy                           0.63       206
   macro avg       0.63      0.63      0.63       206
weighted avg       0.63      0.63      0.63       206
```

Discussion & Interpretation: The LSTM model achieves the best performance, reaching an accuracy of up to **62%**, and its F1-score further indicates that the model can effectively predict both upward and downward price movements

with good accuracy.

- **Random Forest**
  - **Before h**yperparameter tuning

```
--- 基本決策樹模型 (未調參) 評估 ---
測試集 Accuracy: 0.5809
測試集 F1-Score (Weighted): 0.5801
分類報告 (0=跌/平, 1=漲):
              precision    recall  f1-score   support

           0     0.5718    0.6252    0.5973       707
           1     0.5917    0.5371    0.5630       715

    accuracy                         0.5809      1422
   macro avg     0.5817    0.5811    0.5802      1422
weighted avg     0.5818    0.5809    0.5801      1422
```

  - **After h**yperparameter tuning

```
--- 最佳決策樹模型 (GridSearchCV 調參後) 評估 ---
測試集 Accuracy: 0.6217
測試集 F1-Score (Weighted): 0.6217
分類報告 (0=跌/平, 1=漲):
              precision    recall  f1-score   support

           0     0.6199    0.6181    0.6190       707
           1     0.6234    0.6252    0.6243       715

    accuracy                         0.6217      1422
   macro avg     0.6216    0.6216    0.6216      1422
weighted avg     0.6217    0.6217    0.6217      1422
```

5.2 Stacking Ensemble Results:

Here, we select the three models as base learners. The results show that the ensemble achieves higher accuracy on both the training and test sets compared to any single model. This improvement highlights the effectiveness of ensemble learning. Because individual models tend to make different types of errors, integrating their predictions enables the Meta-Learner to compensate for these weaknesses, resulting in a more robust and accurate prediction system.

```
Base Models Accuracy:
Model      | Train Acc    | Test Acc
----------------------------------------
KNN        | 0.5572       | 0.6505
LSTM       | 0.5736       | 0.5728
XG         | 0.5005       | 0.5631
----------------------------------------
Fitting 3 folds for each of 27 candidates, totalling 81 fits
Best Parameters: {'learning_rate': 0.01, 'max_depth': 3, 'n_estimators': 100}
Best CV Accuracy: 0.6179082419192499
Stacking Accuracy on Test Set: 0.6553
```

6. Discussion & Analysis

6.1 Feature Importance: Diagnosing Data Leakage through Visualization

In financial forecasting, Feature Importance is not merely a measure of predictive power but a critical diagnostic tool for ensuring model integrity. By cross-referencing **Heatmap-based Correlation Matrices** with **Scatter Plots**, we identified and rectified a significant instance of **Data Leakage**.

1. The Detection: High Correlation Heatmap & Linear Scatter Plot

Initially, our model achieved a suspiciously high accuracy of approximately 80%[1]. We used two visualization techniques to investigate this "too-good-to-be-true" result:

- **Heatmap Observation**: As shown in **Figure 1(Initial Observation)**, the Correlation Matrix (Heatmap) revealed an exceptionally high correlation of **0.46** between the target variable (movement) and the same-day TSMC ADR price.
- **Scatter Plot Observation**: The corresponding scatter plot of "TSMC vs ADR" showed data points tightly clustered along a **positive linear trend**.
- **Diagnostic Insight**: This synchronized linear relationship indicated that the model was using the U.S. market's closing price to "predict" the Taiwan market's movement for the same day. Since these markets have overlapped influences, this constituted **Data Leakage**, as the model was essentially "reading the answer" rather than forecasting.

2. The Correction: Correlation Drop & The "Cloud" Distribution

To ensure the model only learned from valid historical leads, we realigned the

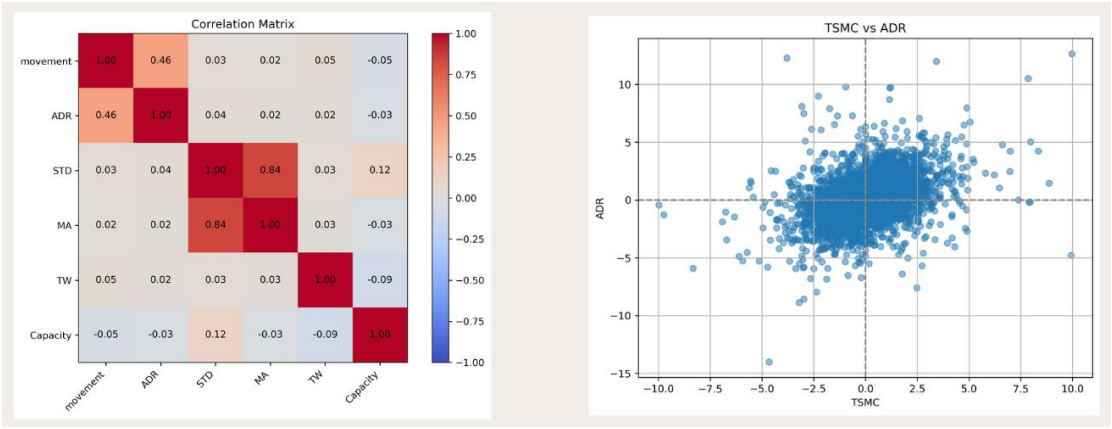
temporal sequence: **U.S. Market (T-1) + Taiwan Market (T) → Prediction (T+1)**. We then re-verified the data through visualization:

- **Heatmap Shift**: As seen in **Figure 5 (Corrected Observation)**, the correlation between movement and ADR dropped drastically from **0.46 to -0.03**.
- **Scatter Plot Shift**: The previously linear trend in the scatter plot collapsed into a **randomly distributed "cloud" of data points**.
- **Scientific Significance**: While a "cloud" might seem like a failure in other domains, in financial forecasting, it proves that the synchronous bias (leakage) has been removed. The model is now forced to identify subtle, non-linear patterns rather than relying on a direct linear "cheat".

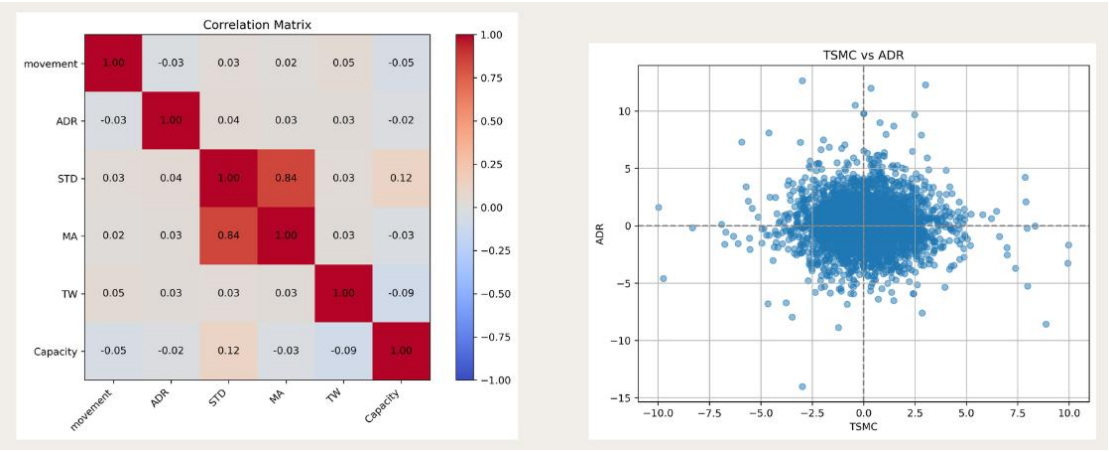
3. Conclusion on Visualization

This rigorous diagnostic process using both Heatmaps and Scatter Plots allowed us to transition from an overfitted 80% accuracy to a **robust Stacking accuracy of 65.53%** on the 2025 independent test set. This underscores the necessity of visual diagnostics in preventing the catastrophic failure of trading models in real-world deployment.

- Figure 1



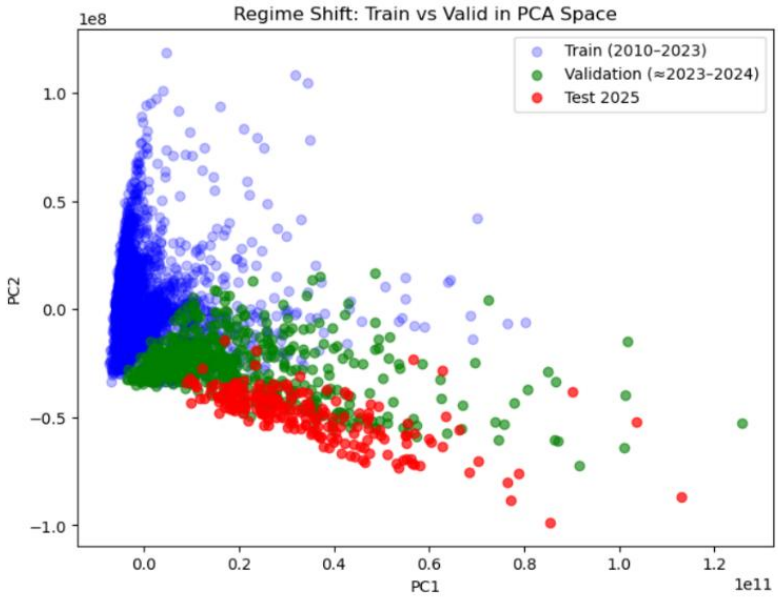
- Figure 2



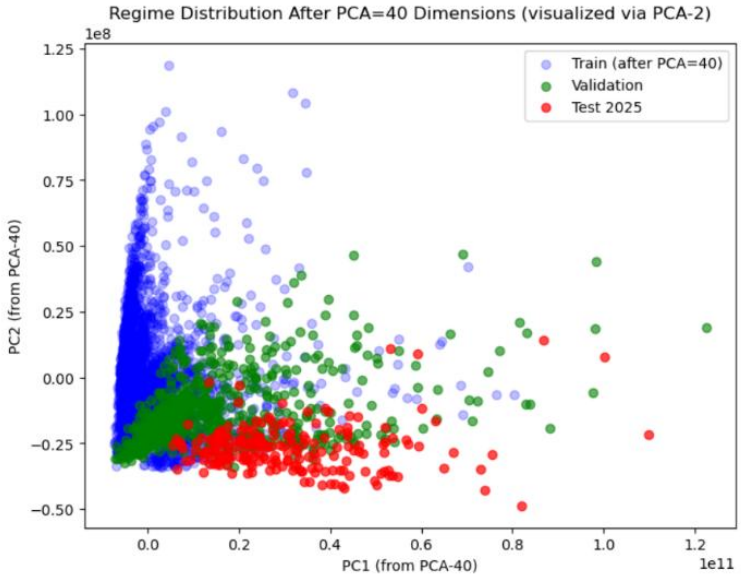
6.2 Distribution Shift Analysis:

Experimental Analysis and Visualization by KNN

- The initial performance of the standard KNN model on the test set was suboptimal. Through feature distribution analysis and dimensionality reduction visualization, we identified a significant **distribution shift** between the 2025 test data and the historical training data.
- To mitigate this, I first applied **PCA** to standardize and reduce the dimensionality of high-dimensional technical indicators, effectively converging the feature spaces of different periods. Subsequently, **K-Means clustering** was utilized for market regime identification, enabling the model to locate historical trading segments that share structural similarities with the current environment.
    - **Figure A: Baseline Feature Space** In the unoptimized feature space, the 2025 data points (red cloud) exhibit a pronounced offset from historical data. This explains the failure of the standard KNN: due to the **distribution shift**, the algorithm is forced to identify "neighbors" that are statistically dissimilar to the current market state.

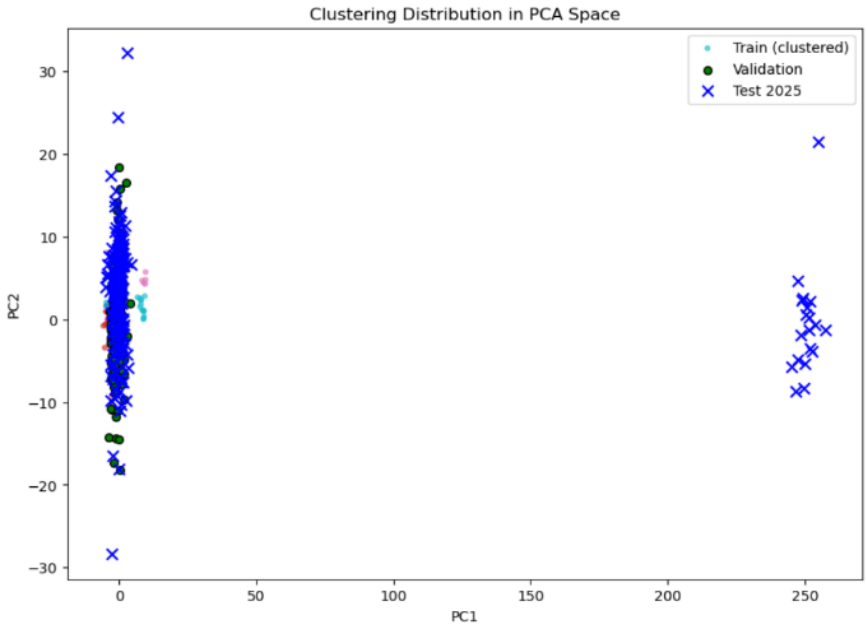


- **Figure B: PCA-Transformed Space** After PCA-based dimensionality reduction and denoising, the gap between the two datasets is significantly narrowed. PCA successfully filtered out the idiosyncratic noise unique to 2025 while extracting cross-temporal commonalities. Furthermore, the impact of the distribution shift was attenuated during the projection process, providing a preliminary solution to noise and variance issues.



- **Figure C: Cluster-Augmented Space** With the integration of **K-Means labels**, the data points align along a trend axis while distinct color blocks partition the space into various **market regimes** (e.g., strong

bullish, consolidation, or oversold). This allows the KNN model to search for neighbors strictly within the corresponding market regime. I believe this mechanism is the core factor that successfully elevated the predictive accuracy to **over 60%**.



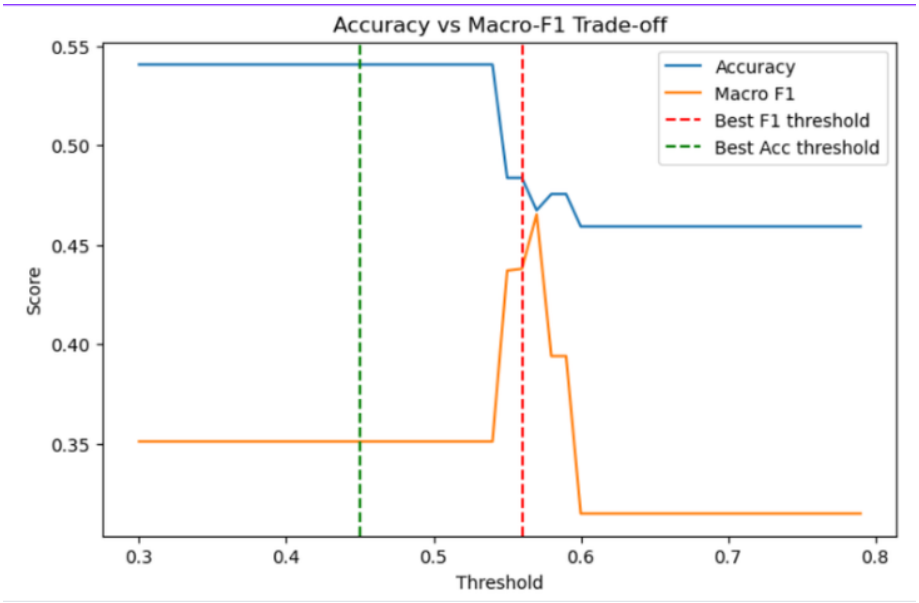
6.3 F1-score vs. Accuracy Trade-off:

In typical machine learning scenarios, **Accuracy** and **Macro-F1 score** often complement each other, moving in the same direction. However, as illustrated in the **"Accuracy vs Macro-F1 Trade-off"** plot, financial markets present a unique challenge where these two metrics frequently diverge due to **Class Imbalance**.

1. The Challenge of Class Imbalance in TSMC Data

Our dataset for TSMC exhibits a natural "long bias," where the number of upward-movement days significantly outweighs downward days.

- **Accuracy Bias**: Because the market trends upward more often, a model can achieve a baseline accuracy simply by "blindly" predicting an upward trend (1s).
- **The Macro-F1 Conflict**: While such a biased model has high accuracy, its **Recall** for downward movements (0s) will be extremely low, resulting in a poor Macro-F1 score that fails to reflect true predictive reliability.



2. Visual Analysis of the Trade-off Curve

As shown in the provided visualization above, we analyzed the model's performance across various decision thresholds:

- **The Accuracy Plateau**: The green dashed line indicates the **Best Accuracy Threshold**, where the model maximizes its total correct predictions but likely favors the majority "Up" class.
- **The Macro-F1 Peak**: The red dashed line represents the **Best F1 Threshold**. At this point, although the overall Accuracy drops, the Macro-F1 score reaches its maximum, indicating a better balance between identifying both "Up" and "Down" movements.
- **The Sharp Decline**: Beyond a threshold of 0.6, both metrics drop significantly as the criteria for a "valid signal" become too restrictive for the model to maintain abstract patterns.

3. Strategic Importance in Trading Decisions

This trade-off is critical for real-world trading strategy:

- **Reducing False Positives**: In trading, a "False Positive" (predicting a rise before a crash) is far more costly than a "False Negative" (missing a small gain).

- **Prioritizing Precision**: By choosing a threshold closer to the **Best F1** rather than the Best Accuracy, we intentionally sacrifice some correct predictions to ensure that when the model *does* signal a trade, the probability of it being a "fatal error" is minimized.

6.4 Model Complexity vs. Generalization - Simpler is Better

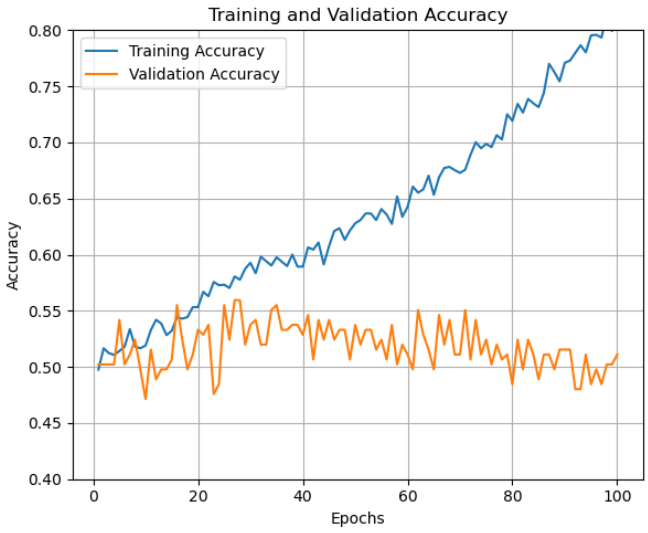
experiment analyzed by LSTM

At the beginning of this study, given the inherent complexity of financial markets, we hypothesized that a deep and complex LSTM architecture would be required. Intuitively, we believed that a high-capacity model was necessary to capture the complex causal relationships and long-term dependencies in stock price movements.

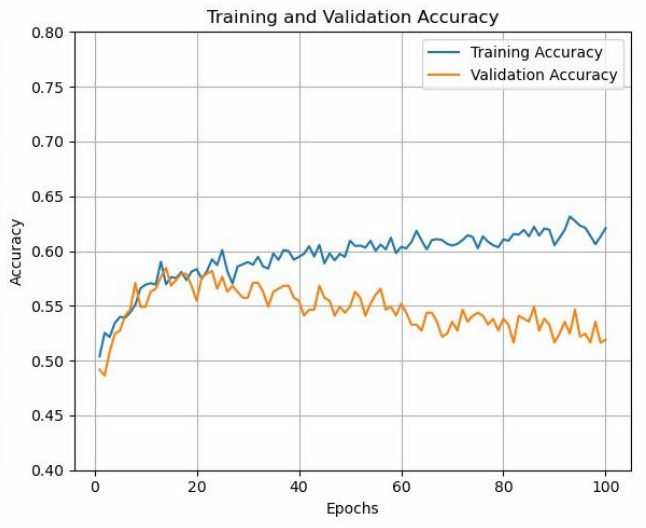
However, our experimental results revealed a counter-intuitive finding: simpler architectures (1 to 2 LSTM layers) significantly outperformed deeper networks. We observed that as the model complexity increased, the training accuracy improved, but the testing performance drops rapidly. This indicates that complex models were suffering from severe overfitting, failing to generalize unseen data.

As shown in the figure below, compared with deeper architectures, the model with one LSTM layer is less prone to overfitting and therefore attains higher accuracy.

model with 4 LSTM layer



Model with 1 LSTM layer



We attribute this phenomenon to the data limitations mentioned in our Introduction. Stock prices are driven by a multitude of latent factors such as economic policy, breaking news, and investor sentiment, which are not present in our input dataset (which consists only of historical prices and technical indicators).

A complex model attempts to find a "perfect rule" to explain every price of movement in the training set. Since the input features alone are insufficient to explain all price fluctuations, a high-capacity model inevitably starts "memorizing" the stochastic noise and random walks in the training data, rather than learning true market trends.

In contrast, a simpler architecture acts as a form of regularization. By limiting the model's capacity, we force it to focus only on the most robust and recurring patterns. The model literally "does not have enough memory" to memorize the noise, so it is compelled to learn the general rules of price movement.

This finding highlights an insight: when the input data contains high noise and incomplete information (missing external variables), architectural simplicity is crucial for robustness. A constrained model prevents the overfitting of noise and leads to better generalization on future data.

6.5 The Limitations of Binary Classification and accuracy

experiment analyzed by stacking prediction

The Disconnect Between Accuracy and Profit:

In our trading simulation (Backtesting), we observed a counter-intuitive phenomenon: a model with lower accuracy (57%) generated higher cumulative returns than a model with higher accuracy (67%). As shown in the comparison figures, the "less accurate" model achieved a better final profit. We identified three primary reasons for this discrepancy:

Movement and Magnitude:

The binary classification model only predicts the probability of an upward movement, not the magnitude of the price change. Thus, models might have high accuracy by correctly predicting many days with small price increases. However, if it makes a single wrong prediction on a day with a massive price drop, the accumulated profits can be wiped out instantly. What's more, our trading strategy adjusts position sizes based on confidence (probability). If the high-accuracy model is "overconfident" on the wrong days, it incurs heavy losses. Conversely, the lower-accuracy model might have been "wrong" often, but conservative in its trade weighting, while catching a few crucial, large-magnitude trends correctly.

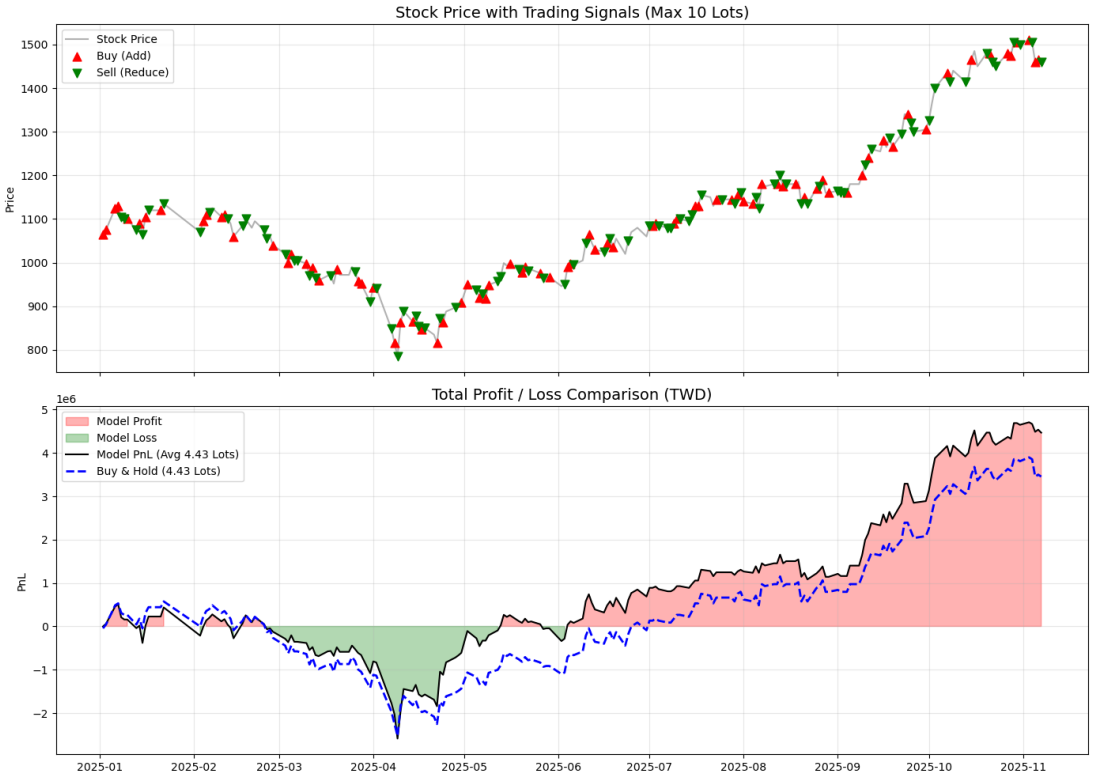
Overfitting to Normal Market Patterns:

A higher accuracy score often implies that the model has learned the "normal" daily fluctuations of the stock market very well. However, stock prices, especially the TSMC, are influenced by unpredictable external factors such as geopolitical news. A model with very high accuracy may have overfitted to stable, historical patterns. When an unexpected "black swan" event occurs, these "smart" models often fail to adapt and make confident but incorrect predictions, leading to significant financial loss. In contrast, a "dumber" model (lower accuracy) might be less sensitive to these specific historical rules, making it ironically more robust—or simply lucky enough to avoid betting heavily against a market crash.

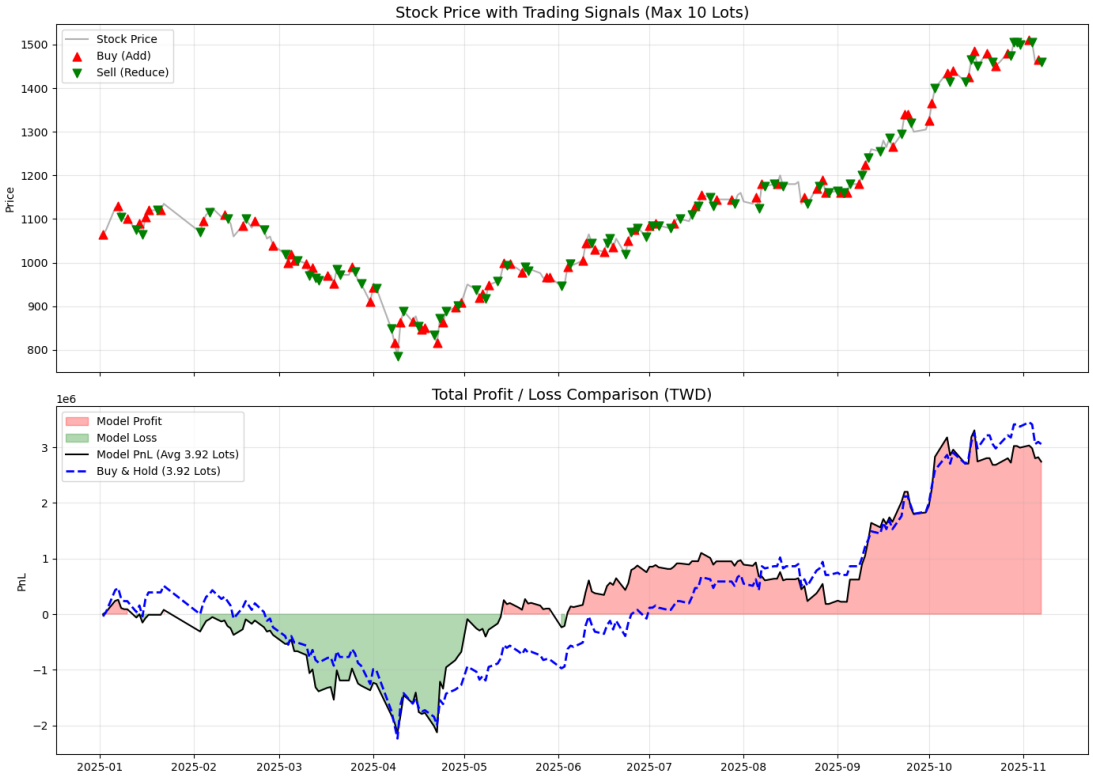
Asymmetry of Prediction Errors:

In binary classification metrics like Accuracy or F1-score, every error is treated equally—a False Positive is mathematically the same as a False Negative. While in trading, errors have asymmetric costs. Missing a buying opportunity (False Negative) only results in an opportunity cost (zero profit). However, buying before a price crash (False Positive) results in actual loss. The high-accuracy model might have minimized total errors but committed more "fatal" errors (buying before a drop). The lower-accuracy model might have made more mistakes overall, but those mistakes were mostly harmless, allowing it to earn better profit.

Ensemble model with accuracy = 57%



Ensemble model with accuracy = 67%



7. Conclusion & Future Work

Throughout our experiments, our models achieved an accuracy range of 55% to 60%, with our best-performing configurations (after extensive fine-tuning and stacking) exceeding the 60% threshold. While these numbers might appear low compared to other machine learning domains, in the context of financial time-series forecasting, they are statistically significant and highly acceptable, given the noisy, non-stationary, and highly stochastic nature of stock market data.

Moving forward, we aim to build upon these foundations through several key initiatives:

- **Advanced Feature Engineering & Multi-Market Validation**: We plan to refine our feature engineering process to capture deeper market insights. Furthermore, we intend to test the robustness of our models across diverse asset classes and markets, such as the U.S. stock market or different local sectors like China Steel (CSC), to ensure

generalizability.

- **Architectural Evolution & Ensemble Methods**: We will continue to explore diverse machine learning architectures and alternative ensemble techniques to further mitigate the "lagging" and distribution shift issues identified during this study.
- **Sentiment Analysis via Natural Language Processing (NLP)**: To complement our quantitative technical indicators, we aim to integrate NLP models to analyze news headlines and social media sentiment. This will allow the model to factor in market psychology and react more effectively to news-driven volatility.
- **Diagnostic Refinement**: We will focus on identifying and analyzing deeper underlying patterns and anomalies within our data to continuously improve our model's predictive precision and decision-making logic.

8. Author Contributions

曾煜展（30%）：report, data collection, preprocessing, LSTM model, stacking ensemble.

郭宣汝（20%）：report, research work, KNN model, GitHub repository management.

謝尚錡（16%）：report , implementation of Random Forest, backtesting&simulation

郭子維（12%）：report, implementation of XGBoost model.

高予謙（11%）：report, implementation of NN model.

盛樺（11%）：report, implementation of Naïve Bayes.