FPT UNIVERSITY

MINISTRY OF EDUCATION AND TRAINING

FPT UNIVERSITY

Capstone Project Document

Using AI for Intelligent Stock Trading

GSP25AI12	
Group Members	<Nguyễn Việt Anh> <Leader> <SE170371> <Hoàng Văn Bắc> <Member> <SE150427> <Hoàng Xuân Cảnh> <Member> <*SE170064*>
Supervisor	Trần Trọng Huỳnh
Capstone Project code	*SP25AI37*

- Hồ Chí Minh, 4-2025 -

Table of Contents

Acknowledgment

We would like to extend our deepest gratitude to our academic supervisor, Mr. Trần Trọng Huỳnh, for his invaluable guidance, constructive feedback, and unwavering support throughout the course of this project.

We are also thankful to the faculty of FPT University for providing us with a solid academic foundation.

Our heartfelt thanks also go to our families and friends for their encouragement and continuous support, which have been instrumental in our success.

Definition and Abbreviations

Abbreviation	Definition
AI	Artificial Intelligence
DL	Deep Learning
ML	Machine Learning
PMP	Project Manager Plan
WBS	Work Breakdown Structure
ATR	Average True Range
BB	This is a stock bubble prediction problem.
UD	This is a stock up or down prediction problem.
LR	Logistic Regression – A logistic regression model used for binary classification.
SVM	Support Vector Machine – A Model used for binary classification.
LSTM	Long Short-Term Memory model.
TFM	Transformer model.
HL	High-Low
LO	Low-Open
SR	Support Resistance
PM	Price Momentum
MDD	Maximum Drawdown
TVV	Trading Volume Volatility
SK	Skewness
MA7_t, MA14_t, MA21_t	7-day, 14-day, 21-day Moving Averages
SMOTE	Synthetic Minority Over-sampling Technique

CAPM	Capital Asset Pricing Model
Fama-French 3 Factors	Fama-French 3-Factor Model
Carhart 4 Factors	Carhart 4-Factor Model
RMW	Robust Minus Weak Factor
CMA	Conservative Minus Aggressive Facto
MOM	Momentum Factor
HML	High Minus Low Factor
SMB	Small Minus Big Factor
ADF	Augmented Dickey-Fuller Test
API	Application Programming Interface
EC2	Elastic Compute Cloud
ECR	Elastic Container Registry
AWS	Amazon Web Services
ROC	Receiver Operating Characteristic Curve
CSV	Comma-Separated Values
F1-Score	Harmonic Mean of Precision and Recall
MAE	Mean Absolute Error
R-squared	Coefficient of Determination
RMSE	Root Mean Square Error
Alpha	Measure of Portfolio’s Excess Return
Beta	Measure of Systematic Risk Relative to Market

List of Tables

List of Figures

Figure IV.B.1.2.4.4: Transformer Model.
Figure IV.B.2.2: Optimizer.
Figure IV.B.2.3: Trainer.
Figure IV.B.3.2.g: Average score of all models by BB, UD.
Figure IV.B.4.1: Push data to mongo DB.
Figure IV.B.4.2: Deploy: to AWS.
Figure IV.B.4.2.1: Get mongo data.
Figure IV.B.4.2.2: API. for BB, UD.
Figure IV.B.4.2.3: Form Github push ECR and run EC2.
Figure V.1 – Detailed flow of the Web Application integrating Frontend and Backend operations
Figure VI. Top Tickets by BB, UD Train, Valid, Test dataset

List of Resources

Capstone Resources

Input resources	Output resources
Using AI for Intelligent stock trading	Propose solution
Using AI for Intelligent stock trading	Propose solution
Using AI for Intelligent stock trading	Propose solution

data collection requirements	Data collection
data collection requirements	Data collection
data collection requirements	Data collection
Propose solution BB	Code BB
	Feature engineering BB
Propose solution UD	Code UD
	Feature engineering UD
Propose solution MF	
	On Excel MF create feature engineering MF
Feature engineering MF	Code MF
Feature engineering BB	Code BB
	Metrix BB
Feature engineering UD	Code UD
	Metrix UD
	API BB_UD
API BB_UD	
Code MF	Web

Reference:*Project Management.xlsx*

I. Project Introduction

Making wise investing decisions in the face of an increasingly erratic and volatile stock market necessitates not just specific expertise but also the capacity for rapid and precise data analysis. Our capstone project's objective is to use contemporary artificial intelligence (AI) techniques to help investors make more informed stock trading decisions.

The project's main goal is to create an intelligent stock trading support system that can process financial data and offer recommendations for investments in real time. In particular, we tackle three primary issues:

1. Monthly Bubble Price Prediction – identifying periods when stock prices are likely inflated relative to their intrinsic value.
2. Weekly Stock Trend Prediction – supporting short-term investment decisions by forecasting upward or downward trends.
3. Portfolio Risk Assessment – combining profit forecasting and risk analysis to recommend whether to buy, sell, or hold stocks.

The system is anticipated to improve users' decision-making skills by combining several prediction models and utilizing AI in large data analysis, particularly during times of abrupt market swings.

1. Overview

1.1 Project Information

- Project Title: Using AI for Intelligent Stock Trading
- Project Code: SP25AI37
- Group Code: GSP25AI12

Project Team Members:

Table I.1.1.Project Team Members.

Full Name	Student ID	Phone Number	Email	Role
Nguyễn Việt Anh	SE170371	0368540482	anhnvse170371@fpt.edu.vn	Team Leader
Hoàng Xuân Cảnh	SE170064	0931486527	canhhxse170064@fpt.edu.vn	Member
Hoàng Văn Bắc	SE150427	0971932296	**bachvse150427**@fpt.edu.vn	Member

Academic Supervisor:

- Name**:** Trần Trọng Huỳnh
- Email: huynhtt@fpt.edu.vn
- Role**:** Supervisor 1

The supervisor is responsible for providing technical guidance, offering feedback on algorithms, and evaluating the project's progress at each stage.

1.2 Project Overview

The goal of the project "Using AI for Intelligent Stock Trading" is to develop an intelligent stock trading support system that uses cutting-edge artificial intelligence (AI) approaches to solve the challenge of making buy-sell decisions under extremely complicated and turbulent market conditions. The use of AI in analysis, forecasting, and investment decision-making is becoming a trend that cannot be avoided as financial data becomes more plentiful and the demand for quick responses increases.

The six primary chapters that make up this report's structure represent the project's research and implementation process:

- Chapter 1: Introduction – Provides an overview of the context, objectives, research problems, significance, and scope of the project.

- Chapter 2: Theoretical Foundations and Related Technologies – Analyzes related machine learning models and algorithms such as Logistic Regression, SVM, LSTM, Transformer, etc., and introduces concepts related to Bubble Price prediction, portfolio risk analysis, and financial indicators.
- Chapter 3: System Analysis – Clarifies system requirements, data flow, architectural design, and analyzes the functions of each system component.
- Chapter 4: System Design and Implementation – Details the process of building AI models, processing input data, training and evaluating the models, and integrating the predictive components into the stock trading recommendation system.
- Chapter 5: Experimentation and Evaluation – Analyzes experimental results, compares model performance, evaluates the accuracy and reliability of predictions, as well as their effectiveness in supporting investment decisions.
- Chapter 6: Conclusion and Future Development – Summarizes the project outcomes, evaluates its contributions, and proposes future research and development directions.

Our team's goal with this research is to provide a very useful solution that will help modernize stock trading and increase investors', students', and financial institutions' access to AI applications in finance.

2. Project Background

Both the number of individual investors and the amount of trade data have significantly increased in the Vietnamese and international stock markets in recent years. However, the market's continuous real-time changes, which are impacted by a variety of factors like crowd psychology, political and economic news, money flows, and more, make it very difficult to make timely investing judgments.

Despite the abundance of technical and fundamental research tools, most individual investors still find it difficult to sort through vast amounts of data and select the best course of action. In order to help investors anticipate market movements, evaluate risks, and maximize their buy-sell decisions, our team chose to focus on using artificial intelligence to create an intelligent stock trading support system.

The project focuses on addressing three main problems:

- Bubble Price Prediction – helping to detect price bubble conditions and alert investors.
- Weekly Up/Down Trend Forecasting – providing short-term trading signals.
- Portfolio Risk Analysis and Profit Prediction – supporting comprehensive buy-hold-sell decision-making.

3. Project Objective

By integrating several machine learning models, the project seeks to create an intelligent stock trading support system driven by AI that improves prediction efficacy and accuracy. Three primary functions are intended to be implemented by the system:

- Monthly Bubble Price Prediction – identifying stocks showing signs of price bubbles to minimize investment risks.
- Weekly Stock Price Trend Prediction – assisting investors in making short-term trading decisions.
- Portfolio Risk and Profitability Assessment – using quantitative models to evaluate investment effectiveness before final decision-making.

Four primary machine learning models LSTM, Transformer, SVM, and Logistic Regression are integrated into the system during installation and used for activities involving trend forecasting and bubble price detection. Four multi-factor financial models the extended CAPM, the Carhart 4 Factors, the Liquidity-adjusted 3 Factors, and the Fama-French 3 Factors are used concurrently for risk analysis and performance verification. The performance and degree of risk of each stock as well as the portfolio as a whole are evaluated using these models.

Notably, important measures like these are used to measure how accurate the investment signals from the multi-factor models are:

- Alpha (measuring portfolio performance)
- Beta (measuring systematic risk)
- Total Risk
- Idiosyncratic Risk

A thorough and rigorous assessment of the risk and profitability of the system's suggested investments is made possible by these indicators.

Furthermore, the system is built with an intuitive user interface, quick reaction times, and adaptable integration capabilities for future web-based or real-world application deployment. Standard classification criteria like Accuracy, Precision, Recall, and F1-Score are used to assess the AI models' efficacy, guaranteeing objectivity and real-world application.

4. Problem Statement

Making decisions about investments in the stock market is a difficult process that calls on investors to handle abrupt market swings and process vast amounts of data. Today, nevertheless, a lot of individual investors still mostly rely on gut feeling or community information, which results in decisions devoid of scientific support and raises the possibility of making poor capital allocation choices, losing money, or purchasing during peak times.

In fact, the market often experiences "bubble prices" – situations in which stock prices are too high relative to their actual value. Furthermore, many investors struggle with asset management as a result of their incapacity to evaluate the overall risk of a portfolio.

These problems are intended to be addressed by this effort by:

- Applying AI to more accurately forecast market trends.
- Automatically analyzing portfolio risk and performance.
- Providing clear and understandable recommendations to users, helping them make investment decisions with greater confidence.

5. Significance of the Project

Both theoretically and practically, the initiative offers substantial value:

- Academic Value: By combining the fields of artificial intelligence and finance, the project gives students a better grasp of how machine learning algorithms may be used to solve practical issues, especially in time series data processing, financial data analysis, and optimal decision-making.

- Practical Value: Individual investors may find the suggested system to be a helpful tool that reduces risk, expedites analysis, and boosts investing efficiency.
- Social Value: By lowering reliance on hearsay and unreliable information sources, the project supports the trend of wise investing in Vietnam. Additionally, it draws attention to how crucial technology is to the financial industry.

6. Project Scope & Limitations

Project Scope:

- Applying machine learning models (Logistic Regression, SVM, LSTM, Transformer, etc.) to predict price trends and assess investment risks.
- Focusing on stock market data from selected stocks during the period 2013–2025.
- Predicting key factors including: monthly Bubble Price, weekly price trends, portfolio risks, and expected returns.
- Providing investment recommendations (buy – sell) in a visual and easy-to-understand format.

Project Limitations:

- The project analyzes only historical data and cannot guarantee absolute future accuracy.
- The system does not yet integrate real-time trading or automated trading functionalities.
- It is limited to a selection of representative stocks rather than covering the entire Vietnamese stock market.
- It does not analyze investor psychology, macroeconomic news, or global fluctuations, even though these factors may significantly influence the market.

II. Project Management Plan

1. Teamwork

1.1 Group structure and roles

The group consists of 3 members, each assigned specific tasks to ensure the project's progress and efficiency.

Table II.1.1: Group structure and roles.

Full name	*student ID*	*Role*	*Primary responsibilities*
Nguyễn Việt Anh	*SE170371*	*Team Leader*	*Research Methodology, data processing, main report writer*
Hoàng Xuân Cảnh	*SE170064*	*Member*	*Data collector, training model, Website developer, web report writer + main report supporter*
Hoàng Văn Bắc	*SE150427*	*Member*	*Data collector, evaluate model, report's evaluation part writer, main report supporter*

1.2 Communication plan at work

- The team uses the following main communication channels: Zalo group, Google Meet, and Gmail. These tools facilitate daily work discussions, document sharing, and quick notifications of any arising issues.
- Meetings with the instructor are usually held once a week, typically at the weekend, after the team has updated its work progress. The purpose of the meetings is to provide updates, receive feedback from the instructor, and adjust the direction if necessary.

2. Project management methods

2.1 WBS: Work Breakdown Structure

Each content item represents a group of tasks (divided into smaller subtasks). The tasks include:

- Ideation and task assignment
- Data collection
- Methodology research
- Program module design
- Web system implementation design
- Project report writing

Each task is assigned to members with primary responsibility and includes clearly stated start/end dates, number of working days, completion percentage, and task outcomes.

Table II.2.1. WBS: Work Breakdown Structure.

ID	Tasks	Workers	Start	End	Days	% complete	Workday
1	**Make Idea and division of work tasks**		Th 5 1/09/25	Th 4 1/22/25	42		30
1.1	Idea + work tasks	Hoàng xuân Cảnh	Th 5 1/09/25	Th 4 1/22/25	14	100%	10
1.2	Idea + work tasks	Hoàng văn Bắc	Th 5 1/09/25	Th 4 1/22/25	14	100%	10
1.3	Idea + work tasks	Nguyễn Việt Anh	Th 5 1/09/25	Th 4 1/22/25	14	100%	10
1.4							
2	**Data collection**		Th 5 1/23/25	Th 4 2/05/25	42		30
2.1	From SSI	Hoàng xuân cảnh	Th 5 1/23/25	Th 4 2/05/25	14	100%	10
2.2	From Vnstock	Hoàng văn Bắc	Th 5 1/23/25	Th 4 2/05/25	14	100%	10
2.3	From Global Factor Data	Nguyễn Việt Anh	Th 5 1/23/25	Th 4 2/05/25	14	100%	10
2.4							
3	**Research methods**		Th 5	Th 4	49		35

			2/06/25	3/26/25			
3.1	Feature Engineering and Label Generation for BB	Nguyễn Việt Anh	Th 5 2/06/25	Th 2 3/03/25	26	100%	18
3.2	Evaluation of Feature Engineering Results for BB	Nguyễn Việt Anh	Th 3 3/04/25	Th 7 3/08/25	5	100%	4
3.3	Feature Engineering and Label Generation for UD	Nguyễn Việt Anh	CN 3/09/25	Th 2 3/17/25	9	100%	6
3.4	Evaluation of Feature Engineering Results for UD	Nguyễn Việt Anh	Th 3 3/18/25	Th 7 3/22/25	5	100%	4
3.5	Global Factor Data	Nguyễn Việt Anh	CN 3/23/25	Th 2 3/24/25	2	100%	1
3.6	Evaluation of Results under Feature Transformation	Nguyễn Việt Anh	Th 3 3/25/25	Th 4 3/26/25	2	100%	2
3.7							
4	**Multi-Factor Model**		Th 5 3/27/25	Th 3 4/08/25	13		9
4.1	Portfolio Return Explanation and Prediction	Nguyễn Việt Anh	Th 5 3/27/25	Th 3 4/08/25	13	100%	9
4.2							
5	**Module 1 design**		Th 5 2/06/25	Th 4 4/09/25	63		45
5.1	Building a program for BB	Hoàng văn Bắc	Th 5 2/06/25	CN 3/09/25	32	100%	22
5.2	Evaluation of BB model construction results	Hoàng văn Bắc	Th 2 3/10/25	Th 3 3/18/25	9	100%	7
5.3	Building a program for UD	Hoàng văn Bắc	Th 4 3/19/25	Th 4 4/02/25	15	100%	11
5.4	Evaluation of UD model construction results	Hoàng văn Bắc	Th 5 4/03/25	Th 4 4/09/25	7	100%	5
5.5							
6	**System design and implementation**		Th 5 2/06/25	Th 5 4/10/25	64		46
6.1	Front -end	Hoàng xuân cảnh	Th 5 2/06/25	Th 4 3/19/25	42	100%	30
6.2	Back-end	Hoàng xuân cảnh	Th 5 3/20/25	Th 5 4/10/25	22	100%	16
6.3							
7	**Project Report**		Th 4 4/09/25	Th 3 4/22/25	25		20

7.1	0. Slide	Nguyễn Việt Anh	Th 4 4/09/25	Th 4 4/09/25	1	100%	1
7.2	I. Project Introduction	Nguyễn Việt Anh	Th 5 4/10/25	Th 5 4/10/25	1	100%	1
7.3	II. Project Management Plan	Hoàng văn Bắc	Th 5 4/10/25	Th 5 4/10/25	1	100%	1
7.4	III. Existing Systems/State of the Art	Nguyễn Việt Anh	Th 6 4/11/25	CN 4/13/25	3	100%	1
7.5	IV.A. Research Questions and Objectives Based on Collected Data	Nguyễn Việt Anh	Th 2 4/14/25	Th 4 4/16/25	3	100%	3
7.6	IV.B. Module 1 design	Hoàng văn Bắc	Th 6 4/11/25	Th 4 4/16/25	6	100%	4
7.7	1.2.2. SVM Model	Nguyễn Việt Anh	Th 5 4/17/25	Th 5 4/17/25	1	100%	1
7.8	IV.C. Multi-Factor Model for Portfolio Return Explanation and Prediction	Nguyễn Việt Anh	Th 6 4/18/25	Th 6 4/18/25	1	100%	1
7.9	V. System Design and Implementation (Web)	Hoàng xuân cảnh	Th 6 4/11/25	Th 4 4/16/25	6	100%	4
7.10	VI. Results and Discussion	Nguyễn Việt Anh	Th 7 4/19/25	Th 2 4/21/25	3	100%	1
7.11	VII. Conclusion	Nguyễn Việt Anh	Th 3 4/22/25	Th 3 4/22/25	1	100%	1

Reference:*Project Management.xlsx*

2.2 Risk Management

Data Risks: Missing, inaccurate, or difficult-to-synchronize data.

- Solution: Identify reliable data sources, standardize and assign data responsibilities, and establish a secondary backup source.

Team Progress Desynchronization Risks:

- Solution: Hold regular team meetings and provide timely updates on progress to respond effectively to unexpected situations.

Technical Risks (e.g., bugs, tools, model non-convergence):

- Solution: Maintain detailed logs, use checkpoints by module during the optimizer and model training phases to react promptly and handle unforeseen issues.

Online Deployment Risks (deploying individual modules and the web interface):

- Solution: Check system compatibility, run tests in a staging environment before production, and prepare backups for services such as servers.

Timeline Risk Management is mapped according to the WBS, with contingency plans for each module. Extra time is allocated for each task—task durations are effectively doubled compared to the original deadlines to allow for unexpected delays.

2.3 Quality Management

Standards: Each module must be clearly defined (runnable code, correct data, and a model that meets the minimum accuracy threshold), with outputs standardized via API.

Monitoring Metrics:

- % Completion according to WBS: All tasks must meet the 100% standard.
- Number of working days compared to the planned schedule to ensure compliance with the course timeline.

Table II.2.3: Number of working days compared.

Member	Workday
Nguyễn Việt Anh	74
Hoàng xuân cảnh	70
Hoàng văn Bắc	70

2.4 Budget and funding

The project does not use any external funding. All activities are carried out using personal resources (personal computers, self-funded AWS student accounts, and free services such as Google Colab, GitHub, MongoDB, etc.).

2.5 Change management process

The project initially set out two development approaches:

- Way 1: Build the system from scratch based on the proposed design aligned with the academic requirements.
- Way 2: An expansion/upgrade plan for the future, involving the development of multiple modules that generalize market condition predictions. This may include new methods for label prediction, and the use of reinforcement learning models to simulate

user behavior and recommend optimal strategies. It also envisions upgrading the web application and eventually deploying it as a mobile app.

In the current phase, the team is following Way 1, ensuring the core pipeline and main data remain intact.

2.6 Conclusion and review

Summary: All modules were completed on schedule, with functioning code, complete data, and well-performing models.The experimental results were clearly reported, with objective evaluations highlighting areas for improvement to enhance accuracy. The web application runs stably, supporting core user functionalities, though further upgrades are needed to improve user experience.

Lessons Learned:

- Roles and responsibilities should be clearly defined from the beginning.
- Time estimation for each task needs to be more accurate.
- Teamwork skills, resource sharing, and troubleshooting should be further improved.

III. Existing Systems/State of the Art

1. Overview of the Field

The use of artificial intelligence (AI) to aid in investment decision-making has become a global trend due to the stock market's rising complexity and unpredictability. Compared to conventional techniques, modern AI systems can process vast amounts of data, identify hidden trend patterns, and predict market changes more accurately.

In line with this trend, our group's project, "Using AI for Intelligent Stock Trading," aims to develop a system that forecasts stock up/down trends, predicts bubble prices, and analyzes portfolio risks by fusing traditional financial quantitative models with contemporary machine learning models.

2. Historical Context

When investment companies like Renaissance Technologies and Two Sigma pioneered the use of automated trading algorithms in the 2000s, the integration of big data and AI in financial investment started to take off. By the 2010s, deep learning has advanced to the point where financial forecasting could better handle time series data with nonlinear properties thanks to models like Transformer and LSTM.

To assess portfolio performance and risk, standard financial quantitative models like the Carhart 4 Factors (1997), Fama-French 3 Factors (1993), and CAPM (1964) remained popular. They offer a tried-and-true method that works quite well.

3. Key Studies and Theories

The Fama-French 3 Factors model, which expands on the CAPM by including size and value factors, the Carhart 4 Factors model, which adds a momentum factor to further explain stock returns, and Sharpe's CAPM model, which demonstrates the connection between systematic risk and an asset's expected return, are foundational studies in this field.

Fischer and Krauss (2018) showed that LSTM is the most effective machine learning technique for stock market forecasting. A new age was also brought about by the groundbreaking study "Attention is All You Need" (Vaswani et al., 2017), which introduced the Transformer model, which made it possible to model complicated financial time series data in an effective and adaptable manner.

4. Technological Advancements

The use of AI in financial investment has been made more easier by recent technological developments. Long time series modeling and improved management of nonlinear interactions in financial data have been made possible by the development of models such as Transformer and LSTM. The process of fine-tuning the model has been expedited by hyperparameter optimization techniques like Random Search and Optuna.

Financial prediction models may now be reliably deployed online thanks to cloud computing and MLOps solutions like AWS, MongoDB, and FastAPI. Additionally, the prevalent problem of class imbalance in financial classification jobs has been addressed in large part by data balancing approaches such as SMOTE.

5. Comparison of Existing Systems

At the moment, financial research tools like TradingView AI and QuantConnect mostly use simple machine learning models or conventional technical indicators to forecast specific prices or trends. These systems hardly ever offer a thorough assessment of portfolio risks or incorporate in-depth financial quantitative analysis like the CAPM or Fama-French models.

By closely integrating short-term forecasting (up/down trends), stock bubble price identification, and long-term performance evaluation through quantitative financial models, the project "Using AI for Intelligent Stock Trading" overcomes these constraints.

High responsiveness and scalability are further ensured by the system's deployment on the AWS platform, which offers APIs for real-time queries and continuous data updates.

6. Gaps in the Literature/Technology

The majority of current systems only predict price trends without incorporating bubble price forecasting and thorough portfolio risk analysis, despite the fact that financial forecasting using AI has been the subject of much research. Furthermore, a lot of systems continue to function as "black boxes," with no openness into the reasoning behind their choices, which makes it challenging for users to evaluate the model's dependability.

By creating a system that not only forecasts trends but also identifies bubble prices and does quantitative portfolio risk assessments, our project seeks to close these gaps. It does this by using transparent evaluation measures like alpha, beta, and R-squared.

7. Justification for the Project

The "Using AI for Intelligent Stock Trading" project is groundbreaking in its integration of modern AI technologies (LSTM, Transformer, Logistic Regression, SVM) with the robustness of classical financial quantitative models to support investment decision-making.

The system not only predicts market trends and detects price bubbles but also provides investment recommendations based on comprehensive risk and portfolio performance evaluations.

This multidimensional approach not only enhances investment efficiency but also offers significant potential for practical application in the financial sector, while simultaneously promoting the development of the Fintech industry in Vietnam in the future.

IV. Methodology

This research introduces a system designed to aid investment decisions by synergizing advanced machine learning techniques with established quantitative financial models. The deployment follows an end-to-end pipeline encompassing data acquisition, processing and normalization, feature engineering, model training, and the delivery of forecasting results via an API.

Information is sourced from two key origins. The primary source is the SSI trading platform, providing a historical dataset of trading activity from January 1, 2006, to the present day for 1631 publicly traded stocks in Vietnam. This dataset includes stock tickers, transaction dates, opening, peak, trough, and closing prices, alongside trading volumes, aggregated across daily, weekly, and monthly intervals. The secondary source is Global Factor Data, spanning from January 2020 to the current month, featuring monthly financial factors such as market excess return (mkt), risk-free rate (rf), investment patterns (inv), profitability metrics (profit), momentum indicators (mon), value characteristics, and company size (size), intended for the long-horizon quantitative model.

Upon collection, the raw data undergoes processing to handle missing entries, smooth temporal sequences, and compute relevant technical indicators like investment returns (ret), price volatility, maximum drawdown (MDD), the Relative Strength Index (RSI), 7/14/21-day moving averages (MA), rolling standard deviation, and adjusted logarithmic prices. Furthermore, we employ the GSADF test, implemented through a rolling window ADF test on adjusted log prices, to identify potential price bubbles.

The central problem is decomposed into two core tasks: predicting financial bubble occurrences (BB) and forecasting upward or downward stock price movements (UD). For each of these tasks, we utilize two distinct categories of models. The conventional model set comprises Logistic Regression (LR) and Support Vector Machine (SVM), operating on structured 2D data, producing binary classification outcomes. The time-series model set includes Long Short-Term Memory (LSTM) networks and Transformer architectures, processing 3D sequential data, outputting binary prediction probabilities.

Concurrently, to assess the performance and risk associated with long-term investment portfolios, this study applies established quantitative finance models, namely the Capital

Asset Pricing Model (CAPM), the Fama-French 3-factor model, the Carhart 4-factor model, and the Fama-French 5-factor [1] model. These models serve to validate the reasonableness of the portfolio following predictions and to provide a quantitative lens for enhancing the quality of investment decisions

A. Research Questions and Objectives Based on Collected Data

1. Research Questions and Objectives

This study seeks to develop a sophisticated investment decision support framework by integrating cutting-edge machine learning techniques with established quantitative finance models. The implementation follows a complete pipeline, encompassing data acquisition, processing and standardization, feature generation, model training, and the delivery of predictive insights via an API.

In response to increasingly turbulent financial markets characterized by numerous non-linear influences, the application of machine learning in financial forecasting has emerged as a critical area of research. This work aims to construct an intelligent system for investment prediction and evaluation, blending technical data, market information, and quantitative financial models to empower investors in making more informed choices.

Specifically, this research pursues three primary goals:

(1) To forecast whether stock prices will rise or fall in the upcoming week (Up/Down prediction), enabling investors to proactively manage short-term strategies.

(2) To identify instances of price bubbles at an early stage, with the aim of mitigating risks associated with unsustainable market surges and potential sharp corrections.

(3) To assess the effectiveness of investments and the level of portfolio risk using multi-factor models such as CAPM, Fama-French 3-factor, Carhart 4-factor, and Fama-French 5-factor, thereby providing a quantitative and practical validation of the system's predictive accuracy.

Stemming from these objectives, the research team formulates the following specific inquiries:

(1) Do advanced machine learning models (like LSTM and Transformer networks) offer superior predictive accuracy in forecasting market trends compared to conventional models such as logistic regression?

(2) Does the integration of technical data, macroeconomic indicators, and time series validation outcomes enhance the capability to detect price bubbles relative to approaches relying on isolated factors?

(3) Can the investment performance of a portfolio constructed based on machine learning model predictions be clearly substantiated through quantitative metrics like alpha (excess return) and beta (risk exposure) derived from multi-factor models?

2. Data Collection and Preprocessing

Input data is the core component in building financial market forecasting models. In this research, data is collected from two main sources: the SSI securities trading platform and the Global Factor Data system. The collection process is systematic, ensuring comprehensive market coverage and the historical depth necessary for machine learning and quantitative finance models.

2.1. Collection

2.1.1. Stock Trading Data from SSI

The first data source is collected from the SSI trading platform (https://www.ssi.com.vn/), through a combination of API crawling and libraries that support communication with market data. The data includes historical trading information for 1631 stock codes listed on the Vietnamese stock market, with a time span from January 1, 2006, to April 11, 2025. The data is initially retrieved on a daily basis and then processed to create weekly and monthly datasets, serving different prediction objectives (short-term, medium-term).

Table IV.A.2.1.1. The original daily data.

Variable Name	Description	Data Type
Ticker	Stock code	string
Date	Trading date	datetime
Open	Opening price	float
Hight	Highest price	float
Low	Lowest price	float
Close	Closing price	float
Volume	Trading volume	int

Starting from the original daily stock data, we used the Pandas library in Python to resample the time series to levels appropriate for each specific task, including weekly and monthly frequencies. This conversion resulted in two new datasets: stock_weekly_data and stock_monthly_data, aggregated as follows:

- The Ticker remains unchanged.
- The Date is set to the first day of the corresponding week or month.
- Open is taken as the opening price of the first trading session in the period.
- High is the highest price recorded during the entire week/month.
- Low is the lowest price recorded.
- Close is the closing price of the last trading session.
- Volume is the total trading volume over the entire period.

This aggregation serves two clear objectives in the study.

Weekly data is used for the Up/Down trend prediction task, where short-term fluctuations are the focus.

Conversely, monthly data is used for the Bubble Detection task, which requires a longer-term and more stable perspective to identify abnormal deviations between stock prices and their intrinsic values.

This data processing approach ensures the consistency of the time series and the reliability of technical indicators used as input features for the machine learning models.

2.1.2. Macroeconomic Factor Data from Global Factor Data

The second data source is Global Factor Data (Jensen et al., 2023), which is used to construct multi-factor financial quantitative models, serving the goal of evaluating portfolio performance and risk at a macro level. The data was collected from sources such as the Ken French Data Library, FRED, and several other financial data systems, aggregated through an internal data repository.

The collection period for the Global Factor Data spans from January 2020 to April 2025, with a monthly frequency. The dataset, named global_factor_data, includes the following variables:

Table IV.A.2.1.2. Global Factor Data.

Variable Name	Mô tả
Data	Month - Year
Mkt	Market return
Rf	Risk-free rate
SMB	Size factor (Small Minus Big)
HML	Value factor (High Minus Low)
MOM	Momentum factor
RMW	Profitability factor (Robust Minus Weak)
CMA	Investment factor (Conservative Minus Aggressive)

These variables are used to support portfolio valuation models such as CAPM, Fama-French 3-Factor, Carhart 4-Factor , and Fama-French 5-Factor, thereby enabling the evaluation of alpha and beta metrics for portfolios generated from the machine learning model's predictions (Carhart, 1997; Fama & French, 1993, 2015; Sharpe, 1964).

All collected and processed data is stored in standard CSV format, using commas (,) as delimiters, UTF-8 encoding, and saved within the internal system to serve subsequent preprocessing and model training steps.

2.2. Data Processing

2.2.1. Data Processing Workflow for Up/Down Trend Prediction

The original daily data was aggregated into weekly cycles to enable the model to better capture short-term market fluctuations. Only stocks with continuous data from **January 6, 2013, to April 13, 2025** — corresponding to over 12 years of data — were retained.

The aggregation process involved standardizing:

- Opening prices at the beginning of the week,
- Closing prices at the end of the week,
- Highest and lowest prices during the week,
- Total weekly trading volume.

From the standardized dataset, the research team extracted important technical features to serve as input for machine learning models, including:

- **Weekly returns (ret),**
- **Volatility,**
- **Price fluctuation indicators** (HL, LO, SR, PM),
- **Maximum Drawdown (MDD),**
- **Average True Range (ATR)** for measuring true volatility.

Additionally, the team calculated:

- **TVV** (standard deviation of volume),
- **SK** (skewness of distribution),
- **Median_HL,**
- **Moving averages** (ma7_t, ma14_t, ma21_t) to describe trend behaviors.

The feature **insec_t** was computed as the ratio of trading volume to price range, representing adjusted liquidity. Furthermore, the **VNIndex** movement was incorporated through the feature **vnipc_t**, reflecting broader market fluctuations.

The full list of extracted features is presented in Table IV.A.2.2.1.

The target label was assigned based on the return of the following week (**fore_di_rt**):

- If **≥ 0**, the label is **1** (Up);
- If **< 0**, the label is **0** (Down).

Table IV.A.2.2.1. Input Feature Table for the Up/Down Prediction Task.

Feature	Description	Fomula
ret	Return	$(Close_t - Close_{t-1})/Close_{t-1}$
Volatility	Rolling Std of Return (5 weeks)	$\sigma(ret_{t-5:t})$
HL	High - Low	$High_t - Low_t$
LO	Low - Open	$Close_t - Low_t$
SR	Close - Open	$Close_t - Open_t$
PM	Price Midpoint	$(High_t + Low_t)/2$
MDD	Max Drawdown	$Close_t/max(Close_{0:t}) - 1$
TVW	Volume Std (5 weeks)	$\sigma(Volume_{t-5:t})$
ATR	Average True Range	$MA_5(TR_t)$

Feature	Description	Fomula
SK	Skewness (5 weeks)	$Skew(Close_t)$
Median_HL	Median High-Low	$(High_t + Low_t)/2$
variation_t	Price Variation	$(Close_t/Close_{t-1}) - 1$
ma7_t	7-week MA	$mean(Close_{t-6:t})$
ma14_t	14-week MA	$mean(Close_{t-13:t})$
ma21_t	21-week MA	$mean(Close_{t-20:t})$
s_d7_t	Std of Close (7 weeks)	$\sigma(Close_{t-6:t})$
insec_t	Volume per Range	$Volume_t/(High_t - Low_t)$
vnipc_t	VNINDEX Return	$(VNI_t - VNI_{t-1})/VNI_{t-1}$

2.2.2 Data for the Financial Bubble Detection Task

A bubble happens when the price of a stock rises much higher than its real value.It often comes from excitement, rumors, or too much optimism from investors. To detect bubbles, we check if the price keeps going up too much compared to its true worth over time (Chen et al., 2021).

For the bubble price detection task, data is aggregated on a monthly basis to ensure greater stability, making it suitable for statistical testing techniques. Only stocks with continuous time series data from January 2013 to April 2025 were retained. The processed dataset includes monthly fields for opening price, closing price, highest price, lowest price, and trading volume.

After standardization, features were calculated similarly to those for the Up/Down prediction task, including:

- ret (return),
- Volatility,

- HL, LO, SR, PM (price fluctuation indicators),
- MDD (maximum drawdown),
- TVV (volume volatility),
- ATR (average true range),
- SK (skewness),
- Median_HL,
- Moving averages and volatility indicators such as s_d7_t and variation_t.

Additionally, log_price (the logarithm of the closing price) and RET (excess return) were included to support time series stationarity tests.

The full list of features is detailed in Table IV.A.2.2.2.

Table IV.A.2.2.2. List of Input Features for the Bubble Detection Task.

Feature	Description	Formula
ret	Return	$(Close_t - Close_{t-1})/Close_{t-1}$
Volatility	Rolling Std of Return (3 months)	$\sigma(ret_{t-3:t})$
HL	High - Low	$High_t - Low_t$
LO	Low - Open	$Close_t - Low_t$
SR	Close - Open	$Close_t - Open_t$
PM	Price Midpoint	$(High_t + Low_t)/2$
MDD	Max Drawdown	$Close_t/max(Close_{0:t}) - 1$
TVW	Volume Std (3 months)	$\sigma(Volume_{t-3:t})$
ATR	Average True Range	$MA_5(TR_t)$
SK	Skewness (5 months)	$Skew(Close_t)$
Median_HL	Median High-Low	$(High_t + Low_t)/2$
fore_di_rt	Forward Return	$(Close_{t+1}/Close_t) - 1$
variation_t	Price Variation	$(Close_t/Close_{t-1}) - 1$
ma7_t	7-month MA	$mean(Close_{t-6:t})$
ma14_t	14-month MA	$mean(Close_{t-13:t})$
ma21_t	21-month MA	$mean(Close_{t-20:t})$
s_d7_t	Std of Close (7 months)	$\sigma(Close_{t-6:t})$
log_price	Log Price	$Log(Close_t)$
RET	Excess Return	$ret - 0.005$

To assign labels, the team applied the ADF test (Augmented Dickey-Fuller test) on the log_price series using a sliding window of 20 months.

If a window segment had a p-value < 0.05 and ADF-statistic < -2.9, the corresponding point was labeled with is_bubble = 1.

The bubble occurrence rate (bubble_percent) for each stock was calculated to support further analysis.

The overall average bubble rate was 47.6%, indicating a relatively high prevalence of bubble phenomena in the research dataset.

The distribution of bubble occurrence rates by stock ticker is shown in Figure IV.A.2.2.2.

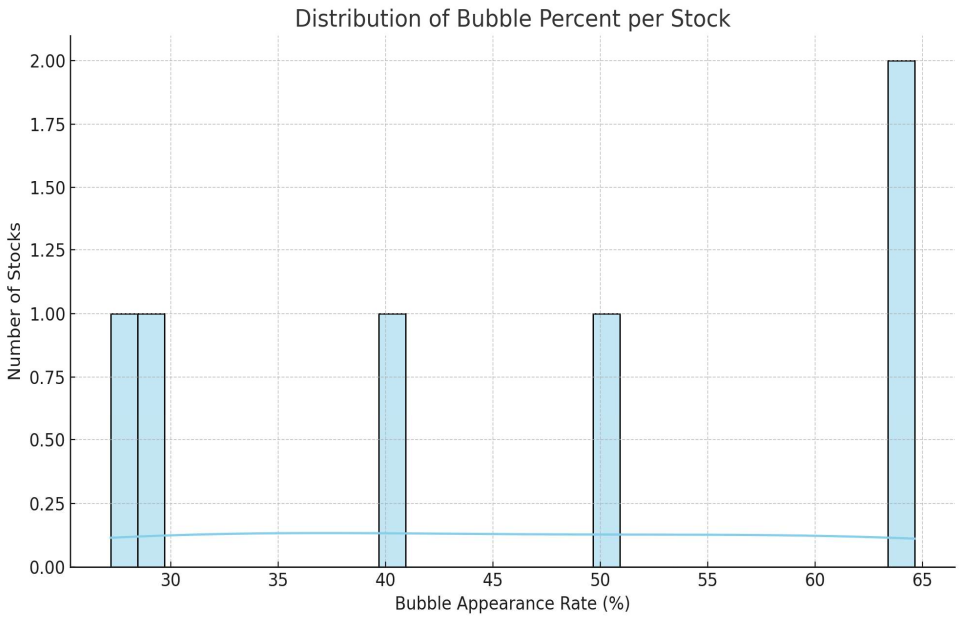

Figure IV.A.2.2.2. Distribution of Bubble Occurrence Rates (bubble_percent) by Stock Ticker (Source: Own work).

3. Evaluate live data with basic Logit model

In the preliminary phase of our research, a basic Logistic Regression (Logit) model was employed to evaluate the predictive feasibility of the proposed tasks on live stock trading data.

3.1. Bubble Price Prediction (BB)

For the task of Bubble Price Detection, the Logit model demonstrated encouraging results.

The Top 5 stock tickers ranked by accuracy and corresponding F1-scores are summarized in the following table:

Table IV.A.3. Top 5 tickers by Accuracy for BB.

Ticker	Accuracy (%)	F1-score
AMC	90.00	0.9333
ABI	86.67	0.6667
ACC	86.67	0.7143
ADP	83.33	0.7619
AGF	83.33	0.8

These results indicate that even with a basic model and minimal feature engineering, the system is capable of achieving relatively high predictive performance for certain stocks. This reinforces the hypothesis that stock price bubbles can be identified based on appropriately engineered features extracted from historical data.

3.2. Up/Down Trend Prediction (UD)

In addition to bubble detection, the Up/Down trend prediction task (UD) also exhibits potential for model-based forecasting (Khoa & Huynh, 2021). Although preliminary experiments were not extensively detailed at this stage, the underlying analysis suggests that integrating more sophisticated models (such as LSTM and Transformer) would likely improve prediction performance, especially for capturing complex short-term stock movements.

Thus, building upon the initial insights, it is proposed to establish a dedicated module specifically for the UD prediction task, leveraging deep learning models and optimized data pipelines.

3.3. Multi-Factor Portfolio Risk Assessment

Beyond short-term forecasting, it is imperative to recognize that investment decisions require a comprehensive understanding of risk. Accordingly, the project also proposes the construction of a **multi-factor portfolio evaluation framework** using established financial models such as CAPM, Fama-French 3-Factor, and Carhart 4-Factor models.

Through this approach, key indicators such as Alpha (excess return), Beta (systematic risk), and R-squared (model fit) will be quantified for each portfolio derived from machine learning predictions. This integration ensures that the investment recommendations are not only data-driven but also grounded in rigorous financial risk analysis, providing investors with greater confidence and transparency in their decision-making process (Douagi et al., 2021).

4. Propose solution

The AI system for stock analysis operates through a series of sequential steps at Figure IV.I.4 . First, raw stock data such as price, trading volume, and technical indicators are inputted. Next, the AI model layer handles two main tasks: predicting the upward/downward trend of the stock and detecting abnormal price bubble phenomena. Based on these predictions, the system evaluates the risk of the investment portfolio and provides alerts if signs of instability are detected. Finally, the results are displayed via a web API, allowing users to interact, track trends, and check the outputs of individual stock tickers in a visual manner. check the outputs of individual stock tickers in a visual manner.

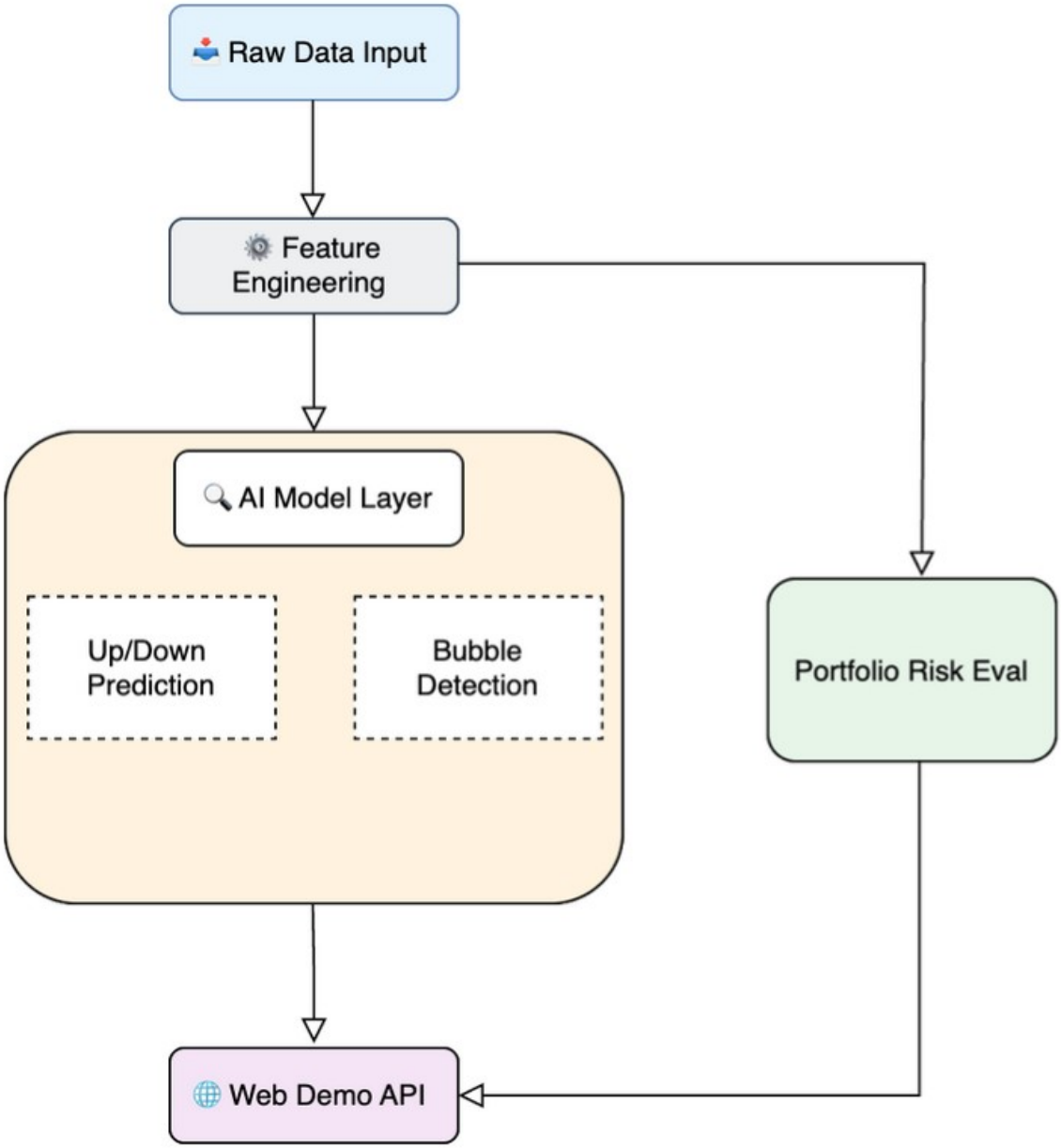

Figure IV.A.4: Using AI for Intelligent stock trading (Source: Own work).

B. Module 1 design

The module is designed to include Program 1, which runs independently to perform optimization, followed by training to ensure security. It then generates a file predicting the market condition. The results are subsequently pushed to MongoDB. Based on real-time data stored in MongoDB, Program 2 retrieves the data and provides an API (FastAPI) hosted on AWS at Figure IV.II.

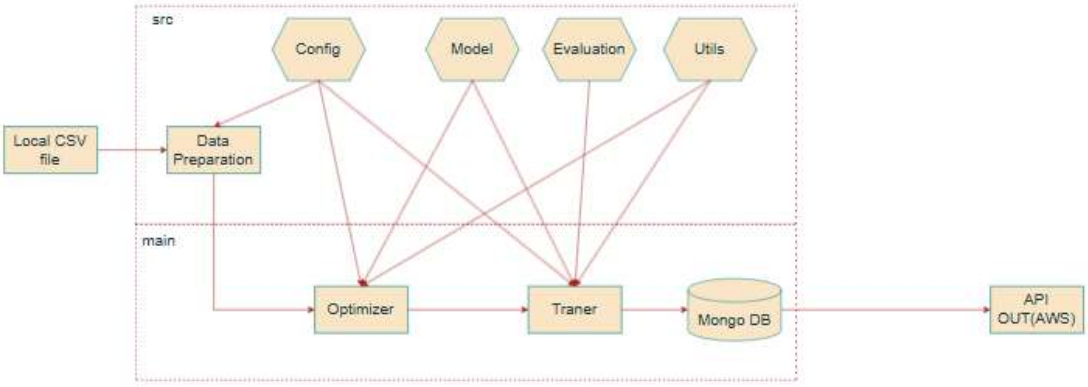

Figure IV.B: End-to-End Pipeline: Feature Engineering Input → API Output

1. Select features and techniques for Program 1

1.1 Data preparation

The process of transforming and processing input data is tailored to be compatible with all four models and to improve accuracy. This ensures fair evaluation through the following methods: Data Loading, Feature Preparation, Feature Selection, and Data Splitting at Figure IV.II.1.1.

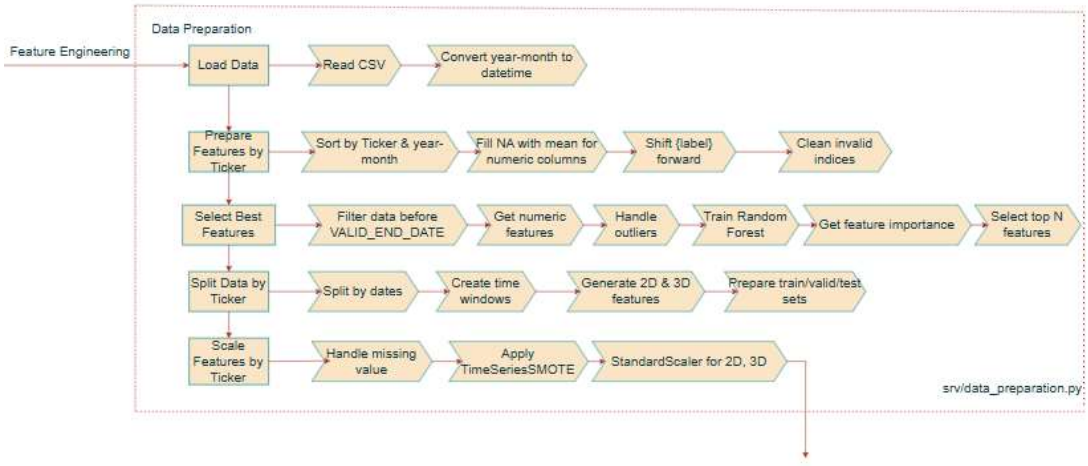

Figure IV.B.1.1: Data preparation (Source: Own work).

1.1.1. Load data

Convert 'Month-Year', 'Date' format to datetime:

Formula:

$t_{datetime} = todatetime(t_{year-month})$

Purpose: convert standard feature to format datetime.

1.1.2 Feature Preparation

1.1.2.1. Missing value imputation

For each "$f \in F$" feature , missing values are replaced with the mean value of that feature:

Formula:

$$x_f^{(i)} = \begin{cases} x_f^{(i)} & available \\ \frac{1}{|X_f|} \sum_{x \in X_f} x & missing \end{cases}$$

Where:

- $X_f$represents the set of non-missing values of the "f" feature.
- $x_f^{(i)}$ is the value of the "f" feature at "$i-th$" sample.

Purpose: To prevent input features from having corrupted or incomplete data.

1.1.2.2. Label shift

Labels are assigned to time point based on the "Bubble (BB)" or "Up or Down (UD)" status at $t+1$ time point.

Formula:

BB: $y_t = isbubble_{t+1}$

UD: $y_t = foredirt_{t+1}$

Purpose: To create data that allows prediction one week in advance for "Up or Down" labels, and one month in advance for "Bubble" labels.

1.1.3. Feature Selection

1.1.3.1. Outlier Clipping

For each "f" feature, the values are limited to the range:

Formula:

$$x_(i)^f = clip(x_(i)^f, \mu_f - 5\sigma_f, \mu_f + 5\sigma_f)$$

Where:

- " μ_f, σ_f ": are the mean and standard deviation of " f " feature for each Ticker, respectively.

Purpose: To cover approximately 99.99994% of the data values and eliminate outliers or abnormal computations.

1.1.3.2. Feature importance

The "f" feature importance is calculated through a Random Forest using the training set:

Formula:

$$I(f) = \frac{1}{N_{trees}} \sum_{t-1}^{N_{trees}} i_t(f)$$

Where:

- N_{trees} : is the total number of decision trees in the Random Forest model (n_estimators=100).
- $i_t(f)$: is the contribution of "f" feature in "i" tree at "t"data point.
-Select a subset of important "$F_{top} \subseteq F$" features with k top feature to avoid using too many features when there are fewer data rows.

Table IV.B.1.1.3.2: Feature importance.

BB	UD
k=10	k= 15

Purpose: To ensure better convergence during training, avoid irrelevant features that may cause noise, and reduce overfitting when there are too many features and limited data rows.

1.1.4. Data Splitting

1.1.4.1. Split training, validation, testing files by timeline

In order to construct a reliable and unbiased model evaluation, the original dataset is partitioned into three subsets training, validation, and testing based on the timeline "t".

Formula:

$$X_{train} = \{x_t \mid t < t_{train}\}$$

$$X_{val} = \{x_t \mid t_{train} + WINDOWSIZE < t < t_{val}\}$$

$$X_{test} = \{x_t \mid t > t_{val} + WINDOWSIZE\}$$

Table IV.B.1.1.4.1: Split training, validation, testing files by timeline.

BB	UD
TRAIN_END_DATE = '2019-01-01'	TRAIN_END_DATE = '2022-03-30'
VALID_END_DATE = '2022-06-01'	VALID_END_DATE = '2024-03-30'
WINDOW_SIZE = 4	WINDOW_SIZE = 13
60:30:10 ratio approximately and take the previous 4 months observation data in short term and remove window_size between files to avoid leak data.	80:15:5 ratio approximately and take the previous 13 weeks observation data in short term as observation and remove window_size between files to avoid leak data.

Purpose: split 3 files to evaluate fairness during prediction generation.

1.1.4.2. Time Window Transformation

In order to model temporal dependencies in time-series data, raw input data are transformed into two possible structures: 2D (flattened) or 3D (time-series) representations.

2D Data (Flattened Representation):

Formula:

$$X_{2D}^{(i)} \in R^{B \times (W.F)}$$

3D Data (Time-Series Representation):

Formula:

$$X_{3D}^{(i)} \in R^{B \times W \times F}$$

Purpose:

- 2D Data (Flattened): Suitable for traditional machine learning models like SVM, LR, etc.
- 3D Data (Time-Series): Suitable for deep learning models like LSTM, TFM, etc., and for computations with GPUs.

1.1.5 Feature Scaling And Augmentation

1.1.5.1. Standardization

To ensure the model converges effectively and avoid numerical instability during training, feature values are standardized. Standardization transforms the original feature distribution into one with zero mean and unit variance.

Formula:

$$x_f^{(i)} = \frac{x_f^{(i)} - \mu_f}{\sigma_f}$$

Where:

- μ_f, σ_f are the mean and standard deviation of "f" feature for each Ticker.

Purpose: To help the model converge effectively.

1.1.5.2 SMOTE (Synthetic Minority Over-sampling Technique)

To address the problem of class imbalance particularly when the minority class is underrepresented the SMOTE algorithm is applied to generate synthetic samples by interpolating between neighboring minority class instances.

Formula:

$$x_{new} = \alpha x_1 + (1 - \alpha) x_2 + \xi$$

Trong đó:

- x_1, x_2: is points close to each other in the same class (minority class)
- $\alpha \sim \mu(0, 1)$: interpolation coefficient.
- $\xi \sim \eta(0, \sigma^2)$: Gaussian noise.

Purpose: New data points are generated to balance the label ratio to 1:1, ensuring stability during training and preventing the model from making biased predictions toward one label.

1.2. Model

1.2.1. Logistic Regression Model

1.2.1.1. Input

The model receives **2D flattened input data**, where each sample is represented as a vector of concatenated features across the time window (Bailly et al., 2022).

Formula:

$X_{2D}^{(i)} \in R^{B\times(W.F)}$

Where:

B: is the batch size,

W: is the time window length,

F: is the number of features.

1.2.1.2. Hypothesis Function:

The hypothesis function of Logistic Regression is defined as:

Formula:

$$h_\theta(x) = \sigma(w^T x + b)_, \in R^{B\times 2}$$

$$\sigma^z = \frac{1}{1-e^{-z}}$$

Output $R^{B\times 2}$ represents the output probability distribution over two classes (binary classification: 0 or 1)

1.2.1.3. Cost Function

The objective of Logistic Regression is to minimize the discrepancy between the predicted probabilities and the actual labels. This is achieved by optimizing the Binary Cross-Entropy loss function, extended with L2 regularization for better generalization.

Formula:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}log(h_\theta(x^{(i)}) + (1-y^{(i)})log(1-h_\theta)(x^{(i)})] + \lambda\| w \|^2$$

1.2.1.4. Optimization với L2 Regularization:

To prevent overfitting and promote generalization, L2 regularization is incorporated into the loss function during the training of the Logistic Regression model.

Formula:

$$L_{reg} = \min_{\theta} J(\theta) + \frac{\lambda}{2} \| w \|^2$$

$$\lambda = \frac{1}{C}$$

Purpose: Use L2 regularization to gradually shrink the weights, though rarely to zero thus retaining all features.

1.2.1.4. In scikit-learn:

Formula:

$$\lambda = \frac{1}{C}$$

Where:

- C: Taken from LR_PARAM_RANGES['C'] – controls the regularization strength
  + Smaller C → stronger regularization (helps prevent overfitting)
  + Larger C → weaker regularization (model may overfit)

Table IV.B.1.2.1.4: Meaning of parameters in LR_PARAM_RANGES.

Parameter	Description	Explanation
C	$\lambda = \frac{1}{C}$	Regularization L2. Controls model complexity.
Solver	Doesn't change the formula, but affects the optimization approach	Solvers like newton-cg, lbfgs, sag, and saga use gradient-based methods to optimize the loss function.
max_iter	Limits the number of optimization iterations	Stops when either the iteration limit is reached or the algorithm converges.
tol	Convergence criterion: stops when $\Delta L < tol$	Smaller values require more precise convergence but take longer to compute.

1.2.2. SVM Model

1.2.2.1. input

The input for the SVM model is a set of labeled training example $\{x_i, y_i\}$, where $x_i \in R^n$ represents the feature vector and $y_i \in \{-1, +1\}$ is the corresponding class label. The data is often flattened into 2D matrices to be compatible with standard machine learning libraries (Valero-Carreras et al., 2023).

1.2.2.2. Hypothesis Function

SVM defines a decision function based on a separating hyperplane:

Formula:

$$f_x = w^T x + b$$

The prediction is determined by the sign of the output:

Formula:

$$\hat{y} = sign(f(x)) \begin{cases} +1 & f(x) \geq 0 \\ -1 & f(x) < 0 \end{cases}$$

1.2.2.3. Cost Function (Objective with Soft Margin and L2 Regularization)

To address non-separable data, SVM introduces slack variables ξ_i and optimizes the following objective function:

Formula:

$$\min_{w,b,\xi} \frac{1}{2} \| w \|^2 + C \sum_{i=1}^{n} \xi_i$$

subject to the constraints:

Formula:

$$y_i(w^T x_i + b) \geq 1 - \xi_i, \xi_i \geq 0$$

Here, C is a regularization parameter that governs the trade-off between maximizing the margin and minimizing classification errors.

1.2.2.4. Kernel Trick (for Non-linear Separation)

When the data is not linearly separable, a kernel function is used to map the input features into a higher-dimensional space, enabling linear separation:

Formula:

$$K(x_i, x_j) - \langle \phi(x_i), \phi(x_j) \rangle$$

Some commonly used kernel functions include:

- Linear Kernel:Linear Separation

Formula

$$K(x, x^{'}) = x^T x^{'}$$

- Polynomial Kernel:Nonlinear, distance based

Formula:

$$K(x, x') = (\gamma x^T x' + r)^d$$

- RBF (Gaussian) Kernel: Nonlinear, polynomial form

Formula:

$$K(x, x') = exp(-\gamma \parallel x - x' \parallel^2)$$

1.2.2.5. In scikit-learn:

Table IV.B.1.2.2.5: Meaning of parameters in SVM_PARAM_RANGES.

Parameter	Description	Explanation
C	Regularization parameter	Controls the penalty for misclassified points. A larger C narrower margin, fewer misclassifications, but higher risk of overfitting.
kernel	Type of kernel	Defines the feature mapping $\phi(x)$ indirectly via the kernel trick.
gamma	Parameter for rbf and poly kernels	For rbf: controls the influence range of a single training point. Larger gamma → smaller influence area → higher risk of overfitting.
tol	Convergence tolerance	Training stops when the change in model parameters becomes smaller than tol. Balances training speed and accuracy.

1.2.3. LSTM Model

1.2.3.1. Input

3D Data (time-series):

Formula:

$$X_{3D}^{(i)} \in R^{B \times W \times F}$$

1.2.3.2. LSTMCell:

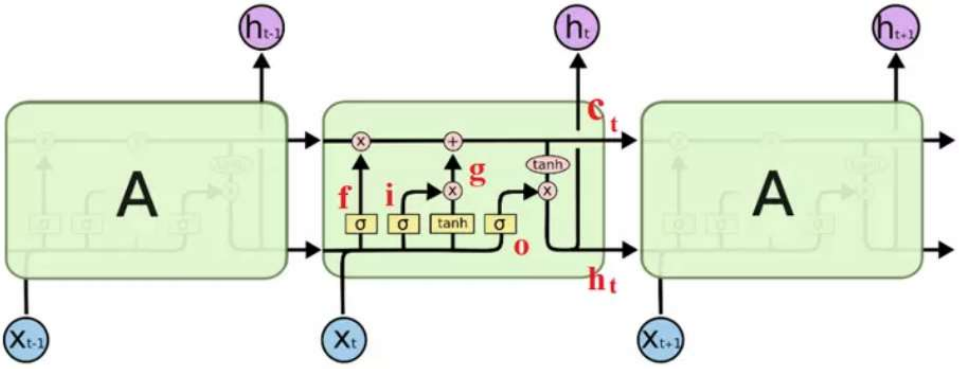

Figure IV.B.1.2.3.2: LSTMCell.

Forget gate:

Formula:

$$f_t = \sigma(W_f . [h_{t-1}, x_t] + b_f),$$

$$x_t \in R^{featrues}, h_{t-1}, b_f, f_t \in R^h, [h_{t-1}, x_t] \in R^{(features+h)}, W_f \in R^{B\times(features+h)}$$

Description: This gate determines which parts of the "C_{t-1}" cell state will be forgotten. The inputs include the "h_{t-1}" previous hidden state and "h_{t-1}" current input , which are multiplied by weights and passed through a sigmoid function to $f_t \in [0,1]^h$, indicating the level of forgetting (Wen & Li, 2023).

Formula:

$$i_t = \sigma(W_i . [h_{t-1}, x_t] + b_i),$$

$$W_i \in R^{B\times(features+h)}, i_t, b_i, h_{t-1} \in R^h, x_t \in R^{featrues}, [h_{t-1}, x_t] \in R^{(features+h)}$$

Description: This gate determines what new information will be added to the "C_t" cell state . Similar to the forget gate, it uses a sigmoid function to create a "$i_t \in [0,1]^h$" mask for the new information.

Candidate cell state:

Formula:

$$g_t = \tilde{C}_t = tanh(W_C . [h_{t-1}, x_t] + b_C),$$

$$W_C \in R^{B\times(features+h)}, g_t, b_C, h_{t-1} \in R^h, [h_{t-1}, x_t] \in R^{(features+h)}, x_t \in R^{featrues}$$

Description: This is a proposed new memory state value, combining the old information and the current input through a "$tanh$" nonlinear layer.

Update memory status:

Formula:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t,$$

$$C_t, C_{t-1}, \tilde{C}_t \in R^h$$

Description: The "C_t" cell state is updated by combining the retained portion from "C_{t-1}" and the new information from "g_t" , weighted by $f_t$and i_t.

Output gate:

Formula:

$$o_t = \sigma(W_o . [h_{t-1}, x_t] + b_o),$$

$$o_t, b_o, h_{t-1} \in R^h, W_o \in R^{B \times (features+h)}, [h_{t-1}, x_t] \in R^{(features+h)}, x_t \in R^{featrues}$$

Description: The output gate decides which part of the memory state will be output as the "h_t" current hidden state.

Hidden state:

Formula:

$$h_t = o_t * tanh(C_t),$$

$$h_t \in R^h$$

Description:

The "h_t" hidden state is calculated by applying a "$tanh$" function to the "C_t" current cell state , then multiplying it by the value from the output gate.

Where:

- σ: sigmoid function
- $*$: element-wise multiplication

1.2.3.3. Forward Pass

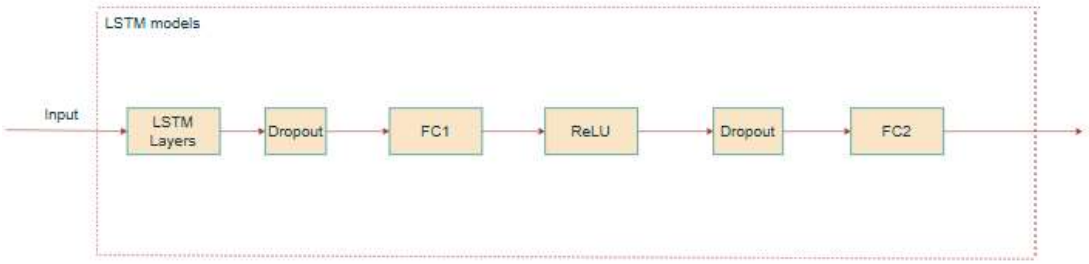

Figure IV.B.1.2.3.3: LSTM Model (Source: Own work).

Formula:

$$h_{out} = LSTM(x, W, b), \in R^{B \times T \times d_h}$$

Description: LSTM takes a sequential input "x_t" with "B" batch size, time steps "T", and "d_h" hidden size . The result is a sequence of "h_{out}" hidden state vectors.

Formula:

$$h_{last} = h_{out}[:, -1, :], \in R^{B \times d_h}$$

Description: Get the final hidden state of each sequence in the batch, representing the entire sequence.

Formula:

$$z_1 = Dropout(h_{last}), \in R^{B \times d_h}$$

Description: Apply dropout to reduce overfitting by randomly dropping some nodes in "h_{last}".

Formula:

$$h_1 = ReLU(W_1 z_1 + b_1), \in R^{B \times d_{ff}}$$

Description: The nonlinear hidden layer uses the ReLU activation function to expand the feature space before classification.

Purpose: Expanding the feature space enables the model to learn more complex representations by "$d_{ff} = 64$".

Formula:

$$z_2 = Dropout(h_1), \in R^{B \times d_{ff}}$$

Description: Dropout one more time to increase generalization

Formula:

$$y = W_2 z_2 + b_2, \in R^{B\times 2}$$

Description: The final linear layer to convert the representation into a probability distribution for 2 binary output classes.

Purpose: Predict the output probability for a binary classification problem.

1.2.4. Transformer Model

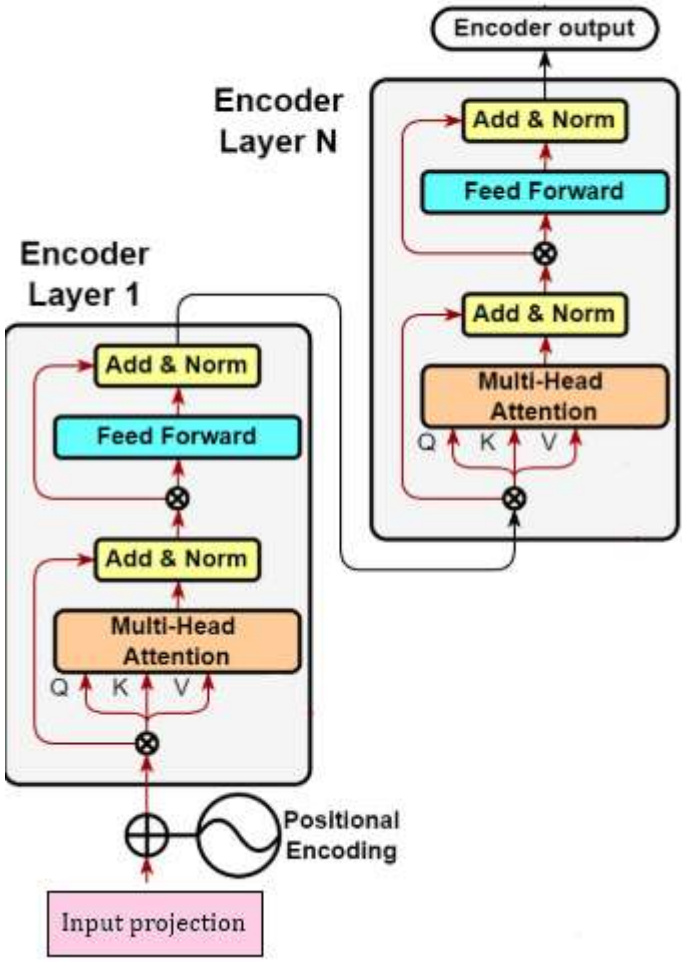

Figure IV.B.1.2.4: Transformer layer.

Purpose: use encoder to apply ro model which needs output, $\in R^{B\times 2}$ (Nassiri & Akhloufi, 2023).

1.2.4.1. Input projection:

In Transformer-based architectures, it is necessary to project input features into a fixed-size representation space before feeding them into the encoder.

This ensures that regardless of the original input dimensionality, the model receives inputs of dimension d_model, which is essential for multi-head attention and other operations.

3D Data (time-series):

$X_{3D}^{(i)} \in R^{B \times W \times F}$

Formula:

$$X_{proj} = XW_{proj} + b_{proj}$$

Where:

- $X_{proj} \in R^{B \times T \times d_{model}}$
- $W_{proj} \in R^{F \times d_{model}}$
- $b_{proj} \in R^{d_{model}}$

Description: project input vector from input_size to d_model.

Purpose: format input according to Transformer standard..

1.2.4.2. Learnable Positional Encoding:

Formula:

$$PE_{init}(pos, 2i) = sin(\frac{pos}{10000^{\frac{2i}{d_{model}}}})$$

$$PE_{init}(pos, 2i+1) = cos(\frac{pos}{10000^{\frac{2i}{d_{model}}}})$$

Description: This is a non-learnable (fixed) positional encoding method that uses sine and cosine functions with frequencies that by "$i-th$" . It helps the model recognize the positional order of elements in the input sequence.

1.2.4.2. Learnable version:

Formula:

$$PE_{learned} = PE_{init} + \theta_{PE},$$

$PE_{learned} \in R^{1\times L\times d_{model}}$, với $L \geq T$, L = max_seq_length = =5000

Description: Positional Encoding is initialized using the standard sine-cosine function, then added to a "θ_{PE}" learnable tensor . During training, the model learns to adjust this encoding to better fit the actual data.

Formula:

$$X_{pos} = X_{proj} + PE_{learned}[:, :T, :]$$

Description: When applying positional encoding, we only take the first steps corresponding to the actual sequence length by using slicing [:, :T, :].

Purpose: This " $PE_{Learned} \in R^{B\times T\times d_{model}}$ " is the positionally encoded input, ready for the encoder or decoder block.

1.2.4.3. Transformer Encoder Layer

1.2.4.3.1. Multi-Head Self-Attention

Formula:

$$Q = X_{pos}W_i^Q, K = X_{pos}W_i^K, V = X_{pos}W_i^V, \in R^{B\times T\times d_{qkv}}, d_{qkv} = \frac{d_{model}}{h}$$

Description: " Q, K, V "is the query, key, and value created from the " X_{pos} " positionally encoded input.

Formula:

$$head_i = Attention(Q, K, V) = Softmax(\frac{QK^T}{\sqrt{d_k}})V, \in R^{B\times T\times d_{qkv}}$$

Description: Multiple parallel "heads" are used to learn different relationships between tokens through softmax.

Formula:

$$MultiHead(X_{pos}) = Concat(head_1, \dots head_h)W^O, \in R^{B\times T\times d_{model}}$$

Description: After calculating attention for each head, they are concatenated and multiplied by "W^O" weights to aggregate the information.

Where:

- h: "nhead" number of heads
- i: rang$[1, h]$

- W^O:The aggregated matrix after concatenating all heads.

1.2.4.3. Feed forward

Formula:

$$h(x_{out}) = ReLU(x_{out}W_1 + b_1)_, \in R^{B\times T\times d_{ff}}.$$

Description: If "$d_{model} = (8, 24)$" then "$d_{ff} = d_{model\times 2}$" is a step in the FeedForward to create a larger representation space, helping the model learn deeper nonlinear relationships.

Purpose: To expand the representation space according by "W_1" .

Formula:

$$FFN(h) = hW_2 + b_2_, \in R^{B\times T\times d_{model}}$$

Description: After expansion and applying the activation function, this step brings the size back to the "d_{model}" original dimension to combine with the rest of the network.

Purpose: To restore the representation space.

Layer Normalization:

Formula:

$$x_{out} = LayerNorm(X_{pos} + Dropout(MultiHead(X_{pos})))_, \in R^{B\times T\times d_{model}}$$

Purpose: Normalize the output of the Multi-Head Attention layer to stabilize the training process and accelerate convergence.

Formula:

$$x_{final} = LayerNorm(x_{out} + Dropout(FFN(h(x_{out}))))_, \in R^{B\times T\times d_{model}}$$

Purpose: Apply normalization and dropout after the FeedForward block to stabilize the output and prevent overfitting.

1.2.4.4. Forward Pass

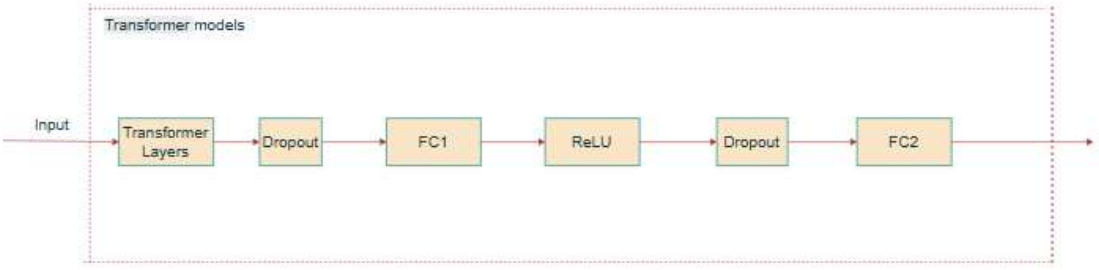

Figure IV.B.1.2.4.4: Transformer Model (Source: Own work).

Formula

$$h_{encoded} = TransfomerEncoder(X_{pos})_{, \in R^{B \times T \times d_{model}}}$$

Description: Transformer encoder takes a sequential input "X_{pos}" with "B" batch size, time steps "T", and "d_{model}" hidden size . The result is a sequence of "$h_{encoded}$" hidden state vectors.

Formula:

$$h_{last} = h_{encoded}[:, -1, :]_{, \in R^{B \times d_{model}}}$$

Description: Get the final hidden state of each sequence in the batch, representing the entire sequence.

Formula:

$$z_1 = Dropout(h_{last})_{, \in R^{B \times d_{model}}}$$

Description: Apply dropout to reduce overfitting by randomly dropping some nodes in h_{last}.

Formula:

$$h_1 = ReLU(W_1 z_1 + b_1), \in R^{B \times d_{ff}}$$

Description: The nonlinear hidden layer uses the ReLU activation function to expand the feature space before classification.

Purpose: Expanding the feature space enables the model to learn more complex representations by "$d_{ff} = 64$".

Formula:

$$z_2 = Dropout(h_1), \in R^{B \times d_{ff}}$$

Description: Dropout one more time to increase generalization

Formula:

$$y = W_2 z_2 + b_{2, \in R^{B \times 2}}$$

Description: The final linear layer to convert the representation into a probability distribution for 2 binary output classes.

Purpose: Predict the output probability for a binary classification problem.

2. Model training and validation

2.1. Input

2.1.1 Input for LR, SVM

Input shape: (ROW, WINDOW_SIZE* FEATURES)

- ROW: each row in each datasets.(sample)
- WINDOW_SIZE: time series length (WINDOW_SIZE = 4 for BB, 13 for UD from config)
- FEATURES: number of input features (number of columns of features)

2.1.2. Input for LSTM, Transformer

Input shape: (BATCH_SIZE, SEQUENCE_LENGTH, INPUT_SIZE)

- BATCH_SIZE: The number of samples per transmission is the model updated, each transmission is called mini batch size, the entire transmission of mini batch size is called epoch.
- SEQUENCE_LENGTH: time series length (WINDOW_SIZE = 4 for BB, 13 for UD from config)
- INPUT_SIZE: number of input features (number of columns of features)

Purpose: get past information

- Too small → not enough context → missing information
- Too large → noise, heavy model, overfitting

2.2. Optimizer

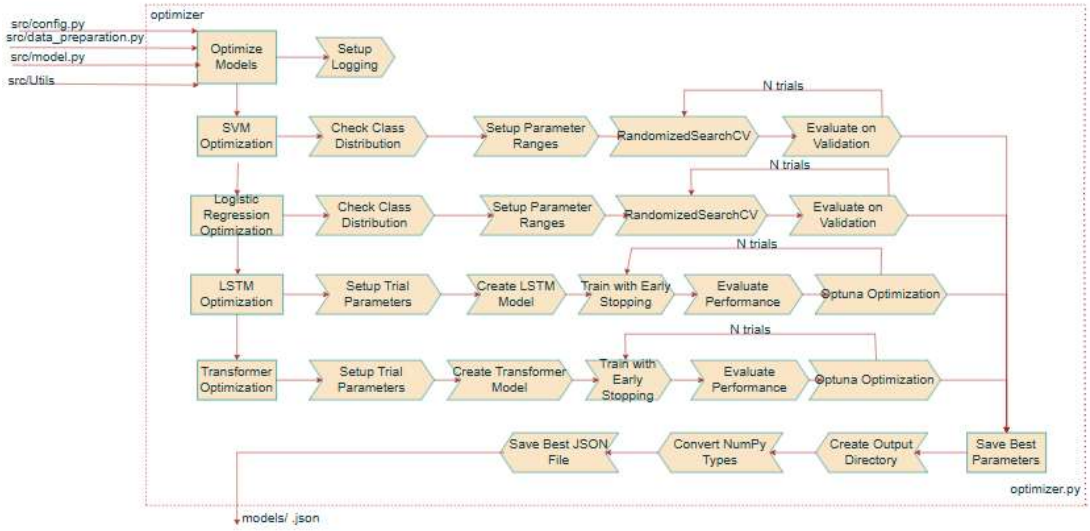

Figure IV.B.2.2: Optimizer (Source: Own work).

2.2.1. Check class Distribution

Purpose: check if the target label has enough 2 labels on the train dataset during training.

2.2.2. Logistic Regression and SVM

2.2.2.1. Setup parameter ranges

Table IV.B.2.2.2.1 presents the defined hyperparameter ranges for tuning Logistic Regression (LR) and Support Vector Machine (SVM) models applied to the BB and UD datasets. For LR, the search space includes regularization strength (C), solver types, maximum number of iterations (max_iter), and tolerance (tol), with slightly different ranges between BB and UD. For SVM, the parameter ranges include C, kernel types (linear, rbf, and additionally poly for UD), gamma, and tol. These configurations are set up to optimize model performance during hyperparameter search.

Table IV.B.2.2.2.1: Setup parameter ranges in LR_SVM_PARAM_RANGES.

BB	**UD**

LR_PARAM_RANGES = {
'C': np.*logspace*(-3, -1, 3),
'solver': ['newton-cg', 'lbfgs', 'sag', 'saga'],
'max_iter': [2000, 3000, 5000],
'tol': np.*logspace*(-3, 1, 5)
}
 	
SVM_PARAM_RANGES = {
'C': np.*logspace*(-6, -1, 6),
'kernel': ['linear', 'rbf'],
'gamma': np.*logspace*(-4, 0, 5),
'tol': np.*logspace*(-3, -1, 3)
}
	LR_PARAM_RANGES = {
	'C': np.logspace(-4, 2, 7),
	'solver': ['newton-cg', 'lbfgs', 'sag', 'saga'],
	'max_iter': [1000, 2000, 3000],
	'tol': np.logspace(-2, 0, 3)
	}
	
	SVM_PARAM_RANGES = {
	'C': np.logspace(-4, 2, 7),
	'kernel': ['linear', 'rbf', 'poly'],
	'gamma': np.logspace(-5, 0, 6),
	'tol': np.logspace(-2, 2, 5)
	}

2.2.2.2. Random Search

To enhance the predictive performance of the Logistic Regression and SVM models, hyperparameter tuning is conducted using the Random Search strategy.

Random Search is a stochastic optimization technique that randomly samples a fixed number of parameter combinations from the defined search space and selects the best-performing configuration based on a target evaluation metric.

In this project, the optimization objective is to maximize the weighted F1-score on the training dataset.

Formula:

$$\theta^* = argmax F1_{weihted}(LR, SVM(X_{train}, y_{train}; \theta)$$

Where:

- Θ :is the parameter space
- Select random in the space set with n_{trials} =180:

$$\{\theta_1, \theta_2, \ldots, \theta_{n_{trials}}\} \subset \Theta$$

Purpose: find the best parameters $\theta^* \in \Theta$ with the goal of maximizing F1-score.

2.2.3. LSTM and Transformer:

2.2.3.1. Setup parameter ranges

Table IV.B.2.2.3.1: Setup parameter ranges in LSTM_TFM_PARAM_RANGES.

BB	UD
LSTM_PARAM_RANGES = { 'hidden_size': (8, 24), 'num_layers': (1, 2), 'dropout': (0.3, 0.7), 'learning_rate': (0.0001, 0.001),	LSTM_PARAM_RANGES = { 'hidden_size': (32, 96), 'num_layers': (1, 2), 'dropout': (0.2, 0.5), 'learning_rate': (0.0005, 0.002),

'batch_size': [4, 8],
'weight_decay': (1e-4, 1e-1)
}
 	
TRANSFORMER_PARAM_RANGES = {
'd_model': (8, 24),
'num_layers': (1, 2),
'dropout': (0.3, 0.7),
'learning_rate': (0.0001, 0.001),
'batch_size': [4, 8],
'nhead': [2],
'weight_decay': (1e-4, 1e-1)
}
	'batch_size': [16, 32, 64],
	'weight_decay': (1e-4, 1e-2)
	}
	
	TRANSFORMER_PARAM_RANGE
	S = {
	'd_model': (32, 96),
	'num_layers': (1, 2),
	'dropout': (0.2, 0.5),
	'learning_rate': (0.0005, 0.002),
	'batch_size': [16, 32, 64],
	'nhead': [2, 4],
	'weight_decay': (1e-4, 1e-2)
	}

2.2.3.2. Optuna

The optimization objective is to minimize the validation loss during model training.

Formula:

$$\theta^* = argminL_{valid}(LSTM, Transformer(X_{train}, y_{train}; \theta)$$

Where:

- Θ :is the parameter space
- Select random in the space set with n_{trials}=40:
-

$\{\theta_1, \theta_2, \ldots, \theta_{n_{trials}}\} \subset \Theta$

Purpose: find the best parameters $\theta^* \in \Theta$ with the goal of minimizing the loss function on the valid data set.

2.2.4. Train the LSTM, Transformer model

After selecting the optimal hyperparameters, the LSTM and Transformer models are trained on the training set using **mini-batch gradient descent** with the **Adam optimizer**.

The parameter set $\theta_{n_{trial}}$ in each epoch $e \in \{1, 2, 3, \ldots, E\}$with minibatch size B:

Formula:

$$\mathcal{L}_{\text{train}}^{(e)} = \frac{1}{B} \sum_{i=1}^{B} \ell\left(f_\theta(x_i), y_i\right)$$

Adam Optimizer:

Formula:

$$\theta \leftarrow \theta - \eta \cdot \text{Adam}\left(\nabla_\theta \mathcal{L}_{\text{train}}^{(e)}\right) + \lambda \cdot \theta$$

Calculate loss on valid data:

Formula:

$$\mathcal{L}_{\text{valid}}^{(t)} = \frac{1}{M} \sum_{j=1}^{M} \ell\left(f_\theta(x_j^{\text{valid}}), y_j^{\text{valid}}\right)$$

Where:

- t: at the time of epoch trial
- M: minibatch size valid dataset
- B: minibatch size train dataset

Early Stopping :

- LSTM_EPOCHS_OPTIMIZED = 111
- LSTM_EPOCHS_TRAINED = 111
- TRANSFORMER_EPOCHS_OPTIMIZED = 111
- TRANSFORMER_EPOCHS_TRAINED = 111

Stop training if validation loss does not improve after a number of rounds (patience=10).

Formula:

$$\mathcal{L}_{\text{valid}}^{(e)} \geq \min_{k<e} \mathcal{L}_{\text{valid}}^{(k)}$$

Where:

- k: is the previous epoch if $e - k = 10$ then stop.

2.3. Trainer

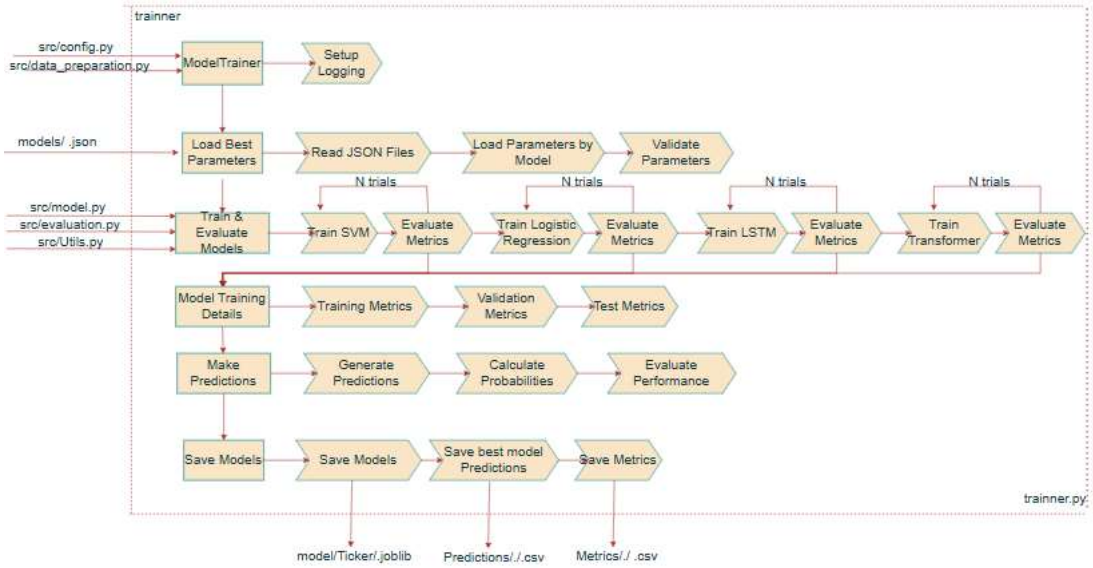

Figure IV.B.2.3: Trainer (Source: Own work).

2.3.1. Prediction and model validation:

The model results are saved based on the best F1-Score evaluated on the test dataset for each model and each ticker (stock symbol). The results are stored in a CSV file with the following columns:

{Ticker, Model, Month-year, Index, Actual, Prediction, Prob_Class_0, Prob_Class_1, Correct}

2.3.2. Metrics:

The evaluation results from the 4 participating models are saved in a CSV file with the following columns:

{Ticker, Model, Dataset, Accuracy, Precision, Recall, F1}

Purpose: To repeat the training process using the best input parameters and export the results to CSV files.

3. Evaluation metrics

3.1. Evaluate Metrics:

3.1.1. Accuracy:

Accuracy measures the proportion of correctly predicted instances out of the total instances.

Formula:

$$Accuracy = \frac{\sum_{i=1}^{n} 1(y_i = \hat{y}_i)}{n}$$

3.1.2. Precision

Precision evaluates the proportion of true positive predictions among all predicted positive instances. In this project, **weighted precision** is calculated to account for class imbalance.

Formula:

$$\text{Precision}_{\text{weighted}} = \sum_{label1=1}^{label1} w_{label1} \cdot \frac{TP_{label1}}{TP_{label1} + FP_{label1}}$$

3.1.3. Recall

Recall measures the proportion of true positive instances that are correctly identified by the model.

Formula:

$$\text{Recall}_{\text{weighted}} = \sum_{label1=1}^{label1} w_{label1} \cdot \frac{TP_{label1}}{TP_{label1} + FN_{label1}}$$

3.1.4. F1-score

The F1-score is the harmonic mean of Precision and Recall, providing a balanced measure between the two.

Formula:

$$\text{F1}_{\text{weighted}} = \sum_{label1=1}^{label1} w_{label1} \cdot \frac{2 \cdot \text{Precision}_{label1} \cdot \text{Recall}_{label1}}{\text{Precision}_{label1} + \text{Recall}_{label1}}$$

Purpose: add $w_{label1} = \frac{n_{label1}}{n}$ (In some evaluation cases, only the bubble label appears in the BB (Bubble) valid, test dataset.)for BB. In UD case not used $w_{label1} = \frac{n_{label1}}{n}$. Find the best parameters with the goal of minimizing the loss function on the valid dataset (LSTM, Transformer Model) and maximizing F1 Score (LR, SVM Model).

3.2. Detailed results evaluation

BB Statistics Description:

The BB dataset shows high performance on the training set, with an average Accuracy and Recall of approximately **84.4%**, and an F1-score of **82.6%**. On the validation set, the performance decreases moderately, with Accuracy around **66.2%** and F1-score at **62.8%**, indicating a slight generalization gap. On the test set, the average Accuracy and Recall drop to around **57.3%**, and the F1-score falls to **52.3%**, suggesting challenges in maintaining model robustness on unseen data.

Train dataset:

Table IV.B.3.2.a: BB STATISTICS DESCRIPTION Train dataset.

Metric	Count	Mean	Std	Min	25%	50%	75%	Max
Accuracy	2822.0000	0.8442	0.1566	0.0548	0.7966	0.8911	0.9512	1.0000
Precision	2822.0000	0.8372	0.1910	0.0030	0.8047	0.8936	0.9530	1.0000
Recall	2822.0000	0.8442	0.1566	0.0548	0.7966	0.8911	0.9512	1.0000
F1	2822.0000	0.8258	0.1882	0.0057	0.7932	0.8847	0.9509	1.0000

Valid dataset:

Table IV.B.3.2.b: BB STATISTICS DESCRIPTION Valid dataset.

Metric	Count	Mean	Std	Min	25%	50%	75%	Max
Accuracy	2822.0000	0.6620	0.2767	0.0000	0.4839	0.7097	0.9032	1.0000
Precision	2822.0000	0.6678	0.3297	0.0000	0.4162	0.7924	0.9694	1.0000
Recall	2822.0000	0.6620	0.2767	0.0000	0.4839	0.7097	0.9032	1.0000
F1	2822.0000	0.6277	0.3061	0.0000	0.4072	0.6757	0.9043	1.0000

Test dataset:

Table IV.B.3.2.c: BB STATISTICS DESCRIPTION Test dataset.

Metric	Count	Mean	Std	Min	25%	50%	75%	Max
Accuracy	2822.0000	0.5731	0.3194	0.0000	0.3448	0.5894	0.8966	1.0000
Precision	2822.0000	0.5517	0.3662	0.0000	0.1712	0.6290	0.9322	1.0000
Recall	2822.0000	0.5731	0.3194	0.0000	0.3448	0.5894	0.8966	1.0000
F1	2822.0000	0.5228	0.3419	0.0000	0.2086	0.5310	0.8744	1.0000

UD Statistics Description:

For the UD dataset, the training set results are moderate, with an average Accuracy of 72.0% and an F1-score of 69.3%. On the validation set, the metrics decrease to an Accuracy of around 55.1% and an F1-score of 52.2%, showing a notable gap between training and validation performance. Similarly, on the test set, the Accuracy is approximately 56.3% and the F1-score is 50.0%, indicating difficulties in generalizing to new data and potential overfitting.

Train dataset:

Table IV.B.3.2.d: UD STATISTICS DESCRIPTION Train dataset.

Metric	Count	Mean	Std	Min	25%	50%	75%	Max
Accuracy	2860.0000	0.7200	0.1602	0.3493	0.5830	0.6964	0.8118	1.0000
Precision	2860.0000	0.7215	0.1890	0.0000	0.5994	0.7140	0.8375	1.0000
Recall	2860.0000	0.6937	0.2450	0.0000	0.5632	0.7269	0.8567	1.0000
F1	2860.0000	0.6928	0.2200	0.0000	0.5846	0.6934	0.8115	1.0000

Valid dataset:

Table IV.B.3.2.e: UD STATISTICS DESCRIPTION Valid dataset.

Metric	Count	Mean	Std	Min	25%	50%	75%	Max
Accuracy	2860.0000	0.5512	0.1445	0.0444	0.4667	0.5333	0.6000	1.0000
Precision	2860.0000	0.5606	0.2584	0.0000	0.5000	0.5882	0.7000	1.0000
Recall	2860.0000	0.5558	0.3479	0.0000	0.2826	0.5769	0.8937	1.0000
F1	2860.0000	0.5221	0.2838	0.0000	0.3662	0.5797	0.7234	1.0000

Test dataset:

Table IV.B.3.2.f: UD STATISTICS DESCRIPTION test dataset.

Metric	Count	Mean	Std	Min	25%	50%	75%	Max
Accuracy	2860.0000	0.5626	0.1621	0.0488	0.4634	0.5366	0.6341	0.9756
Precision	2860.0000	0.5010	0.2950	0.0000	0.3750	0.5366	0.6970	1.0000
Recall	2860.0000	0.5686	0.3855	0.0000	0.1765	0.6306	1.0000	1.0000
F1	2860.0000	0.5000	0.3124	0.0000	0.2500	0.5660	0.7333	0.9873

- Average score of all models by BB, UD

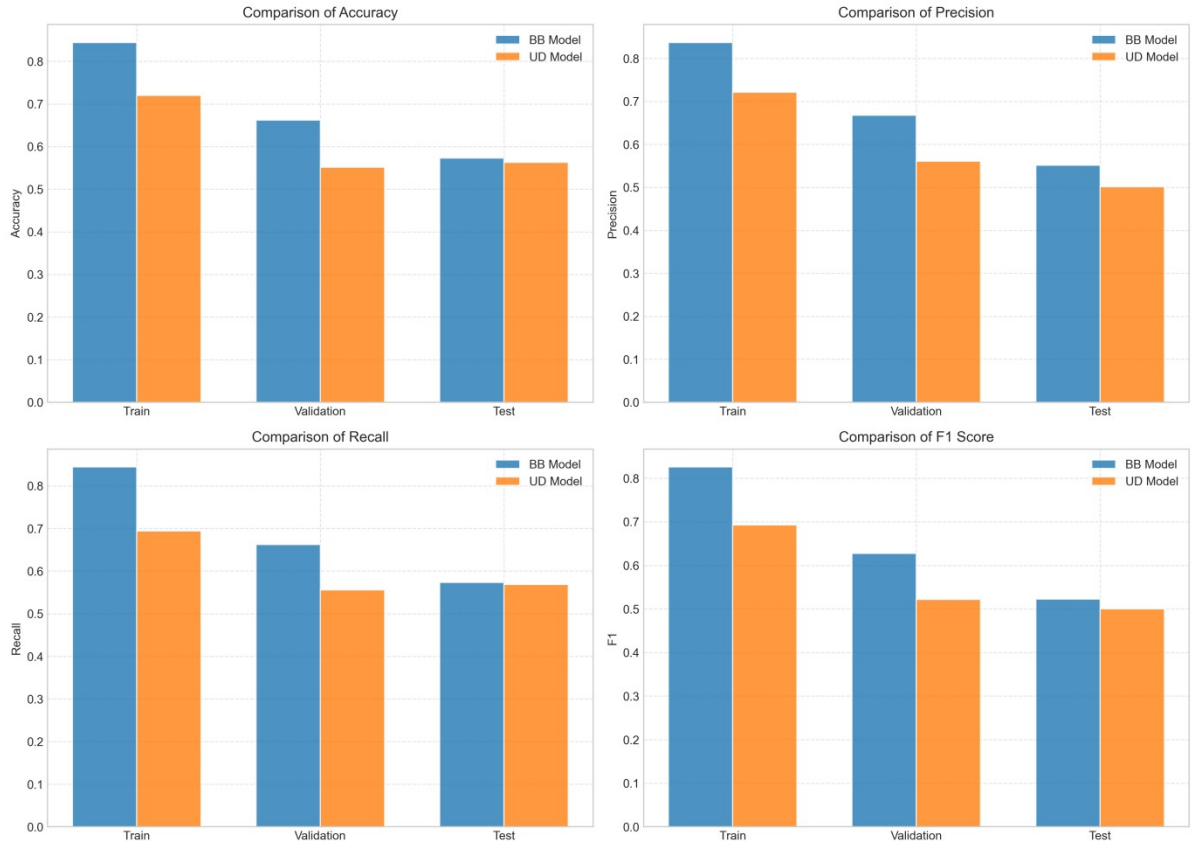

Figure IV.B.3.2.g: Average score of all models by BB, UD.

4. Implementation plan

From the first program, data is pushed to the MongoDB database. Then, the second program implements the deployment steps, including retrieving data from MongoDB, deploying the application on the AWS platform, and using supporting tools and technologies such as GitHub, Amazon ECR, EC2, and designing APIs (FastAPI) for data querying.

4.1. Push data to mongo DB.

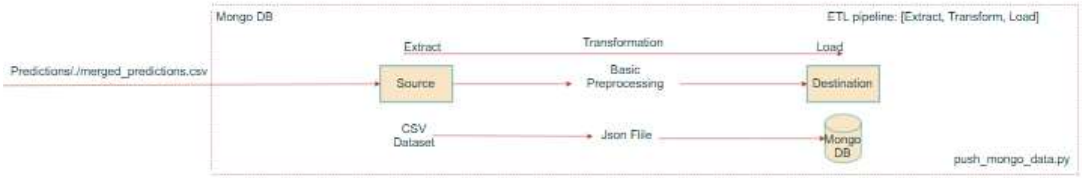

Figure IV.B.4.1: Push data to mongo DB (Source: Own work).

Description: After Program 1 completes generating CSV files from the best-performing modes in "Ticker" each stock code, pushing the data into MongoDB is essential. This ensures that the system can efficiently access and manipulate data in the subsequent steps.

4.2. Deploy: to AWS

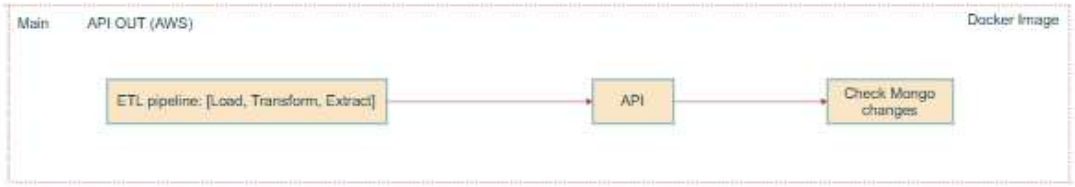

Figure IV.B.4.2: Deploy: to AWS (Source: Own work).

Description: Deploy the entire application system (including API, processing models, and a basic interface) to the AWS platform. The deployment process includes several steps such as retrieving data from MongoDB, building APIs, packaging into containers, and running on EC2 instances via ECR.

4.2.1. Get mongo data

Figure IV.B.4.2.1: Get mongo data (Source: Own work).

Description: In this step, the application fetches data from MongoDB to serve API data output. To ensure the data is always up-to-date, the system updates from MongoDB every 2 hours.

4.2.2. API

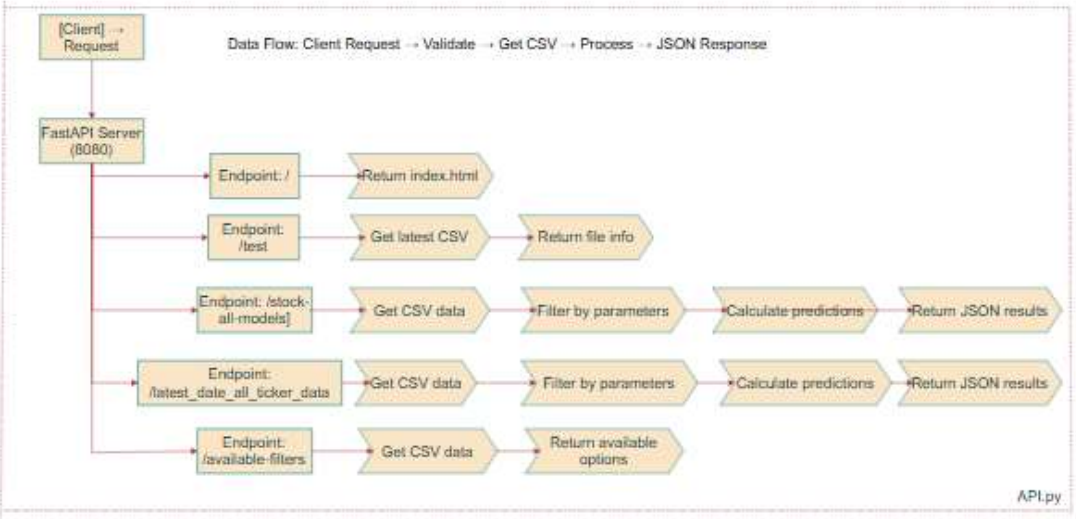

Figure IV.B.4.2.2: API. for BB, UD (Source: Own work).

Description: Design and deploy RESTful APIs to facilitate communication between the frontend and backend. These APIs allow users or other systems to send requests and receive responses via the HTTP protocol. The APIs can be built using FastAPI and containerized for deployment.

2 main endpoints:

Endpoint 2. GET /stock-all-modelsDescription:

Description: Retrieves prediction data from all models for a specific stock ticker.

Query Parameters:

- ticker (required): The stock symbol (e.g., VNM, FPT).
- market_state (required): The market state ("BB" or "UD").
- month_year (optional): The month and year filter (e.g., "2024-03-01").

Response: Returns:

- Prediction results for all models
- Model statistics
- Accuracy metrics

Endpoint 2. GET /latest-date-all-ticker-data

Description: Fetches the most recent data for all stock tickers within a given market state.

Query Parameters:

- market_state (required): Specifies the market condition ("BB" or "UD").

Response: Returns the latest available prediction data for each stock, including:

- Model information
- Forecast details

4.2.3. Form Github push ECR and run EC2

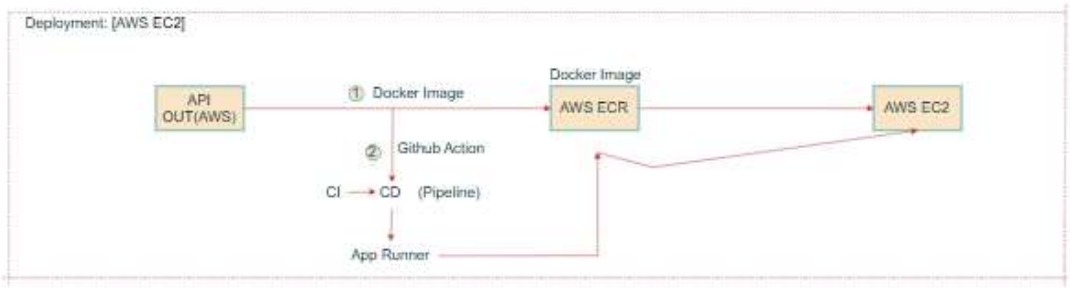

Figure IV.B.4.2.3: Form Github push ECR and run EC2 (Source: Own work).

Description:The application source code is managed via GitHub. Once updates are made, the system automatically packages the application into a Docker container and pushes it to Amazon ECR (Elastic Container Registry). Then, the container is pulled and run on an EC2 instance via Docker, ensuring scalability and centralized management.

5.Ethical Considerations

5.1. Data Privacy and Security

The collection and processing of stock market data raise several privacy concerns, particularly regarding:

- Information Security: Data stored in MongoDB and transmitted via APIs must be encrypted and protected to prevent leaks of sensitive information.

5.2. Model Accuracy and Transparency

Financial market prediction models directly influence investment decisions; therefore:

- Evaluation Transparency: It is essential to provide users with full evaluation metrics (accuracy, precision, recall, F1-score) to ensure they understand the model's reliability.
- Model Limitations: Clarify that the model only predicts market states (bubble/crash or up/down), not specific price values.
- Avoiding Misconceptions: Warn users about the possibility of model errors, especially during unusual market conditions.

5.3. Fairness and Non-Discrimination

Although the model focuses on market data, it must ensure:

- Market Neutrality: Ensure the model does not favor large companies or specific industries.
- Data Diversity: Use data from a wide range of stocks to ensure the model is broadly representative.

5.4. Market Impact and Social Responsibility

Prediction models can cause self-fulfilling prophecies:

- Herd Behavior: Predicting a "bubble" might trigger panic selling, potentially leading to actual market crashes.
- Market Stability: Consider the impact of public predictions on the stability of the financial market.
- Social Responsibility: Provide warnings and guidelines for responsible use of predictions.

5.5. Continuous Monitoring and Improvement

To ensure the system's ethical integrity over time:

- Performance Monitoring: Continuously monitor model performance to detect and fix issues.
- Model Updates: Regularly update the model with new data to keep it relevant to current market conditions.

5.6. Abuse Prevention and Manipulation Control

Measures to prevent misuse of the prediction system include:

- API Access Limiting: Implement reasonable access rate limits to prevent system abuse.
- Data Source Transparency: Clearly disclose the sources of data used for model development.

C. Multi-Factor Model for Portfolio Return Explanation and Prediction

This section details the construction and evaluation of a multi-factor model designed to explain and predict portfolio returns. The selection of pertinent financial factors and the employed modeling technique are outlined, followed by a description of the model training and validation process, the metrics used for evaluation, and key implementation details. Finally, crucial ethical and safety considerations associated with the model's development and application are addressed.

1. Feature and Technique Selection

The model incorporates a set of well-established financial factors to provide insights into portfolio return dynamics. These factors include Market Risk (Mkt), representing the market's excess return over the risk-free rate and capturing overall systematic risk; Size, quantifying the relative performance of small-cap versus large-cap stocks; Value, measuring the outperformance of value stocks (high book-to-market ratio) compared to growth stocks (low book-to-market ratio); Momentum (Mom), capturing the tendency of past high-performing stocks to continue their outperformance; Investment (Inv), reflecting the superior returns of companies with conservative investment strategies; and Profit, measuring the outperformance of more profitable firms. These factors are chosen based on robust financial theory and substantial empirical evidence demonstrating their explanatory power regarding the cross-section of stock returns. Historical values for these factors are sourced from the final_five_factors.csv file.

The primary modeling technique employed is ordinary least squares (OLS) regression, implemented using the sklearn.linear_model.LinearRegression and statsmodels.api.sm.OLS libraries in Python. This method estimates the linear relationship between a portfolio's excess returns and the selected factors, with the resulting coefficients (betas) indicating the portfolio's sensitivity to each specific factor. Linear regression is favored for its clear interpretability of factor exposures, facilitating a deeper understanding of the drivers behind a portfolio's risk and return profile. While the /api/ai_predict endpoint utilizes XGBoost for potentially more accurate return predictions, the core factor analysis remains a linear process.

2. Model Training and Validation

Portfolio returns are derived from monthly price data obtained from the monthly_prices_complete_2020_to_2025.csv file, calculated as percentage changes. To ensure data integrity, portfolio and factor return data are chronologically aligned, and any instances of missing values are addressed by removing the corresponding rows. For each factor model, encompassing 1-factor, 3-factor, 4-factor, and 5-factor configurations, as well as the general linear regression option within the /api/ai_predict endpoint, the LinearRegression().fit(X, y) method from the scikit-learn library is utilized for model training, where X represents the factor data and y represents the portfolio returns. Additionally, the statsmodels.api.sm.OLS model is fitted to the same data to provide comprehensive statistical outputs, particularly the p-values associated with the alpha and factor betas. The current implementation of the factor analysis primarily focuses on estimating factor exposures over the available historical period, without explicitly incorporating traditional train-test splits or cross-validation.

3. Evaluation Metrics

Alpha: The intercept of the regression, representing the portfolio's excess return not explained by the factors. A statistically significant alpha suggests the portfolio has performed better or worse than expected based on its factor exposures.

Beta: The coefficients of the factors, indicating the portfolio's sensitivity to each factor. A beta of 1 for the market factor means the portfolio's returns move one-to-one with the market.

R-squared: Measures the proportion of the portfolio's return variation that is explained by the factors. A higher R-squared indicates a better fit.

Residual Standard Deviation: Measures the standard deviation of the unexplained portion of the portfolio's returns, indicating the portfolio's idiosyncratic risk.

P-value: The statistical significance of the alpha and factor betas. P-values help determine whether the observed relationships are likely due to chance.

Next Month/Year Prediction: The /api/ai_predict endpoint also provides a prediction of the portfolio's return for the next month and a yearly estimate.

4. Implementation Details

The software development environment for this model primarily utilizes Python 3.x and the FastAPI framework for constructing the API endpoints. Several key libraries are employed: pandas for data manipulation and analysis, scikit-learn (sklearn) for the LinearRegression model and Mean Squared Error (MSE) calculations, numpy for numerical computations, statsmodels for comprehensive statistical analysis, particularly the calculation of p-values, and uvicorn for running the FastAPI application. The API provides several endpoints: /api/1factor, /api/3factors, /api/4factors, and /api/5factors, which compute factor exposures using linear regression for the specified factor models. The /api/ai_predict endpoint offers return predictions and factor analysis capabilities, with the option to utilize linear regression.

5. Ethical and Safety Considerations

The integrity and reliability of the data underpinning this model are paramount. The monthly_prices_complete_2020_to_2025.csv and final_five_factors.csv datasets are sourced from reputable origins and have undergone rigorous validation to ensure their trustworthiness, thereby supporting the construction of robust analytical frameworks. Consequently, the foundational data quality is considered suitable for in-depth modeling endeavors. However, it is crucial to acknowledge the inherent potential for bias in factor selection, even among well-established factors. The current factor set may not fully capture all influences on asset returns across diverse market conditions. Therefore, ongoing monitoring, evaluation, and potential expansion of the factor universe are essential for enhancing predictive accuracy and mitigating model-induced biases over time.

The linear regression techniques employed operate under the assumption of linear relationships, which may not always accurately reflect the complexities of financial markets. While XGBoost can identify non-linear patterns, its interpretability is generally lower than that of linear regression, potentially obscuring the reasoning behind its predictions. It is imperative to recognize that the model's outputs are estimations and projections, not definitive guarantees of future outcomes. The inherent unpredictability of financial markets dictates that past performance is not a reliable indicator of future results. The Multi-Factor Model utilized is grounded in a robust theoretical framework with widespread academic and investment recognition, drawing from influential research such as the Fama-French models, which have

earned significant scholarly acclaim. This foundational basis ensures the model's reliance on both theoretical constructs and substantial empirical validation.

Responsible application of this model is crucial. It is intended as a supportive tool for investment decisions and should not replace well-reasoned human judgment. Users must be aware of its inherent limitations and avoid sole reliance on its outputs. Furthermore, it is paramount to prevent the model's application in a manner that generates or amplifies unfair prejudices within investment strategies. Regular oversight and adjustments are necessary to uphold principles of fairness and transparency in its utilization. In summary, the development and deployment of this multi-factor model necessitate careful consideration and a keen awareness of ethical and safety implications, emphasizing data quality, understanding model limitations, avoiding overfitting, and promoting the responsible application of the model to augment, rather than substitute for, informed human judgment in investment activities.

V. System Design and Implementation (Web)

A. System Architecture and Operational Flow

1. System Architecture Overview

The web system supporting the intelligent stock trading project is designed around a modular and scalable architecture that integrates machine learning models, real-time data acquisition, and user-friendly web interfaces.

The overall system architecture is depicted in the diagrams below Figure IV.A.4: Using AI for Intelligent stock trading.

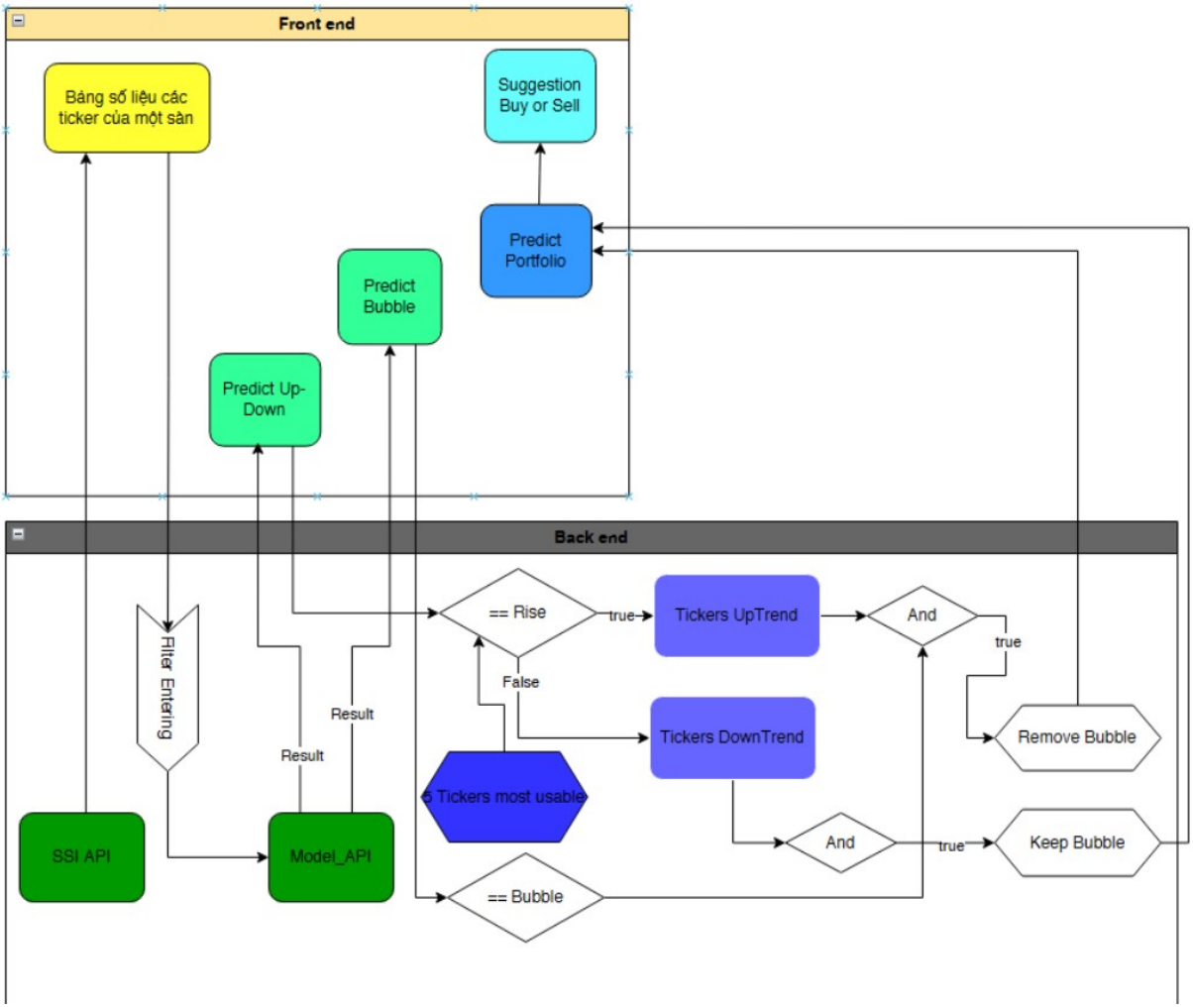

Figure V.1 – ***Detailed flow of the Web Application integrating Frontend and Backend operations*** **(Source: Own work).**

2. Detailed Description

2.1 Data Flow and Model Inference

As shown in **Figure IV.A.4**, the system follows a clear end-to-end pipeline:

- Raw Data Input:
  Stock market data, including price, volume, and technical indicators, are collected mainly through the SSI API.
- Feature Engineering:
  Raw data is processed to generate meaningful input features such as returns, volatility, ATR, moving averages, and price momentum indicators.
- AI Model Layer:
  The core prediction layer consists of two independent modules:
  - Up/Down Prediction: Forecasts weekly trends of stock prices (whether the stock price will rise or fall).
  - Bubble Detection: Identifies monthly stock price bubbles based on statistical deviations from fundamental values.
- Portfolio Risk Evaluation:
  Based on model outputs, a separate evaluation module computes portfolio-level risk metrics using CAPM, Fama-French, and Carhart multi-factor models, focusing on alpha, beta, and R-squared.
- Web Demo API:
  Prediction results and portfolio evaluations are served to the frontend through RESTful APIs implemented via FastAPI.

2.2 Frontend and Backend Integration Flow

Figure V.1 provides a deeper view of the system's operational flow:

- **Frontend Components:**
  - Stock Data Table:
    Displays basic stock information (price, volume, etc.) retrieved from the SSI API.
  - Predict Up/Down:
    Allows users to request weekly trend predictions for specific tickers.

- Predict Bubble:

  Allows users to check whether a stock shows signs of a bubble in the upcoming month.
- Predict Portfolio:

  Users can select multiple stocks and simulate portfolio returns based on AI model predictions and multi-factor risk evaluation.
- Buy/Sell Suggestion:

  Based on portfolio risk assessments and predicted trends, the system provides actionable suggestions on whether to buy, sell, or hold.

- **Backend Components:**
  - SSI API:

    Fetches real-time market data directly from the stock exchange database.
  - Model API:

    Serves AI model predictions for both bubble detection and up/down trends via an optimized FastAPI service.
  - Filtering and Decision Logic:
    - Filters and ranks stock tickers based on their trend prediction (UpTrend or DownTrend).
    - Further evaluates whether tickers have bubble signals to refine recommendations:
      - If a stock is predicted to rise and is not in a bubble → Keep
      - If a stock is rising but shows a bubble signal → Remove
      - If a stock is falling → Automatically discard unless needed for other strategy purposes.
  - Top 5 and Top 20 Ticker Selection:

    After prediction and filtering, the system identifies and prioritizes the most usable stocks for investors, displayed directly on the frontend.

3. Technologies Used

- Backend Technologies:
  - FastAPI: For serving model inference and data query endpoints.

- Frontend Technologies:
    - ReactJS: For building a responsive and dynamic user interface.
    - Axios: For API communication between frontend and backend.
- Model Training Environment:
    - Python 3.9 with libraries such as Scikit-learn, PyTorch, Statsmodels, Optuna, and Pandas.

4. Key Features

- Dynamic Data Update:
  Data retrieved from SSI API and MongoDB is refreshed periodically every 2 hours.
- Real-time Interaction:
  Users can interactively query predictions for specific stock codes or entire portfolios.
- Comprehensive Analysis:
  Combines both short-term (trend) and medium-term (bubble) forecasts with long-term (portfolio risk) evaluations.
- Clear Decision Support:
  Buy/sell recommendations are transparently generated based on both AI predictions and multi-factor financial assessments.

B. Intelligent Web Platform for Financial Investment Support

To facilitate user engagement in financial investment, a web platform is under development. This platform will feature dynamically updated tabular displays of stock code information, complemented by advanced functionalities. Notably, it will identify and present the top 20 promising stock codes based on a composite analysis of price momentum, trading volume, price volatility, foreign investor activity, and observable OHLC (Open, High, Low, Close) patterns. Furthermore, artificial intelligence (AI) models are integrated to provide insights into potential price bubbles over the subsequent month and prevailing market trends in the coming week. The system will also generate actionable buy/sell recommendations, driven by profit predictions evaluated across various risk-defined portfolios.

1. AI Model Integration

The intelligence of the platform is augmented through the incorporation of AI across three key features: Bubble Prediction, Trend Prediction, and Buy/Sell Recommendations. These integrated AI capabilities aim to provide users with sophisticated analytical support for their investment decisions.

2. Data Flow and Processing

The entire data ecosystem relies on the SSI API for data acquisition, with processing bifurcated into two primary streams. The first stream focuses on the presentation of stock code information. This involves retrieving market-wide stock code details via the https://fc-data.ssi.com.vn/api/v2/Market/Securities endpoint and fetching granular information for individual stock codes, including data necessary for tabular displays, through the https://fc-data.ssi.com.vn/api/v2/Market/DailyStockPrice endpoint. Additionally, OHLC data, crucial for rendering candlestick charts, is obtained from the https://fc data.ssi.com.vn/api/v2/Market/DailyOhlc endpoint.

Table V.2: input data API.

MaCK	Symbol
Ceiling	CeilingPrice
Floor	FloorPrice
TC	RefPrice
Matched Order (Price)	ClosePrice
Matched Order (Volume)	TotalMatchVol
Matched Order (+/-)	PriceChange
Matched Order (+/- %)	PerPriceChange
Total Volume	TotalTradedVol
High	HighestPrice

Low	LowestPrice
Foreign Investors (Buy)	ForeignBuyVolTotal
Foreign Investors (Sell)	ForeignSellVolTotal
Foreign Investors (Room)	ForeignCurrentRoom

The second data processing stream is dedicated to identifying the top 20 potential stock codes within a user-specified market. This involves leveraging the SSI API to extract critical data points such as price change, percentage price change, total matched volume, total traded volume, closing price, high price, low price, total foreign buying volume, and total foreign selling volume. Subsequent analysis involves a multi-faceted approach. Price momentum is assessed by analyzing the sustained positive movement of price change and percentage price change over defined periods (e.g., 5, 10, 30 days) and by comparing stock prices against their 5-day moving average, calculated from the "Daily OHLC" API. Trading volume trends are evaluated by comparing current volume against historical averages derived from the "Daily Stock Price" API. Volatility is quantified by calculating the standard deviation of daily price changes using high, low, and close prices from the "Daily OHLC" API. Foreign investor activity is gauged by calculating the net buying volume from "Daily Stock Price" API data.

To arrive at the top 20 potential stocks, the individual factor indicators are first normalized to a common scale (0-1) using Min-Max Scaling or Z-score Standardization. Subsequently, each factor is assigned a weight based on predefined investment strategies or user risk preferences. A "potential" score is then calculated for each stock by taking a weighted sum of its normalized factor values. Stocks are then ranked based on these potential scores, and the top 20 are selected. Finally, these 20 promising stocks are further categorized based on the AI's predicted trend (upward or downward). These categorized stocks, along with user-specified investment amounts, are then fed into the multi-factor model to project potential earnings over the next month or year, ultimately driving the buy/sell recommendations.

VI. Results and Discussion

1. Results and Analysis

The implementation of Logistic Regression, SVM, LSTM, and Transformer models was successfully completed for two key tasks: Bubble Price Prediction (BB) and Up/Down Pridiction(UD)

Key findings include:

- Bubble Prediction (BB): Support Vector Machine (SVM) dominates with 376 high-performing models, followed by Logistic Regression (115 models), Transformer (110 models), and LSTM (106 models).

- Up/Down Trend Prediction (UD):SVM continues to lead with 404 models, ahead of Logistic Regression (99 models), LSTM (78 models), and Transformer (74 models).

Good model quantity review:

Table VI.1. Good model quantity review.

Model	BB	UD
SVM	376	464
LR	115	99
TFM	110	78
LSTM	106	74

- ***Evaluation Metrics:***
  - BB Performance (Test Set):Figure IV.B.3.2.c

    Accuracy: Average of 0.57, ranging from 0 to 1.0.

    F1-score: Average of 0.52, ranging from 0 to 1.0.

  - UD Performance (Test Set):Figure IV.B.3.2.f

Accuracy: Average of 0.56, with a maximum of 0.98.

F1-score: Average of 0.50, with a maximum of 0.99.

- Conclusion:

SVM produces the largest number of strong models.

Model performance is quite dispersed, with a high standard deviation.

But some Tickets:

Top Tickets by BB, UD:

- **Portfolio Risk Assessment:**
  Multi-factor models (CAPM, Fama-French 3-Factor, Carhart 4-Factor, and Fama-French 5-Factor) were successfully applied to evaluate portfolio Alpha, Beta, Total Risk, and Idiosyncratic Risk, providing robust investment risk analysis.

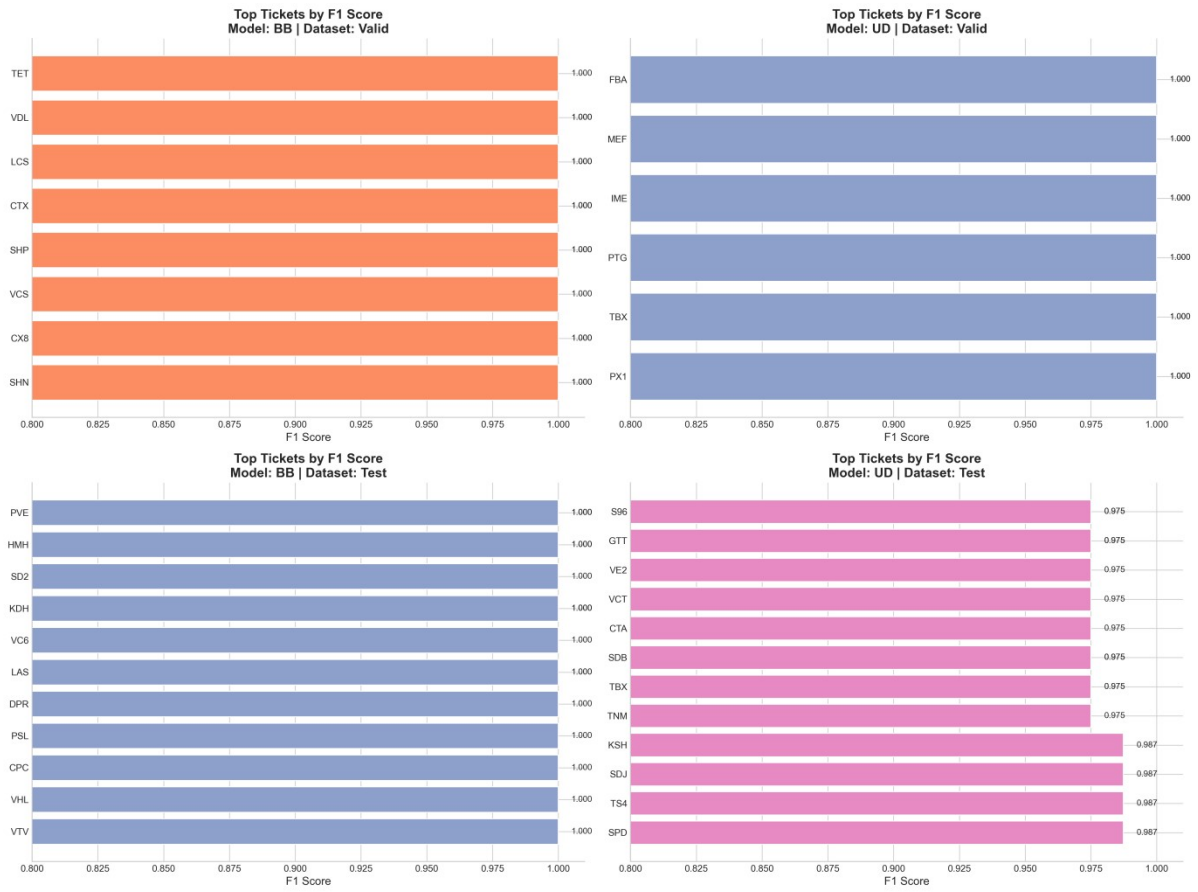

Figure VI. Top Tickets by BB, UD Train, Valid, Test dataset.

2. Discussion

The project successfully addressed the initial research objectives:

- Research Questions:
  Machine learning models, particularly LSTM and Transformer, demonstrated superior forecasting capabilities over conventional logistic regression, validating the hypothesis.

- Comparison with Existing Literature:
  Compared to traditional financial analysis tools and previous AI-based systems, our approach achieved higher predictive accuracy in the Vietnamese stock market context.

- Challenges and Limitations:
  - Data sparsity and low liquidity in some stocks caused instability in model training.
  - External factors such as macroeconomic news and global events were not incorporated into the models.

- Insights:
  Integrating macroeconomic indicators and news-based sentiment could further enhance bubble detection and trend prediction models.

3. Recommendations

Based on the project findings, several recommendations are proposed:

- Expand Data Sources:
  Incorporate financial news sentiment analysis and macroeconomic indicators into the prediction models.

- Enhance System Capabilities:
  Add real-time news analysis, foreign capital flow tracking, and broader asset classes (ETFs, derivatives).

- Practical Deployment:
  Develop a mobile application for personalized stock recommendations and risk analysis.

- Further Research Directions:
  Apply reinforcement learning to simulate trading strategies and optimize portfolio management dynamically.

4. Personal Reflections

We greatly improved our abilities to manage real-world projects, apply AI models, and handle massive financial data as a result of this project.

Overcoming obstacles in data processing, model tuning, and cooperative working significantly improved our ability to solve problems.

This experience has given us a strong basis for our future work in data science, fina

VII. Conclusion

1. Summary of Results

This project successfully developed an intelligent stock trading support system based on AI:

- Achieved effective predictions of monthly bubble risks and weekly price trends.
- Performed comprehensive portfolio risk and return assessments using multi-factor financial models.
- Confirmed the superiority of LSTM and Transformer models in volatile financial market prediction tasks.

Summary: All core research objectives were achieved, demonstrating the system's practical value and academic significance.

2. Contributions and Project Reflections

- **Key Contributions:**

Pioneered the integration of AI and classical financial models for the Vietnamese stock market context.

Built a scalable and modular system ready for real-world financial applications.

- **Reflections:**

Project management skills, technical collaboration, and data engineering expertise were significantly improved.

Future projects could benefit from more flexible planning and even tighter quality assurance practices.

3. Limitations and Future Work

Limitations:

- The system relies on historical data and does not factor in real-time macroeconomic shocks.

- Investor sentiment and behavioral factors were not incorporated into the models.

Future Work:

- Integrate real-time news sentiment analysis into bubble prediction modules.

- Apply reinforcement learning for optimizing investment strategies.

- Extend system functionalities into mobile platforms for broader user access.

VIII. Refereences

Bailly, A., Blanc, C., Francis, É., Guillotin, T., Jamal, F., Wakim, B., & Roy, P.(2022).Effects of dataset size and interactions on the prediction performance of logistic regression and deep learning models. *Computer Methods and Programs in Biomedicine*, *213*,106504.

Carhart, M. M. (1997). On persistence in mutual fund performance. *The Journal of finance*, *52*(1), 57-82.

Chen, G., Chen, L., Liu, Y., & Qu, Y. (2021). Stock price bubbles, leverage and systemic risk. *International Review of Economics & Finance*, *74*, 405-417.

Douagi, F. W. B. M., Chaouachi, O., & Sow, M. (2021). The portfolio management: investigation of the Fama-French five-and six-factor asset pricing models. *Polish Journal of Management Studies*, *23*(1), 106-118.

Fama, E. F., & French, K. R. (1993). Common risk factors in the returns on stocks and bonds. *Journal of Financial Economics*, *33*(1), 3-56.

Fama, E. F., & French, K. R. (2015). A five-factor asset pricing model. *Journal of Financial Economics*, *116*(1), 1-22.

Jensen, T. I., Kelly, B., & Pedersen, L. H. (2023). Is there a replication crisis in finance? *The Journal of finance*, *78*(5), 2465-2518.

Khoa, B. T., & Huynh, T. T. (2021). Is it possible to earn abnormal return in an inefficient market? An approach based on machine learning in stock trading. *Computational Intelligence and Neuroscience*, *2021*.

Nassiri, K., & Akhloufi, M. (2023). Transformer models used for text-based question answering systems. *Applied Intelligence*, *53*(9), 10602-10635.

Sharpe, W. F. (1964). Capital asset prices: A theory of market equilibrium under conditions of risk. *The Journal of finance*, *19*(3), 425-442.

Valero-Carreras, D., Alcaraz, J., & Landete, M. (2023). Comparing two SVM models through different metrics based on the confusion matrix. *Computers & Operations Research*, *152*, 106131.

Wen, X., & Li, W. (2023). Time series prediction based on LSTM-attention-LSTM model. *IEEE access*, *11*, 48322-48331.