Stock Prediction Using Different Machine Learning Techniques with The Dataset of NVIDIA

by

Ying Chen

Bachelor of Engineering, Microelectronics Science and Engineering, Shanghai Jianqiao University, Shanghai, China, 2018

A Project Submitted in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF ENGINEERING

in the Department of Electrical and Computer Engineering

Stock Prediction Using Different Machine Learning Techniques with The Dataset of NVIDIA

by

Ying Chen

Bachelor of Engineering, Microelectronics Science and Engineering, Shanghai Jianqiao University, Shanghai, China, 2018

Supervisory Committee

Dr. Amirali Baniasadi, Supervisor

(Department of Electrical and Computer Engineering)

Dr. Lin Cai, Departmental Member

(Department of Electrical and Computer Engineering)

Dr. Mihai Sima, Departmental Member

(Department of Electrical and Computer Engineering)

ABSTRACT

This project investigates the effectiveness of machine learning models in predicting NVIDIA Corporation's stock price movements. Four distinct ML approaches were implemented and compared: LR (Linear Regression), SVM (Support Vector Machines), NN (Neural Networks), and LSTM (Long Short-Term Memory networks). Using historical price data and technical indicators, model performance was evaluated through RMSE (Root Mean Squared Error) and R^2 (R-squared) metrics.

The results demonstrated that LR and SVM models performed better than complex deep learning architectures, achieving superior accuracy, with RMSE ≈ 3, $R^2 \approx 0.99$, while maintaining interpretability. The Neural Network model performed unexpectedly and poorly, with $R^2 = -0.50$, suggesting significant overfitting challenges. While the LSTM showed promise for capturing temporal dependencies, it required further optimization to achieve higher accuracy and compete with traditional methods.

In addition, the project incorporated the importance of ethical considerations, including bias mitigation strategies and regulatory compliance measures for financial AI applications. The critical balance between model complexity and practical utility was highlighted in stock market prediction, emphasizing that sophisticated architectures don't automatically guarantee better performance.

Keywords: Stock price prediction, machine learning, NVIDIA, algorithmic trading, financial forecasting, AI ethics

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

I am deeply grateful to my supervisor,

Dr. Amirali Baniasadi,

for his invaluable guidance, unwavering support, and profound expertise

that have shaped this work and my growth during the life in UVic.

To

Dr. Lin Cai and Dr. Mihai Sima,

I sincerely appreciate the invaluable feedback and unwavering patience

you have shown as my committee member and the Chair.

Ying Chen

DEDICATION

To the neighbor’s free-roaming cat,

whose quiet companionship soothed my restless hours.

To my globe-spanning friends,

whose voices bridged the night when time zones stretched between us.

To the little whale,

whose gentle nudges pointed the way when the path grew dim.

Gratitude, always.

Chapter 1

Introduction

1.1 Background

A share price is the price of a single share of several saleable equity shares of a company. In layman's terms, the stock price is the highest amount someone is willing to pay for the stock, or the lowest amount that it can be bought for.1 In Economics and Financial Theory, random walk techniques are used by experts and analysts to model behavior of asset prices, especially for the listed share prices from different companies publicly. Based on several research, some of the biggest price deviations from random walks result would be affected by seasonal and temporal patterns. For instance, returns in January exceed more significantly rather than those in other months (called January effect) as well as on stock prices on Mondays go down more than other days in a week. Over decades, these effects have noted in many different and complicated markets, but it is still hard to give a completely and definitely satisfied explanation on this persistence.

As a best-known professional GPUs, AI, and computer hardware technology enterprise, Nvidia Corporation always cause a huge amount of awareness. For a company, the stock price during a period can reflect many things and it is important to analysis for further investments. Technical experts and analysis apply several methods to extract information to predict future price movements based on historical data.

1.2 Problem Definition

Stock prediction involves forecasting future stock prices or market movements using historical data, statistical models, and machine learning techniques. The goal is to assist investors, traders, and financial analysts in making accurate decisions to maximize returns and minimize risks. The Objective is predicting future stock prices (regression problem), classify market movements (up/down) (classification problem) and generate trading signals (buy/sell/hold).

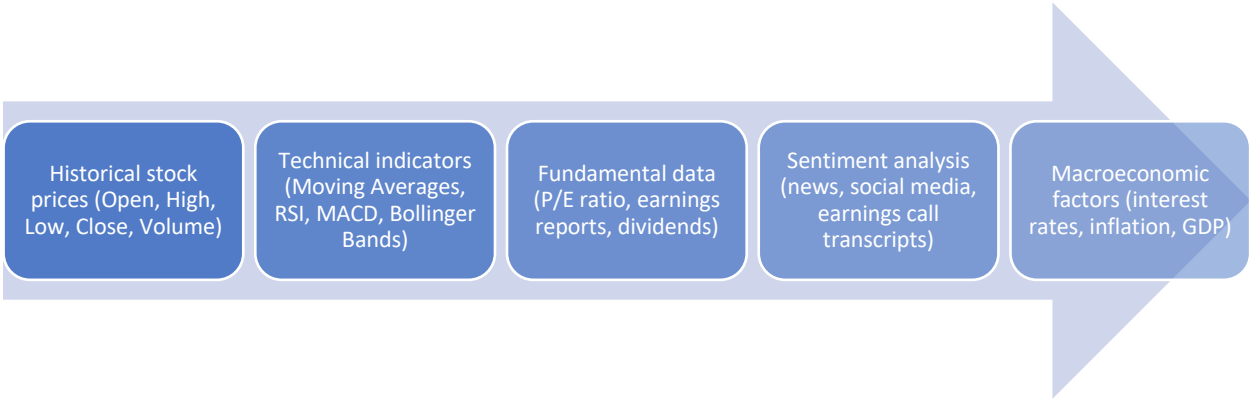

Figure 1. 1 Input Data (Features)

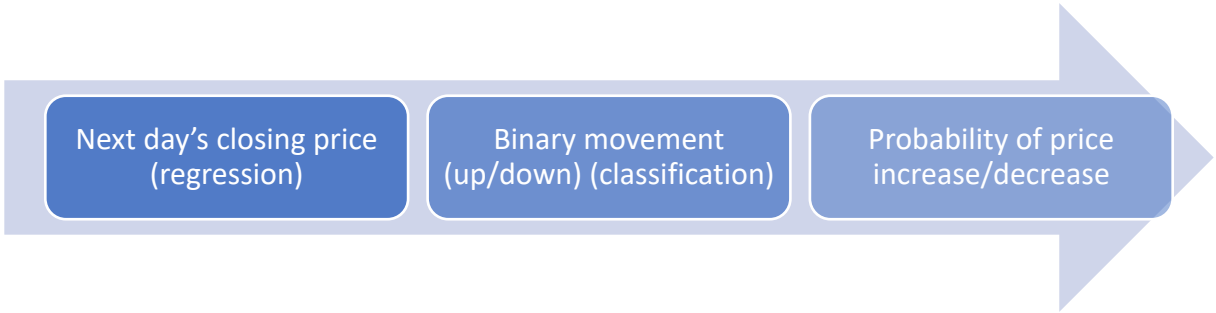

Figure 1. 2 Output (Target Variable)

However, the stock prediction still has many challenges, such as market volatility, non-stationarity, noise and overfitting due to influence by unpredictable factors. For evaluation, regression (e.g. MAE (Mean Absolute Error), RMSE (Root Mean Squared Error) and R-squared) and classification (e.g. Accuracy, Precision, Recall, F1-Score or AUC-ROC (for probabilistic models)) are very common in applications.

1.3 Market Analysis

As the development of Artificial Intelligence (AI), the stock market analysis has been revolutionized by using data-driven predictions, automated trading, and real-time sentiment analysis.

Here are some applications of AI Techniques in Stock Market Prediction:

Traditional Machine Learning (ML) Methods	• Supervised Learning: Random Forest, SVM, XGBoost • Unspervised Learning: K-Means Clustering, PCA
Deep Learning (DL) Methods	• Recurrent Neural Networks (RNNs/LSTMs) • Convolutional Neural Networks (CNNs) • Transformer Models (e.g., BERT, GPT for Sentiment Analysis)
Reinforcement Learning (RL)	• Deep Q-Networks (Learns optimal buy/sell strategies)

Table 1. 1 AI Techniques in Stock Market Prediction

The data sources for AI-based market analysis include price data, fundamental data, alternative data, news and sentiments. For instance, stock prices are influenced by market noise, non-stationarity and unpredictable events, such as geopolitical risks. In addition, AI models may fail in real-world situations and deep learning models are lack of interpretability. These kinds of limitations and challenges cause influence to live market.

Based on the analysis, AI-powered market analysis improves stock prediction through automated pattern detection (e.g. LSTM, CNN), sentiment-driven trading (e.g. NLP) and algorithmic trading strategies (e.g. RL).

1.4 Significance of the Study

Stock price prediction is a critical area of research in finance, economics, and artificial intelligence, due to its widely influence and implications. Here are more details about the significance of this project.

Through providing predictive insights and forecasting risks, stock price prediction helps investors make data-driven decisions, reduces emotional bias and enhances portfolio management. In the applications pf economic and business, it is developing quantitative trading strategies and increasing high-frequency trading efficiency. Also, it assists corporations in strategic planning, such as timing stock buybacks, IPOs, or mergers.

As the advancements of technological and methodological improved, some AI models, such as Transformers, Reinforcement Learning and LSTM, encourage interdisciplinary research area between finance and computer science.

On the aspects of Risk Management and Financial Stability, stock price prediction helps regulators monitor market anomalies and predicts market crashes or extreme volatility. Even through supporting automated trading systems to minimize the losses in stop-loss predictions.

For that individual impact, this prediction research reduces wealth inequality and encourages more financial literacy. Even small investors can get financial insights and benefit from Al tools.

Thus, stock price prediction has huge influence on investors, institutions and enterprises. Not only improving financial decision-making and AI applications but also contributes to reduce risks to make market has more stability.

1.5 Project Objectives and Questions

Clearly defining objectives and research questions helps guide the project toward meaningful outcomes, while developing a stock prediction model. The examples of primary objects are Price Forecasting, Trend Classification, Trading Signal Generation and Risk Assessment. The secondary objectives can be Model Interpretability, Benchmarking Performance and Real-World Applicability. What is more, there are specific requires for research questions, such as Predictive Performance, Practical Trading and Model Limitations/Challenges.

Price Forecasting	• Can an LSTM model predict next-day closing prices with <2% error?
Trend Classification	• Does SVM outperform logistic regression in classifying daily stock movements?
Risk Assessment	• Can GARCH models accurately predict 1-week volatility spikes?
Accuracy & Reliability	• Which model (LSTM, ARIMA, or Random Forest) has the lowest RMSE in predicting stock prices?
Market Regime Dependence	• How does high volatility (e.g., during earnings season) affect prediction accuracy?
Model Interpretability	• Which features (volume, moving averages) contribute most to XGBoost's predictions?

Table 1. 2 Project Objectives and Example Questions

1.6 Proposed Solution Framework

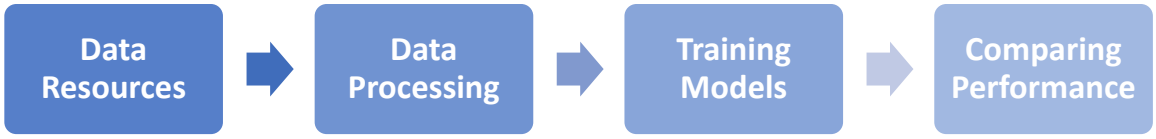

Figure 1. 3 the Framework of this Project

1.7 Report Organization

This report will have 5 parts:

- Introduction
- Literature Review
- Methodology
- Results and Discussion
- Conclusion and Future Work

Chapter 2

Literature Review

2.1 Overview

Stock price prediction has evolved from traditional statistical models to advanced ML (machine learning) and deep learning techniques. Here is a comparison between different approaches.

Traditional statistical Methods, such as ARIMA (AutoRegressive Integrated Moving Average) and GARCH (Generalized Autoregressive Conditional Heteroskedasticity) rely on statistical and time-series analysis. These methods work well for stationary data and risk management, while requires differencing and only for volatility. Other Technical Analysis-Based Models, such as Moving Averages and Bollinger Bands, are widely used by trader.

Then, ML models can capture non-linear relationships and adapt to changing market conditions. For instance, LR (Linear Regression) assumes linearity and SVM (Support Vector Machines) is effective in high-dimensional spaces, respectively. In a word, ML methods improve accuracy but require more feature engineering.

Deep Learning Methods, such as LSTM (Long Short-Term Memory), can learn complex patterns from sequential data. It can solve time-series data better, but requires large datasets, computationally intensive.

To achieve the best result, a hybrid approach (e.g. XGBoost for feature selection + LSTM for time-series modeling) can be used for practical applications.

2.2 Traditional Approaches

The first example is a book about stock markets, A *Random Walk Down Wall Street*, written by Burton Gordon Malkiel, a Princeton University economist, which popularized the random walk hypothesis. Malkiel argues that asset prices typically exhibit signs of a random walk, and thus one cannot consistently outperform market averages.[1]

In 1982, the journal 'Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation', written by Robert F. Engle, introduced a new class of stochastic processes called autoregressive conditional heteroscedastic (ARCH) processes, to generalize this implausible assumption. In this process, a regression model with maximum likelihood estimators and a simple scoring iteration had optimality properties and more efficient. The relative efficiency is calculated and can be infinite. To test whether the disturbances follow an ARCH process, the Lagrange multiplier procedure is employed. The test is based simply on the autocorrelation of the squared OLS residuals. [2]

After that, Tim Bollerslev extended ARCH to GARCH (Generalized Autoregressive Conditional Heteroskedasticity) in 1986. In the abstract, the author derived stationarity conditions and autocorrelation structure while using this new class of parametric models. Maximum likelihood estimation and testing are also considered. Finally, an empirical example relating to the uncertainty of the inflation rate is presented. [3]

In 1970s, Box and Jenkins raised an idea about Classic application of ARIMA (Autoregressive Integrated Moving Average) for stock price forecasting. Later in 2014, the paper 'Stock price prediction using the ARIMA model', written by Ariyo presented more details about ARIMA model which benefit on short term prediction. Published stock data obtained from NYSE (New York Stock Exchange) and NSE (Nigeria Stock Exchange) are used with stock price predictive model developed. [4]

---

[1] https://en.wikipedia.org/wiki/A_Random_Walk_Down_Wall_Street
[2] https://www.jstor.org/stable/1912773?origin=crossref DOI: 10.2307/1912773
[3] https://www.sciencedirect.com/science/article/abs/pii/0304407686900631?via%3Dihub DOI: 10.1016/0304-4076(86)90063-1
[4] https://ieeexplore.ieee.org/abstract/document/7046047

2.3 Advanced Methods using AI

As mentioned before, Automated Pattern Detection (e.g. LSTM, CNN), sentiment-driven trading (e.g. NLP) and algorithmic trading strategies (e.g. RL) are popular approaches.

The Automated Pattern Detection identifies recurring patterns using deep learning in price/volume data, such as head-and-shoulders and trends. The LSTM (Long Short-Term Memory) is best for sequential time series data because it captures long-term dependencies weekly or monthly. CNN (Convolutional Neural Network) detects spatial patterns in stock charts like candlestick shapes through large datasets but sometimes cause overfitting.

Sentiment-driven trading (e.g. NLP) will predict bullish and bearish via news and earing calls. For instance, the transformer-based NLP, BERT and Fin BERT have fine-tuned for financial text. The rule-based sentiment, VADER, focuses on social media via twitter or Reddit. These kinds of methods are good at capturing market-moving event but must deal with noise on social media.

Also, using RL (Reinforcement Learning) to optimize buy or sell decisions in real-time, such as maximize Sharp ratio. For instance, DQN (Deep Q-Network) can learn optimal actions (buy/hold/sell) via rewards. PPO (Proximal Policy Optimization) works better for continuous action spaces, such as adjust position size. Both methods are good at solving the changing market conditions but require careful reward tuning and expensive computation.

If combine all three approaches for robust predictions, it will integrate a Hybrid AI Trading System. In this system, LSTM forecasts price trends, NLP adjusts predictions based on news sentiment, while RL executes trades optimally. There are some challenges, such as overfitting and non-stationary markets. Cross-validation as well as retrain models weekly may be efficient solutions.

2.4 External Factors

Stock prices are influenced not only by historical trends and technical indicators but also by external factors, such as macroeconomic conditions, unpredictable events and market sentiment. To improving the accuracy of predictions, it is crucial to understand and analysis this external factors.

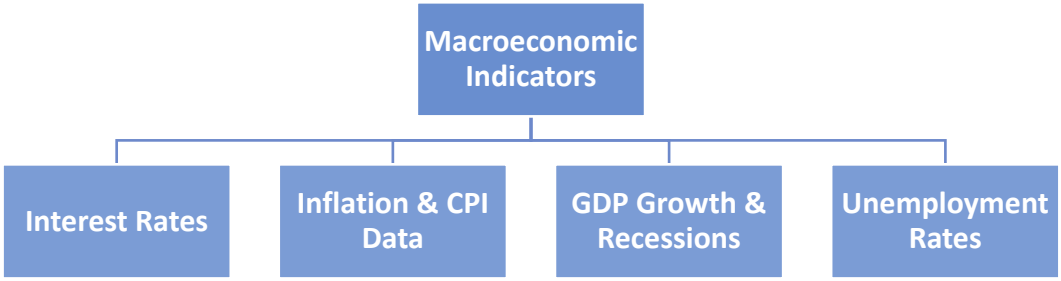

Figure 2. 1 External Factors (Macroeconomic Indicators)

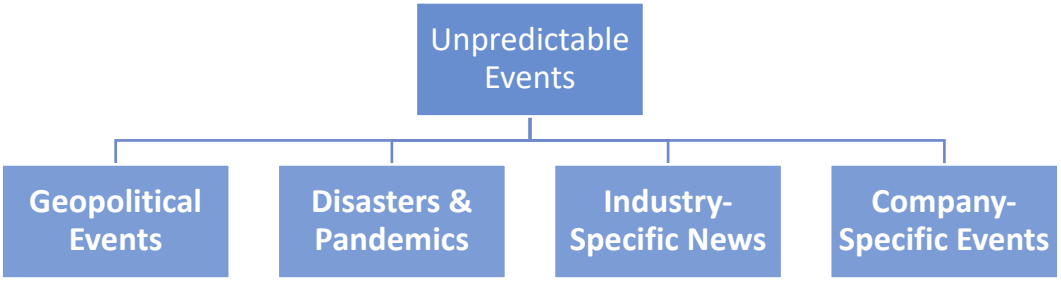

Figure 2. 2 External Factors (Unpredictable Events)

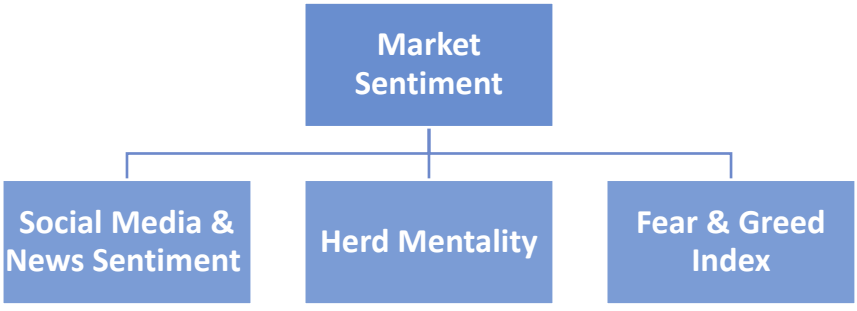

Figure 2. 3 External Factors (Market Sentiment)

2.5 Comparative Analysis of Existing Studies

A comparative analysis of stock prediction methodologies helps identify the robustness, and most efficient cases between traditional and advanced ML techniques. A structured comparison based on accuracy, data requirements, interpretability, and practical applicability as below.

Method	ARIMA/SARIMA	Random Forest/XGBoost	LSTM/GRU	Transformer Models
Pros	Simple, interpretable, works for linear trends	Handles nonlinearity, feature importance	Captures long-term dependencies, ideal for time-series	Parallel processing, superior for NLP integration
Cons	Fails with nonlinear patterns, ignores external factors	Struggles with sequential data, overfitting risk	Computationally heavy, needs large data	High resource demand, overkill for small datasets
Application	Short-term forecasting in stable markets	Medium-term prediction with fundamental data	Volatile markets (e.g. crypto)	Sentiment-augmented price prediction

Table 2. 1 Comparison between Traditional and Advanced ML Techniques

Approach	VADER	FinBERT	GPT-4
Accuracy	Low (~60%)	High (~85%)	Very High
Latency	Low	Medium	High
Data Needs	Social media text	Earnings reports	Multi-source news

Table 2. 2 Sentiment Analysis Techniques

Algorithm	Reward Strategy	Stability
DQN	Profit maximization	Low
PPO	Sharpe ratio	Medium
SAC	Risk-adjusted return	High

Table 2. 3 Comparison between RL Techniques

Chapter 3

Methodology

3.1 Introduction

In this project, I will use the dataset which contains historical stock price data for NVIDIA Corporation. This dataset included 'Open', 'High', 'Low', 'Close', 'Adj Close', and 'Volume' parameters which came from the website Kaggle. Through analysing this dataset, we can predict the future stock price with the model using machine learning, time-series models as well as achieve the insights into market behaviors.

Before I started the project, I use many kinds of techniques to pre-prepare data such as convert ‘file.csv’ into ‘file.mat’, which helps a lot from the beginning.

In order to get the best model and prediction, I will divide the original data (3020 samples, date from 20130101 to 20250215) into 83% train data (2518 samples, date from 20130101 to 20221231) and 17% test data (502 samples, date from 20230101 to 20200215).

For comparison between different methods, four classic machine learning methods will be applied in my project. That is LR (Linear Regression), NN (Neural Network), SVM (Support Vector Machine) and LSTM (Long Short-Term Memory) model. Also, the output graphs will have unsorted and sorted data.

To Evaluate the performance of these ML models, using RMSE and R-squared will helps to show the accuracy of predictions.

3.2 Data sources

The dataset is NVIDIA Stocks Data 2025 from the website Kaggle. This dataset contains historical stock price data for NVIDIA Corporation (NVDA) from 1999 to 2025, extracted from Yahoo Finance using the yfinance library. NVIDIA is a global leader in AI, gaming, and high-performance computing, making its stock highly relevant for financial analysis and machine learning models.[5]

In addition, this dataset can be used for Time Series Forecasting and Deep Learning Models widely.

Data Summary

Column Name	Description
Date	Trading date (YYYY-MM-DD)
Open	Opening stock price of the day
High	Highest stock price of the day
Low	Lowest stock price of the day
Close	Closing stock price of the day
Adj Close	Adjusted closing price after stock splits & dividends
Volume	Number of shares traded

Figure 3. 1 Data Summary6

[5] https://www.kaggle.com/datasets/meharshanali/nvidia-stocks-data-2025/data
[6] https://www.kaggle.com/datasets/meharshanali/nvidia-stocks-data-2025/data

3.3 Data processing

First, to make sure the first row from the data table has been removed, the preview of data as below:

Preview of the data:

Date	Adj Close	Close	High	Low	Open	Volume
25-Jan-1999	0.041556	0.045313	0.045833	0.041016	0.044271	5.1048e+08
26-Jan-1999	0.038331	0.041797	0.046745	0.041146	0.045833	3.432e+08
27-Jan-1999	0.038212	0.041667	0.042969	0.039583	0.041927	2.4437e+08
28-Jan-1999	0.038092	0.041536	0.041927	0.041276	0.041667	2.2752e+08
29-Jan-1999	0.036301	0.039583	0.041667	0.039583	0.041536	2.4403e+08
01-Feb-1999	0.037018	0.040365	0.040625	0.039583	0.039583	1.547e+08
02-Feb-1999	0.034152	0.03724	0.040625	0.036068	0.039583	2.641e+08
03-Feb-1999	0.034869	0.038021	0.038542	0.036458	0.036719	7.512e+07

Figure 3. 2 Preview of the Data Table

As refence, the Correlation heatmap and the Stationarity test (ADF) were also used in these procedures. ADF Test p-value is 0.999, so the conclusion is that 'The time series is not stationary.'

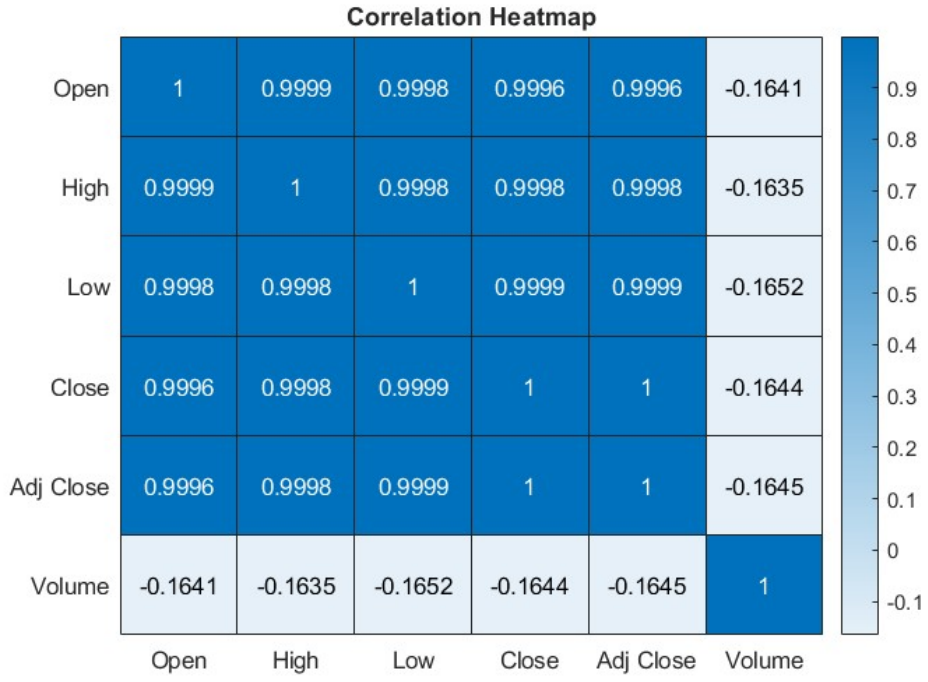

Figure 3. 3 Correlation Heatmap

The original data (3020 samples, date from 20130101 to 20250215) will be divided into 83% train data (2518 samples, date from 20130101 to 20221231) and 17% test data (502 samples, date from 20230101 to 20200215).

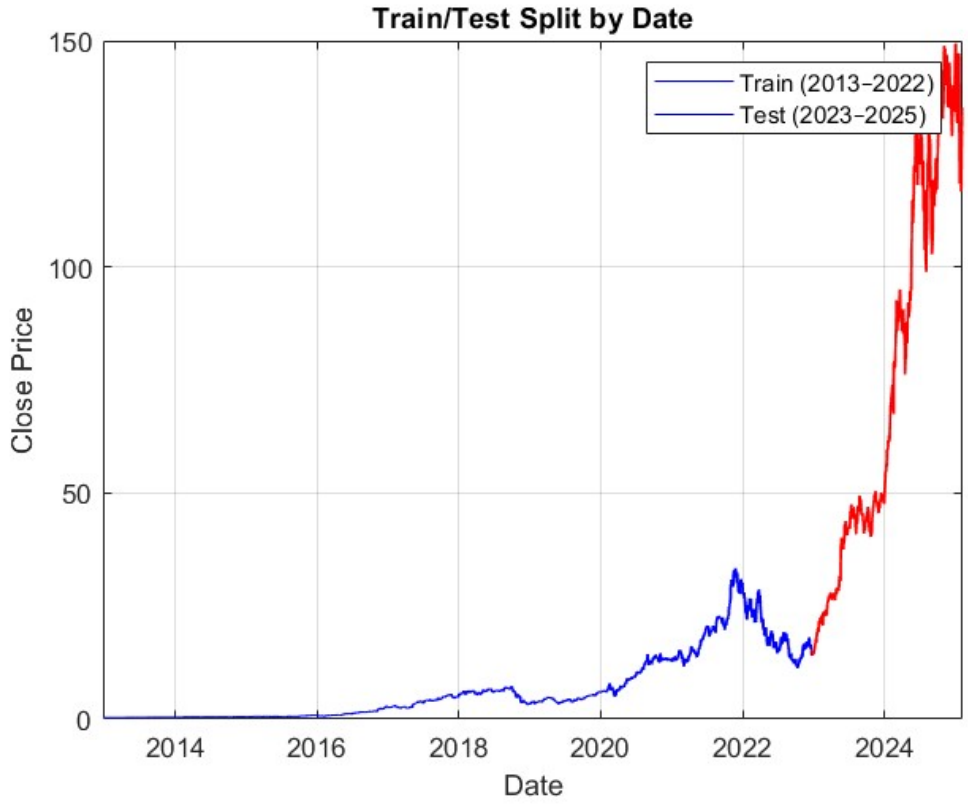

Figure 3. 4 Train/Test Data Split

Data solving procedures:

- For convenience, the first column ‘Price’ would be converted from date to datetime and convert to serial date numbers later.
- Select columns 2 to 6 for features (Open, High, Low, Close, Adj Close) as well as testLabels will be the "Close" column.
- Split the data into training and testing sets (83% train data and 17% test data).
- Apply trainFeatures and trainLabels into different models.
- Evaluate the performance using RMSE and R-squared.

3.4 Model Framework

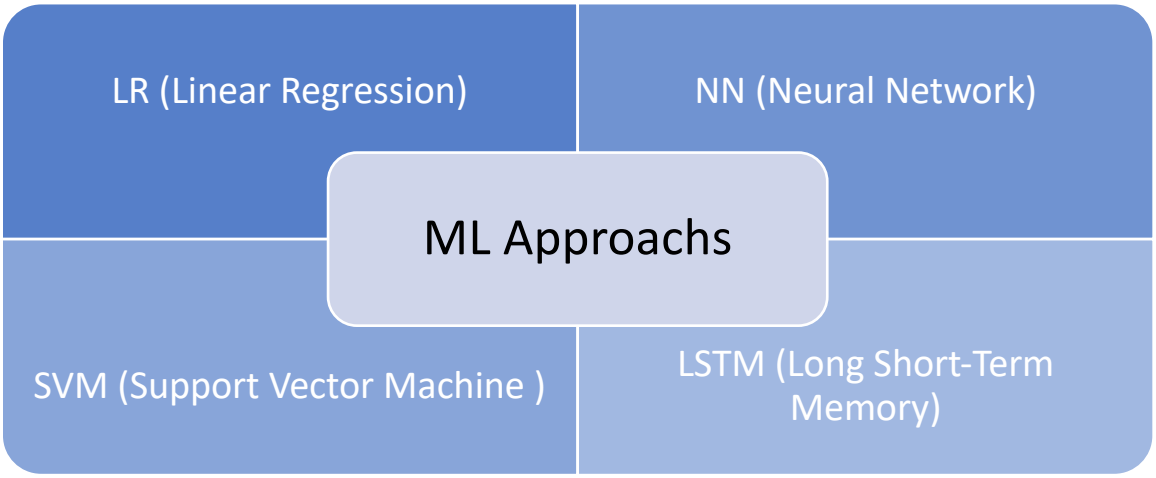

Figure 3. 5 Model Framework

In statistics, Linear Regression is a model that estimates the linear relationship between a scalar response (dependent variable) and one or more explanatory variables (regressor or independent variable). [7] In linear regression, the relationships are modeled using linear predictor functions whose unknown model parameters are estimated from the data.

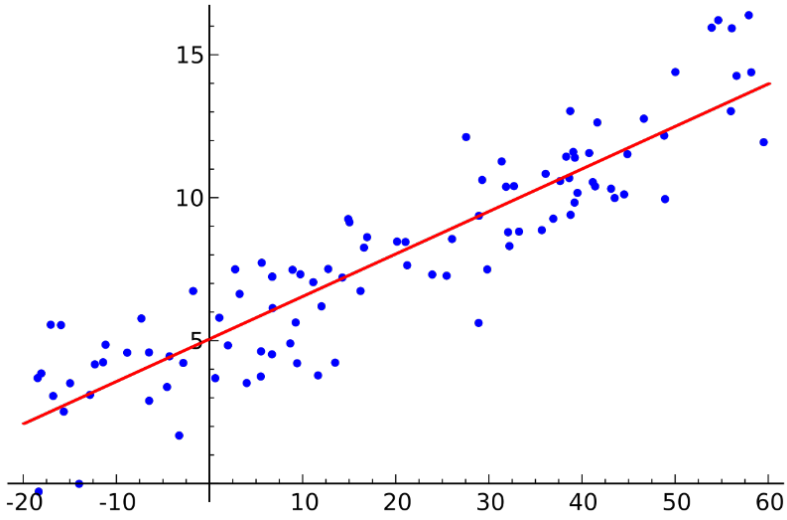

Figure 3. 6 the Example of Linear Regression8

[7] https://en.wikipedia.org/wiki/Linear_regression#cite_note-Freedman-2009-1
[8] https://en.wikipedia.org/wiki/Linear_regression#cite_note-Freedman-2009-1

In machine learning, a neural network (also artificial neural network or neural net, abbreviated ANN or NN) is a computational model inspired by the structure and functions of biological neural networks.[9]

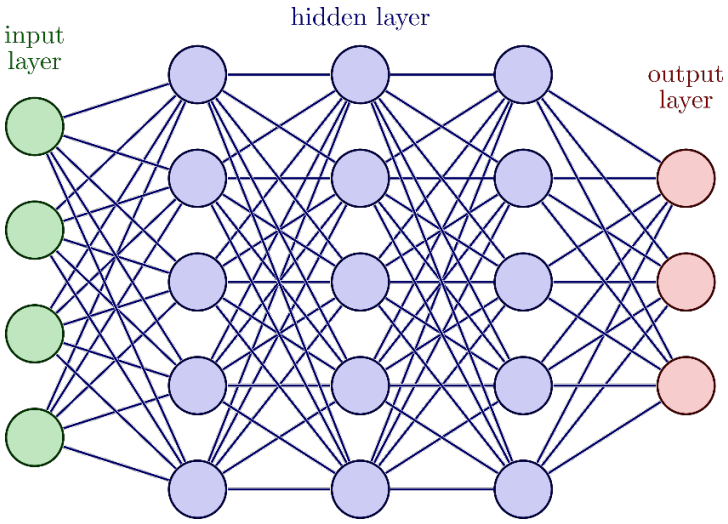

Figure 3. 7 the Example of Neural Network10

In machine learning, support vector machines (SVMs, also support vector networks) are supervised max-margin models with associated learning algorithms that analyze data for classification and regression analysis. [11] In addition to linear classification, SVMs can efficiently perform non-linear classification using the kernel trick.

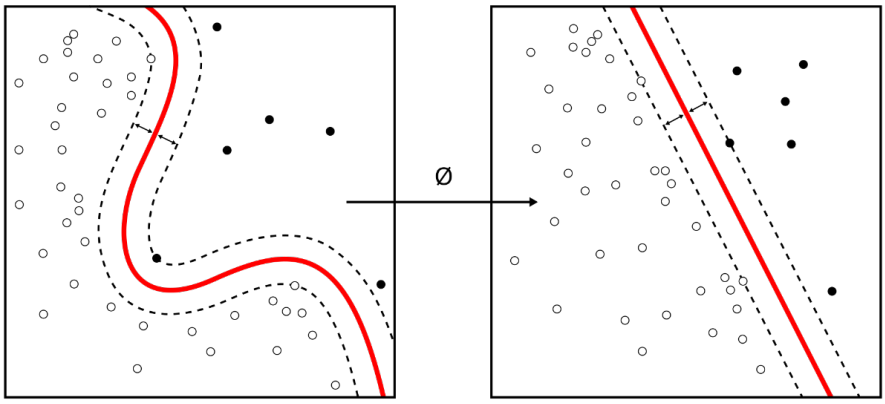

Figure 3. 8 the Example of Kernel SVM12

[9] https://en.wikipedia.org/wiki/Neural_network_(machine_learning)
[10] https://tikz.net/neural_networks/
[11] https://en.wikipedia.org/wiki/Support_vector_machine
[12] https://en.wikipedia.org/wiki/Support_vector_machine

Long short-term memory (LSTM) is a type of recurrent neural network (RNN) aimed at mitigating the vanishing gradient problem commonly encountered by traditional RNNs. Its relative insensitivity to gap length is its advantage over other RNNs, hidden Markov models, and other sequence learning methods.[13]

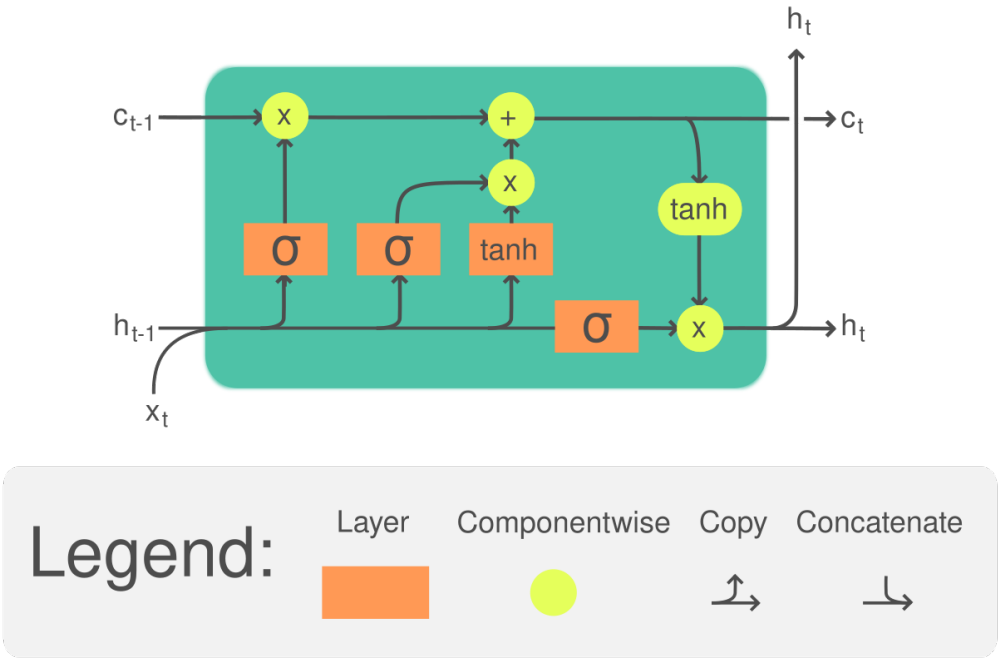

Figure 3. 9 the Example of LSTM Cell[14]

Memory Cell: A core component that acts as a conveyor belt for long-term memory, allowing information to flow through the sequence unchanged unless modified by gates.

Forget Gate: Decides which information from the previous cell state should be discarded (outputting a value between 0 and 1, where 0 means forget completely).

Input Gate: Determines which new information from the current input should be added to the cell state.

Output Gate: Controls what part of the cell state is used to generate the current output (short-term memory/hidden state).

[13] https://en.wikipedia.org/wiki/Long_short-term_memory
[14] https://en.wikipedia.org/wiki/Long_short-term_memory

3.5 Training Strategy

After splitting the original data into 83% train data and 17% test data, I will consider applying the machine learning models as LR (Linear Regression), NN (Neural Network), SVM (Support Vector Machine) and LSTM (Long Short-Term Memory).

Ordinary Least Squares (OLS) Linear Regression is used to learn a linear mapping from features to labels by minimizing mean squared error. The regression model is:

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_d x_d$$

In matrix form:

$$\hat{\boldsymbol{y}} = \boldsymbol{X\beta}$$

where $\boldsymbol{X}$ includes a bias column of ones, $\boldsymbol{\beta}$ contains the intercept and feature weights.

And then, solve the equation below to produce the best linear unbiased estimator under Gaussian noise assumptions: $\min_{\beta} \|\boldsymbol{X\beta} - \boldsymbol{y}\|^2, \boldsymbol{\beta} = (\boldsymbol{X^T X})^{-1}\boldsymbol{X^T y}$.

```
% ======================== Linear Regression ========================
% Train Linear Regression model
%lrModel = fitlm(trainFeatures, trainLabels);

% Predict using the test data
%predictedLabelsLR = predict(lrModel, testFeatures);
% Number of training samples
N = size(trainFeatures, 1);

% Add bias column (x0 = 1)
X_train = [ones(N, 1), trainFeatures];
y_train = trainLabels;

% Compute weights
beta = (X_train' * X_train) \ (X_train' * y_train);

% Add bias to test features
X_test = [ones(size(testFeatures, 1), 1), testFeatures];

% Predict
predictedLabelsLR = X_test * beta;

% ======================== Linear Regression ========================
```

Figure 3. 10 the Function of Linear Regression

In my project, the hidden Layer Size was setup as 20, and ‘trainbr’ is a Bayesian regularization backpropagation algorithm for feedforward neural networks, which helps reduce overfitting.

```
% ======================== Neural Network for Regression ========================
% Normalize features
[trainFeaturesNorm, psInput] = mapminmax(trainFeatures');
testFeaturesNorm = mapminmax('apply', testFeatures', psInput);

% Normalize labels
[trainLabelsNorm, psOutput] = mapminmax(trainLabels');
% Define the neural network structure (e.g., 10 hidden neurons)
hiddenLayerSize = 20;
net = fitnet(hiddenLayerSize, 'trainbr');
% 'trainlm' is the Levenberg-Marquardt algorithm for regression
%net = fitnet(hiddenLayerSize, 'trainlm');

% Normalize features
[trainFeaturesNorm, psInput] = mapminmax(trainFeatures');
testFeaturesNorm = mapminmax('apply', testFeatures', psInput);

% Normalize labels
[trainLabelsNorm, psOutput] = mapminmax(trainLabels');

net = train(net, trainFeaturesNorm, trainLabelsNorm);

predictedLabelsNNNorm = net(testFeaturesNorm);
predictedLabelsNN = mapminmax('reverse', predictedLabelsNNNorm, psOutput)';

% Train the network on the training data
%net = train(net, trainFeatures', trainLabels');  % Transpose data for neural network format

% Predict using the trained neural network
%predictedLabelsNN = net(testFeatures')';
% ======================== Neural Network for Regression ========================
```

Figure 3. 11 the Function of Neural Network

ε-insensitive loss:

$$L_\epsilon(y, f(x)) = \begin{cases} 0, & |y - f(x)| \le \epsilon \\ |y - f(x)| - \epsilon, & otherwise \end{cases}$$

Primal Problem:

$$\min_{w,b,\xi_1,\xi^*} \frac{1}{2}||w||^2 + C\sum(\xi_i + \xi_i^*),$$

with constraints:

$$y_i - f(x_i) \le \epsilon + \xi_i,$$

$$f(x_i) - y_i \le \epsilon + \xi_i^*,$$

$$\xi_i, \xi_i^* \ge 0.$$

Thus, the final model can be defined as

$$f(x) = \sum_i (\alpha_i - \alpha_i^*) K(x_i, x) + b$$

The KernelFunction = 'gaussian', which is

$$K(x_i, x_j) = \exp(-\gamma||x_i - x_j||^2)$$

SVR employs an ε-insensitive loss function to ignore small fluctuations, while kernel functions enable it to capture nonlinear relationships. As a result, SVR exhibits better generalization performance in financial data that contain noise and local nonlinear structures.

```
% ========================= Support Vector Machine (SVM) for Regression ==========================
% Train Support Vector Machine (SVM) Regression model
svmModel = fitrsvm(trainFeatures, trainLabels);

% Predict using the SVM model
predictedLabelsSVM = predict(svmModel, testFeatures);

% ========================= Support Vector Machine (SVM) for Regression ==========================
```

Figure 3. 12 the Function of Support Vector Machine

```
% ========================= LSTM =========================

closePrices = data.Close;
dates = data.Date;

% Ensure chronological order (oldest first)
closePrices = flipud(closePrices);
dates = flipud(dates);

%% Normalize Prices
closePrices = closePrices(:);
mu = mean(closePrices);
sig = std(closePrices);
normPrices = (closePrices - mu) / sig;

%% Create Sequences (window = 20 days)
windowSize = 30;
numSamples = length(normPrices) - windowSize;
X = cell(numSamples, 1);
Y = zeros(numSamples, 1);
for i = 1:numSamples
    X{i} = normPrices(i:i+windowSize-1)';
    Y(i) = normPrices(i+windowSize);
end

% Adjust corresponding dates for targets
targetDates = dates(windowSize+1:end);

%% Train/Test Split Based on Dates
trainStart = datetime(2013,1,1);
trainEnd   = datetime(2022,12,31);
testStart  = datetime(2023,1,1);
testEnd    = datetime(2025,12,31);

trainIdx = targetDates >= trainStart & targetDates <= trainEnd;
testIdx  = targetDates >= testStart  & targetDates <= testEnd;
```

Figure 3. 13 the Part 1 Function of Long Short-Term Memory

```
%% Reshape Input for LSTM [features x timeSteps]
% Now: 1 feature, 20 time steps
XTrain = cellfun(@(x) reshape(x, [1, windowSize]), trainFeatures, 'UniformOutput', false);
XTest  = cellfun(@(x) reshape(x, [1, windowSize]), testFeatures,  'UniformOutput', false);

YTrain = trainLabels;  % For training
YTrue  = testLabels;             % For evaluation

%% LSTM Network Definition
layers = [ ...
    sequenceInputLayer(1)
    lstmLayer(50, 'OutputMode','last')
    fullyConnectedLayer(1)
    regressionLayer];

options = trainingOptions('adam', ...
    'MaxEpochs', 100, ...
    'InitialLearnRate', 0.005, ...
    'GradientThreshold', 1, ...
    'Verbose', false, ...
    'Plots', 'training-progress');

%% Train LSTM
net = trainNetwork(XTrain, YTrain, layers, options);

%% Predict
YPredNorm = predict(net, XTest);
YPred = YPredNorm * sig + mu;   % Denormalize
YTestDenorm = YTrue * sig + mu;

% ========================= LSTM =========================
```

Figure 3. 14 the Part 2 Function of Long Short-Term Memory

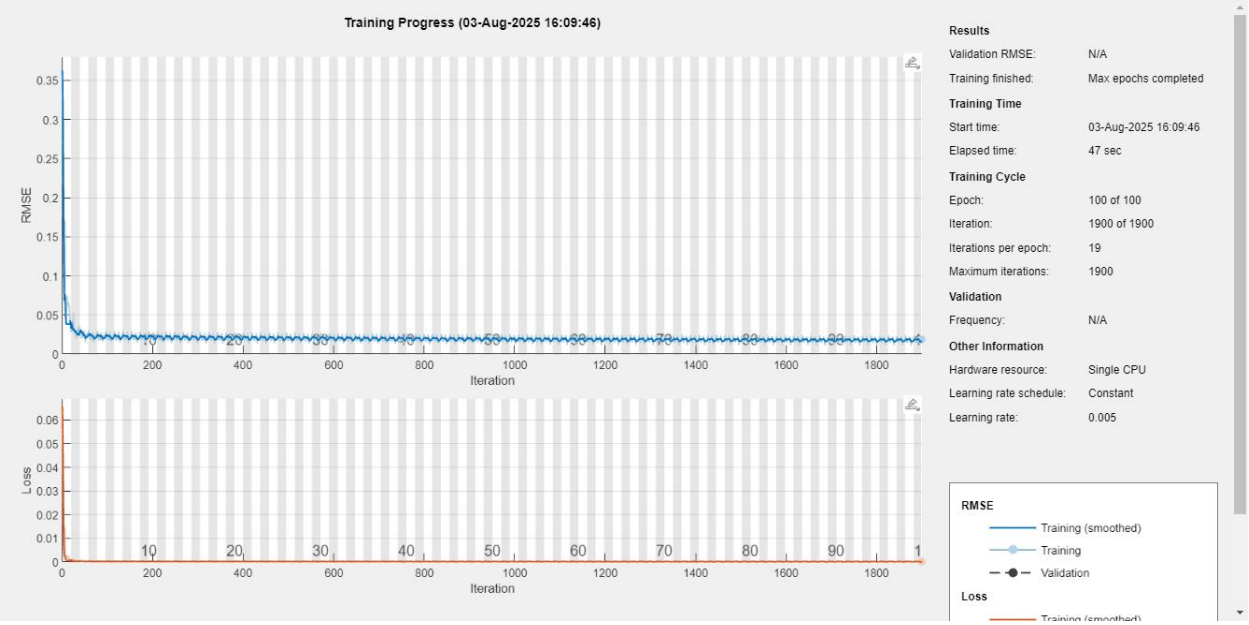

Figure 3. 15 the Training Progress of LSTM

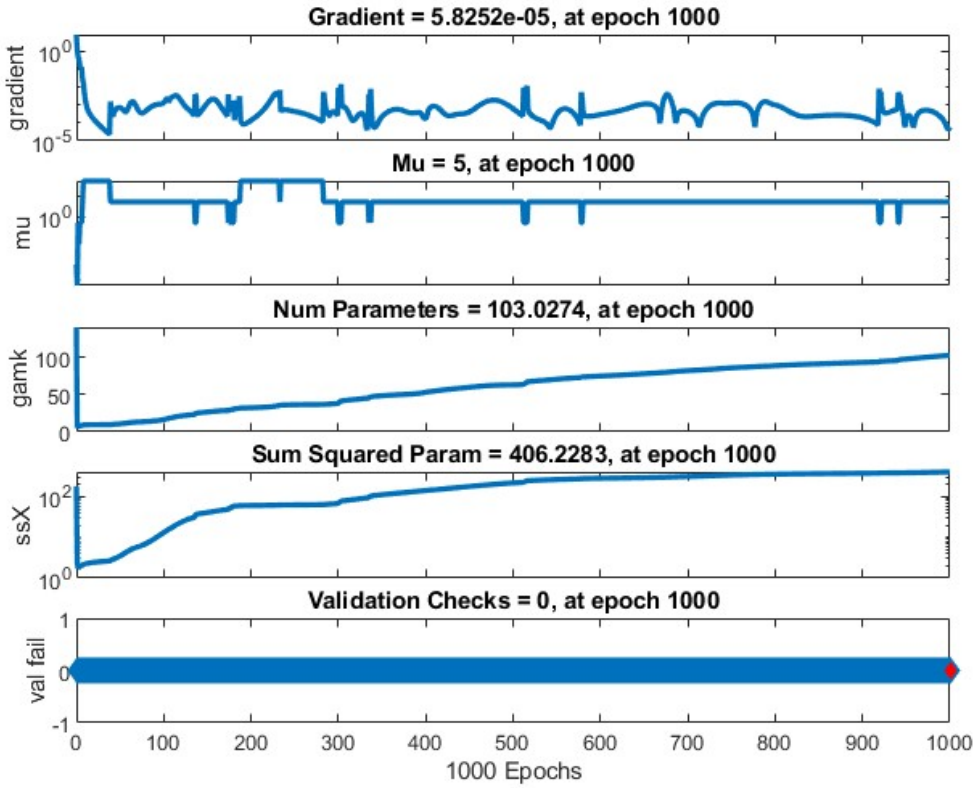

Figure 3. 16 Training State of NN

3.6 Evaluation

When evaluating stock price prediction models, RMSE and R-squared (R^2) are two key metrics used to assess performance. They have different purposes and should be combined for a complete analysis.

Metric	What It Measures	Scale	Use Case
RMSE	Average prediction error	Original units (e.g., $)	"How far off are predictions in dollars?"
R^2	% of variance explained	0 to 1 (or negative)	"Does the model explain price trends well?"

Table 3. 1 Key Differences: RMSE vs. R^2

RMSE (Root Mean Squared Error) is a measurement of the average magnitude of prediction errors in the same units as the target variable (e.g. dollars for stock prices). Lower RMSE means better model (smaller prediction errors) for regression tasks.

The formula as below:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Where y_i = Actual stock price, $\hat{y}_i$ = Predicted stock price, n = Number of observations.

Pros	Cons
•Easy to interpret (in original units). •Useful for comparing models on the same dataset.	•Sensitive to outliers (squares errors). •Doesn't indicate model direction (over/under-predicting).

Table 3. 2 the Pros and Cons of RMSE

R-squared (R^2) measures how much variance in stock prices is explained by the model (0% to 100%). It used for comparing different models on explanatory power and check how well the model captures trends.

The formula as below:

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$$

Where $\bar{y}$ = Mean of actual stock prices.

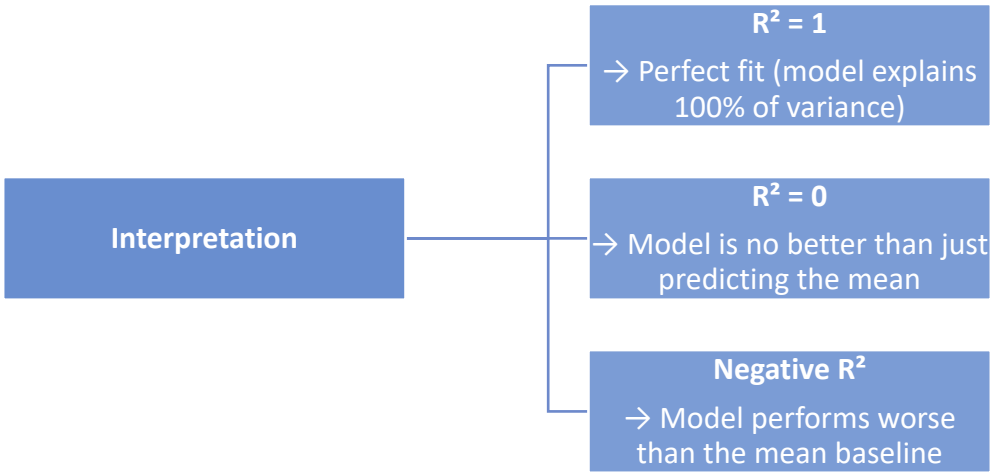

Figure 3. 17 the Interpretation of R-squared

Pros	Cons
• Standardized (0 to 1 scale). • Shows explanatory power.	• Misleading if the model is overfitted. • Doesn't indicate prediction errors in real units.

Table 3. 3 the Pros and Cons of R-squared

3.7 Ethical Considerations and Bias Mitigation

Stock price prediction models, particularly those using machine learning and AI techniques, present significant ethical challenges that must be addressed to ensure responsible deployment. This section introduces key ethical considerations and proposes mitigation strategies for bias in NVIDIA stock price prediction models.

The Financial AI may have market manipulation potential, data integrity concerns and overreliance on automated predictions. Pump-and-dump schemes may be exploited due to the risks of models as well as creating self-fulfilling prophecies through widespread model adoption. Then, challenges with historical data reflecting past market anomalies and potential inclusion of insider information or non-public data. Furthermore, danger of displacing human judgment entirely should be considered in the verification systems. Thus, the mitigations are implementing strict monitoring systems and source verification.

When considering the transparency and accountability measures, the requirement for regulatory compliance, continuous evaluation for concept drift and clear disclosure of model limitations are needed. Thus, documentation standards, appropriate risk warnings as well as transparent performance reporting may be benefit for the applications of ML models.

In addition, financial industry standards and data privacy regulations within the regulatory compliance framework also reduce the risks. For instance, MiFID II requirements for European markets, GDPR compliance for EU-based users and CCPA requirements for California.

3.8 Summary

The methodology followed a structured approach:

- Data preparation (cleaning, feature engineering).
- Model training & tuning (LR, SVM, NN, LSTM).
- Performance evaluation (RMSE, R^2).
- Ethical validation (bias checks, transparency measures).

Chapter 4

Results and Discussion

4.1 Introduction

In this chapter, the results will be evaluated by RMSE and R-squared.

The formula of RMSE as below:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Where y_i = Actual stock price, $\hat{y}_i$ = Predicted stock price, n = Number of observations.

The formula as below:

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$$

Where $\bar{y}$ = Mean of actual stock prices.

With comparison, the figures between Actual and Predicted data, unsorted data and sorted data were shown later. For LSTM models, the figure of Prediction Error also applied to evaluate.

4.2 Model Performance Evaluation

After solving data, plotting Predicted vs Actual Stock Prices with graphs would be easy for comparison. The steps of plotting as below:

- Create a time vector for the test data and convert testDates to datetime format as 'dd-MMM-yyyy'.
- Define the range for the x-axis (from January 2013 to February 2025) and for y-axis is from 0 to 170.
- Plot Actual vs Predicted for LR, NN, SVM Regression and LSTM with different colors.
- After that, I also use sort test data by dates for another analysis.

For instance, the plotting codes as below:

```
% ========================= Plotting Predicted vs Actual Stock Prices =========================

% Plot Actual vs Predicted for Linear Regression
figure;
plot(testDates, testLabels, 'b-', 'LineWidth', 1.5); % Actual Prices (blue)
hold on;
plot(testDates, predictedLabelsLR, 'r--', 'LineWidth', 1.5); % Predicted Prices (red dashed)
title('Linear Regression: Actual vs Predicted Stock Prices');
xlabel('Date');
ylabel('Stock Price');
legend('Actual', 'Predicted');

xlim([xStart, xEnd]);
xtickformat('dd-MMM-yyyy');
xtickangle(45);
ylim([0 max([testLabels; predictedLabelsLR; predictedLabelsNN; predictedLabelsSVM]) + 20])
%ylim([0 140]);
grid on;
```

Figure 4. 1 Plotting Predicted vs Actual Stock Prices

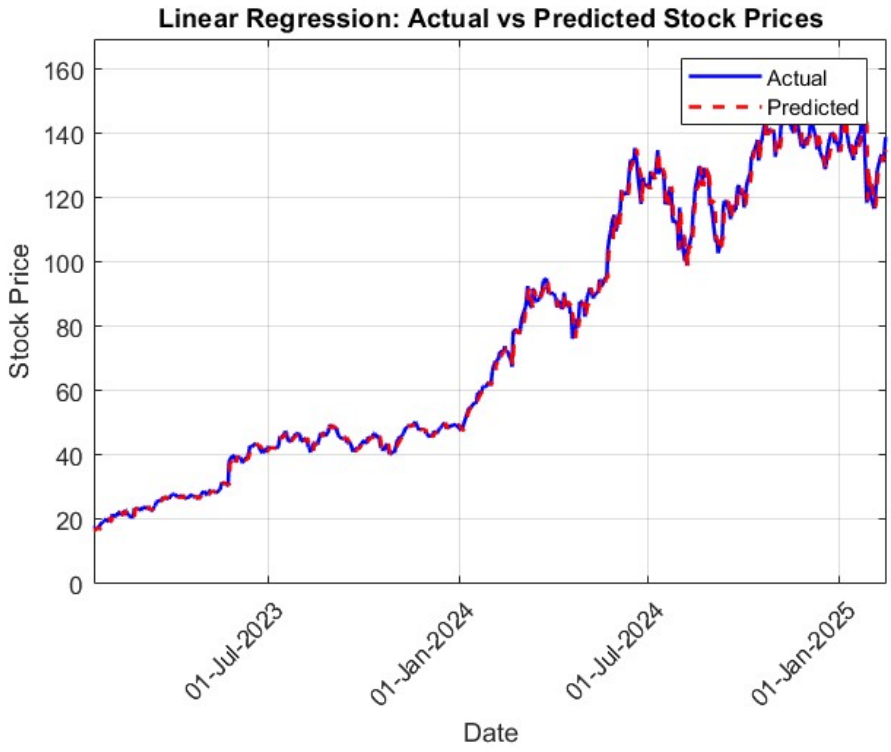

Figure 4. 2 the Actual vs Predicted Prices for LR

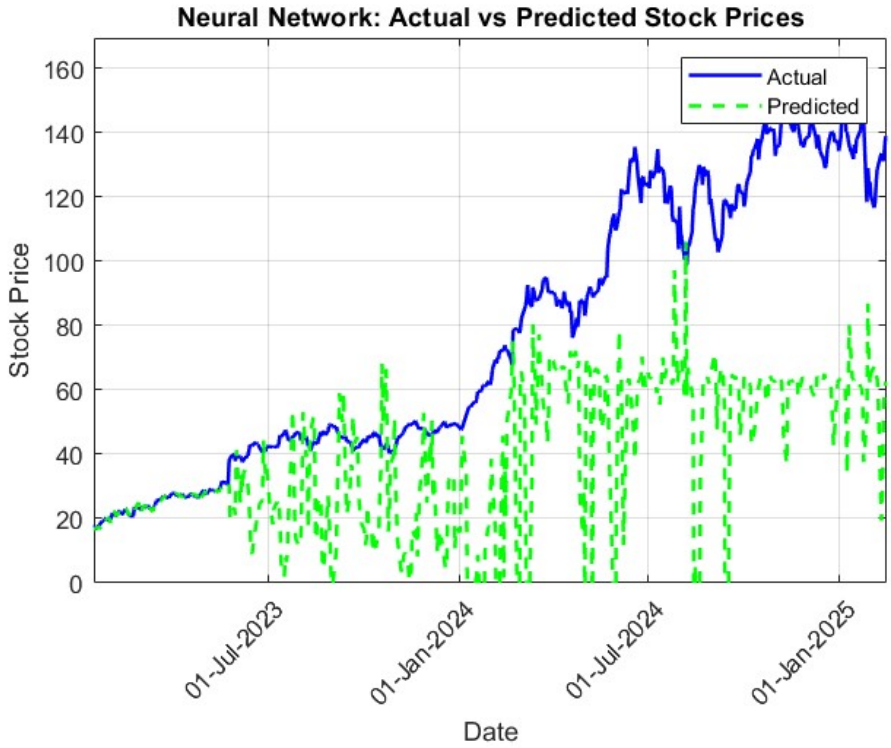

Figure 4. 3 the Actual vs Predicted Prices for NN

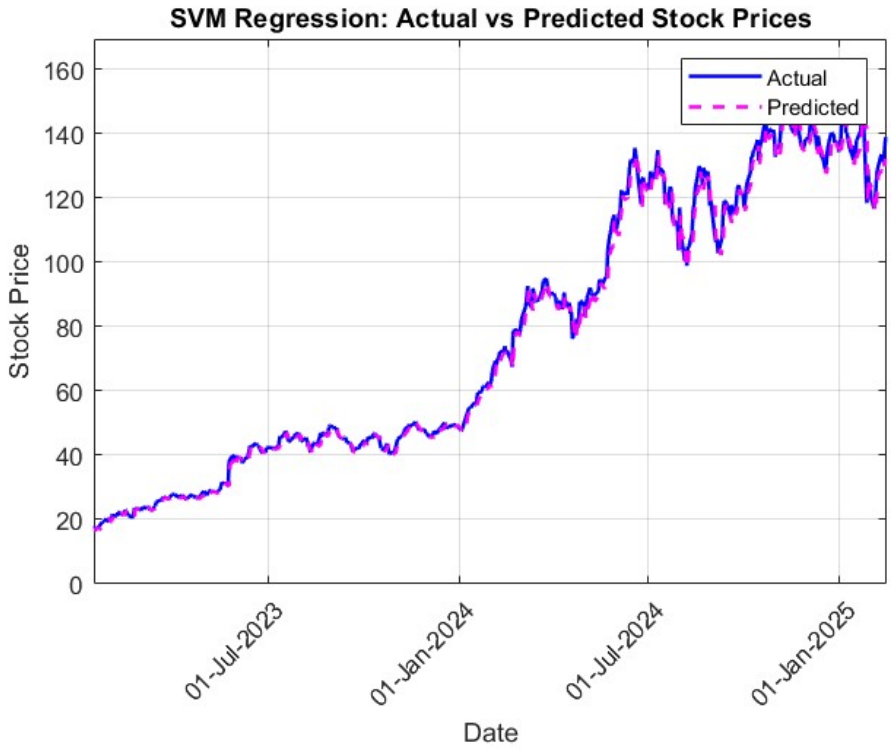

Figure 4. 4 the Actual vs Predicted Prices for SVM

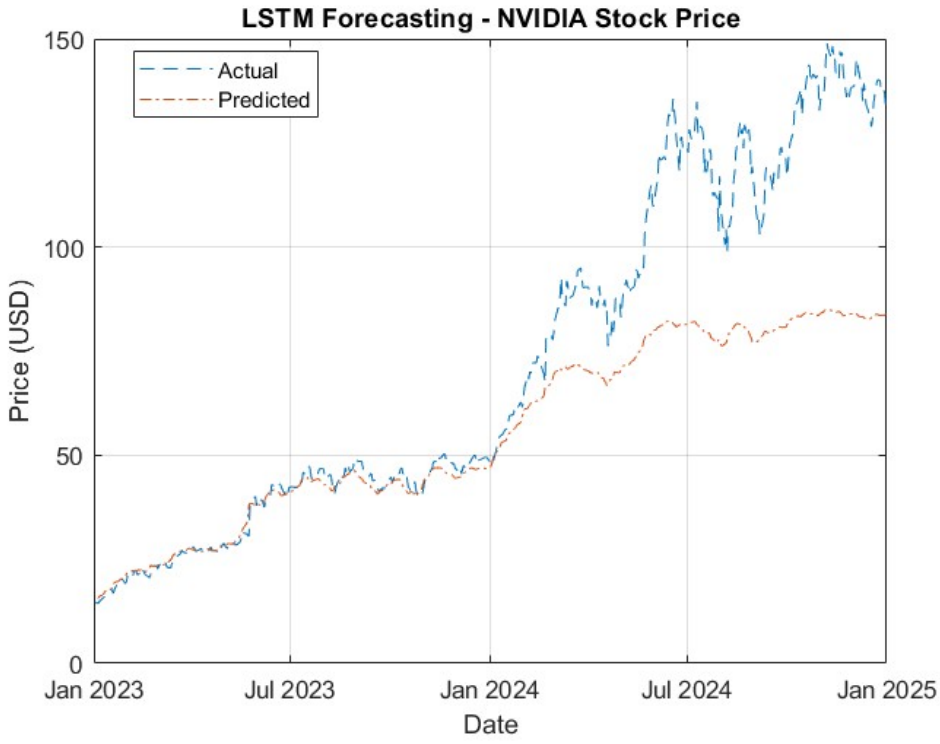

Figure 4. 5 the Actual vs Predicted Prices for LSTM

4.3 Impact of External Factors

The stock price of NVIDIA (NVDA) is influenced by a combination of internal and external factors. The internal factors include company-specific developments such as earnings, product launches and management changes, while the external factors play a crucial role in shaping its stock performance.

It is obvious that NVIDIA's stock is highly sensitive to AI trends, semiconductor cycles, and macroeconomic conditions. Because its importance in AI and related technology provides strong growth potential, risks like geopolitical tensions, competition, and crypto volatility can lead to sharp price swings.

At the aspect of Macroeconomic Conditions, higher interest rates (set by the Federal Reserve) may affect the investors' decisions. Recession fears may lead to reduced spending on gaming, data centers, and AI infrastructure, affecting NVIDIA's revenue.

For Semiconductor Industry Trends, shortages (e.g. during COVID-19), geopolitical tensions or oversupply can impact NVIDIA's ability to meet customers' demand. Also, the competition between Big Tech (e.g. Google, Amazon, Microsoft) could affect NVIDIA's market share.

As the spending of Cloud and Hyperscale increased, more investments from Microsoft Azure, AWS, and Google Cloud in AI infrastructure directly benefit the stock price of NVIDIA. Even Cryptocurrency Market Volatility had huge influence on GPU demand from miners, which caused price crashes (like in 2022) lead to excess inventory.

In addition, Geopolitical Risks, such as US-China tensions and Taiwan conflict risks, have the limitation of export controls on advanced AI chips to China (e.g. A800/H800 restrictions) which hurt NVIDIA's revenue (~20-25% of sales).

4.4 Comparison

```
% ========================= RMSE Comparison Plot =========================
modelNames = {'Linear Regression', 'Neural Network', 'SVM', 'LSTM'};
rmseValues = [rmseLR, rmseNN, rmseSVM, rmseLSTM];

figure;
bar(rmseValues);
set(gca, 'XTickLabel', modelNames, 'XTickLabelRotation', 30);
ylabel('RMSE');
title('Model Comparison - RMSE');
grid on;
```

Figure 4. 6 RMSE Comparison Plot

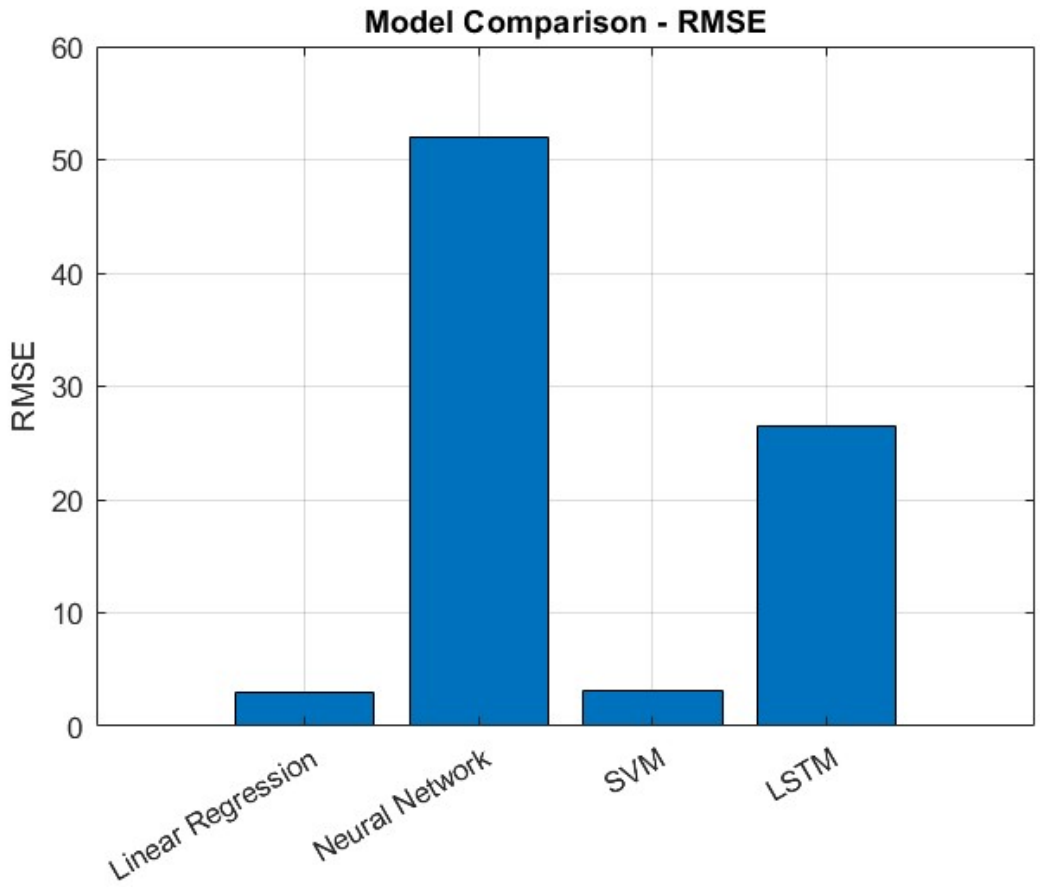

Figure 4. 7 the Comparison of RMSE

```
% ========================= R-squared Comparison Plot ==========================
R2Values = [R2_LR, R2_NN, R2_SVM, R2_LSTM];

figure;
bar(R2Values);
set(gca, 'XTickLabel', modelNames, 'XTickLabelRotation', 30);
ylabel('R-squared');
title('Model Comparison - R-squared');
grid on;
```

Figure 4. 8 R-squared Comparison Plot

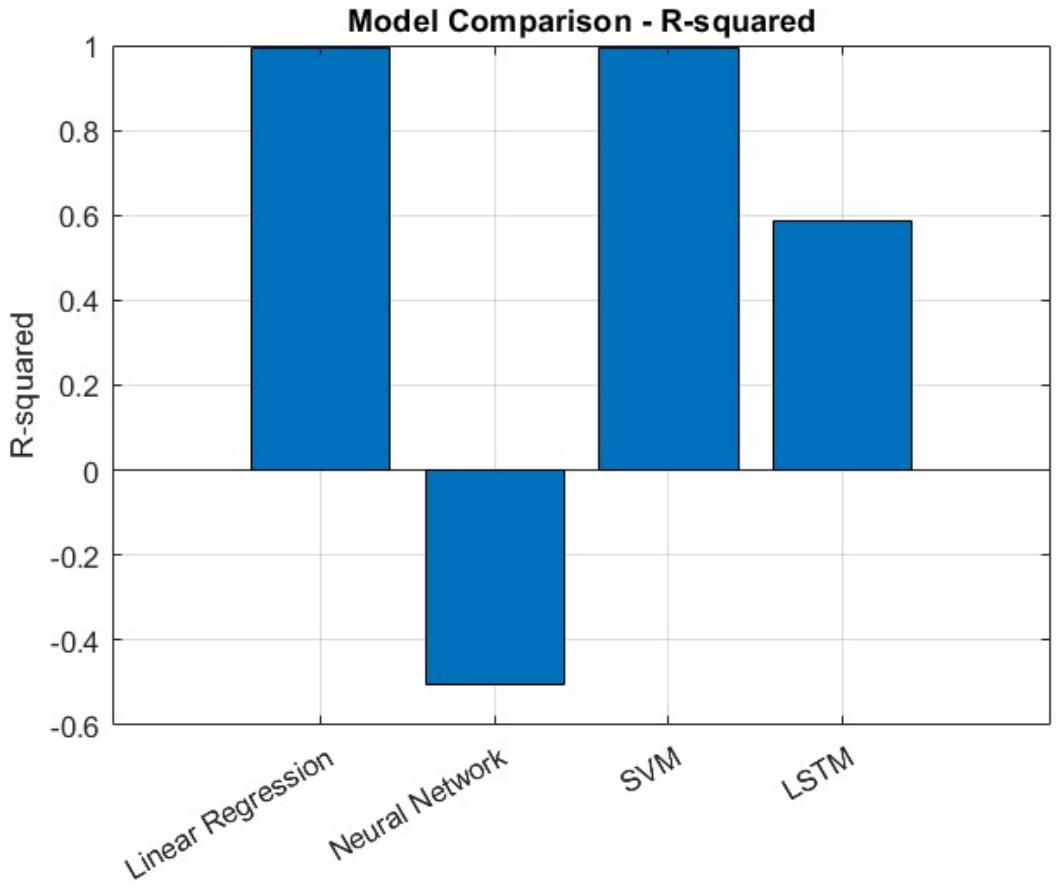

Figure 4. 9 the Comparison of R-squared

4.5 Discussion of Findings

Model	RMSE	R-squared
LR	2.9333	0.99521
NN	52.008	-0.5043
SVM	3.1188	0.99459
LSTM	26.4617	0.5857

Table 4. 1 the Comparison of RMSE and R-squared

Low RMSE indicates very small prediction errors and R-squared close to 1 means best fit. In LR model, RMSE = 2.9333 is lowest among models, R^2 = 0.99521, which means best fit and best performance model in this comparison. The extremely high R^2 shows that the linear model explains 99.5% of the variance in NVIDIA's stock price, meaning the input features (e.g. past prices, technical indicators and macroeconomic factors) have a strong linear relationship with the target.

The second-best model is SVM model, with RMSE = 3.1188 close to LR model and R^2 = 0.99459 which is very strong, slightly below LR. The reason might be SVM model works well for small-to-medium datasets and can capture non-linear relationships if using kernel tricks.

However, both NN and LSTM models do not have good performance. With RMSE = 52.008 and R^2 = -0.5043, NN model is the worst one among models, because the negative R^2 means model performs worse than a horizontal line.

Meanwhile, LSTM model has RMSE = 26.4617 and R^2 = 0.5857 with weak explanatory power. Although LSTM model is typically good for time-series forecasting, it has underperformed expectations in this problem.

4.6 Summary of Results and Insights

In conclusion, Linear Regression and SVM are the best models in this comparison, indicating that NVIDIA's stock price has strong linear trends or that SVM's kernel captured non-linearities well, while Neural Network failed badly and LSTM underperformed.

Although Linear Regression works well in this problem, it still has possible limitations that stock prices are not purely linear. And if tested on unseen data, LR might not generalize as well as expected. The risk of overfitting needs to be considered if the training data was too narrow. For SVM model, the possible limitations are computationally expensive for large datasets and may struggle with high-frequency stock movements.

Since both of NN and LSTM models are not performing well, the possible reasons should be considered. For instance, the NN may have memorized noise in the training data instead of learning pattern due to overfitting. If features were irrelevant or data was sparse, performance suffers caused by insufficient or noisy data. The solution may be trying deeper tuning, regularization (dropout, L2), or more input data.

Obviously, LSTM model needs large datasets and sequences. When stock prices have high randomness, LSTM model may struggle with short-term noise and could not learn patterns. Using different architectures (e.g. Transformer-based models), hybrid approaches or feature engineering (e.g. adding moving averages and volatility measures) may be an efficient solution for improving the performance of LSTM model.

Chapter 5

Conclusion and Future Work

5.1 Introduction

Stock price prediction is critical for investors, traders, and financial analysts, but it remains challenging due to market volatility, noise and external factors such as macroeconomic trends and geopolitical risks. NVIDIA's stock is particularly influenced by AI demand, semiconductor cycles, and GPU market trends, making it an interesting case study.

This project evaluates four models: Linear Regression, Neural Networks, Support Vector Machine and LSTM, to predict NVIDIA's stock price, meanwhile comparing their performance using RMSE and R^2. Also, the goal is to identify which model is the best to capture price trends and discuss limitations with comparison.

As we know, the stock prices are non-stationary, noisy and influenced by many unpredictable internal and external events. That is why Machine learning models must balance complexity and generalization to avoid overfitting.

In the future work, the improvements of models, enhancements of data, evaluation metrics or even broader applications need to be considered for the real-world datasets. In addition, more challenges will come out and be solved.

5.2 Summary of Project Contributions

After comparison using RMSE and R-squared methods, here comes some summary of this project.

Linear Regression (LR) and SVM performed best, with low RMSE (~3) and near-perfect R^2 (~0.99), suggesting that NVIDIA's price movements have strong linear or kernel-capturable patterns. This result implies that NVIDIA's stock may follow trends explainable by linear relationships such as past prices and moving averages.

Neural Network (NN) failed catastrophically (negative R^2), likely due to overfitting or poor hyperparameter tuning. LSTM underperformed (RMSE = 26.46, R^2 = 0.58), indicating that the model struggled with noise or insufficient training sequences. These two models highlight that proper architecture and data preprocessing are crucial.

In conclusion, LR or SVM may perform well for trend-following strategies or short-term traders, but hybrid models (e.g. LSTM + attention mechanisms) could be explored to improve sequential learning for practical predictions.

5.3 Addressing Project Objectives

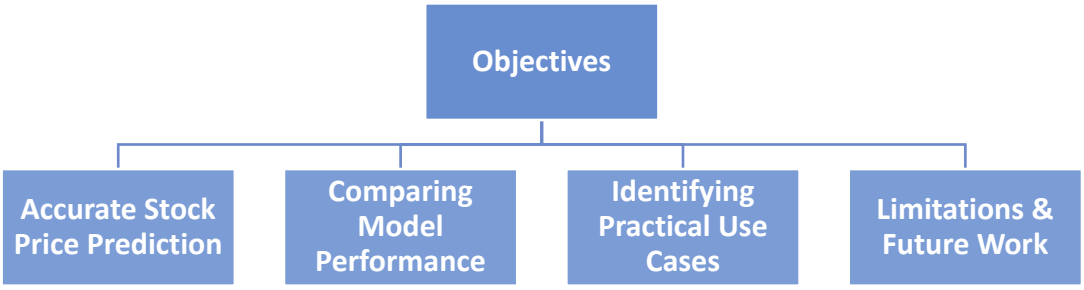

Figure 5. 1 the Objectives of this Project

5.4 Limitations of the Study

Limitations & Challenges

- **Noise vs. Signal** → Not all news impacts stock prices.
- **Data Latency** → By the time news is processed, the market may have already reacted.
- **Overfitting Risk** → Models may falsely correlate unrelated events.

5.5 Future Works

Future Trends

- **Explainable AI (XAI)** → Making AI predictions more transparent.
- **Quantum Machine Learning** → Faster stock price simulations.
- **Decentralized Finance (DeFi) + AI** → Predicting crypto markets.
- **Data Enhancements** → Macroeconomic data and industry-specific signals.
- **Evaluation Metrics** → Sharpe Ratio, Maximum Drawdown for risk-adjusted performance.

5.6 Final Remarks

While Linear Regression and SVM showed strong results, future work should focus on hybrid models and better feature engineering to handle market noise. The failure of NN/LSTM underscores that stock prediction requires domain-specific tuning, not just brute-force deep learning.

Bibliography

Books

[1] B. G. Malkiel, A random walk down Wall Street, W. W. Norton & Company, 1973.

Journal Articles

1. Engle, R. F. (1982). Autoregressive conditional heteroskedasticity with estimates of the variance of United Kingdom inflation. Econometrica, 50(4), 987–1008. https://doi.org/10.2307/1912773
2. Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. Journal of Econometrics, 31(3), 307–327. https://doi.org/10.1016/0304-4076(86)90063-1
3. Ariyo, A. A., Adewumi, A. O., & Ayo, C. K. (2014). Stock price prediction using the ARIMA model. 2014 IEEE Conference on Computer Communications and Networks (ICCCN). https://doi.org/10.1109/ICABCD.2014.6893976
4. Corwin, S. A., & Schultz, P. (2012). A simple way to estimate bid-ask spreads from daily high and low prices. The Journal of Finance, 67(2), 719–760. https://doi.org/10.1111/j.1540-6261.2012.01729.x
5. Fischer, T., & Krauss, C. (2018). Measuring long-term tail risk: Evaluating the performance of the square-root-of-time rule. Journal of Financial Markets, 38, 78–97. https://doi.org/10.1016/j.finmar.2018.03.003

Conference Papers

1. Thunga, S. P., & Neelisetti, R. K. (2015). Identifying metamorphic virus using n-grams and Hidden Markov Model. 2015 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC). https://doi.org/10.1109/ICCIC.2015.7435679
2. Hutto, C. J., & Gilbert, E. (2014). VADER: A parsimonious rule-based model for sentiment analysis of social media text. Proceedings of the International AAAI Conference on Web and Social Media, 8(1), 216–225. https://doi.org/10.1609/icwsm.v8i1.14550

Preprints & Technical Reports

1. Araci, D. (2019). FinBERT: Financial sentiment analysis with pre-trained language models. arXiv. https://arxiv.org/abs/1908.10063
2. Viro, J., & Viro, O. (2020). Fundamental group in the projective knot theory. arXiv. https://doi.org/10.48550/arXiv.2006.04032

Online Reports

Bloomberg. (2023). Introducing BloombergGPT, Bloomberg's 50-billion parameter large language model, purpose-built from scratch for finance. https://www.bloomberg.com/company/press/bloomberggpt-50-billion-parameter-llm-tuned-finance/