SMU
SINGAPORE MANAGEMENT UNIVERSITY

QF634-Applied Quantitative Research Methods

Group Project: Predicting Apple Stock price using LSTM and NLP features

Name	Student ID
Chan Ric	01499870
Lim Fang Yi	01340032
Muhammad Saqif	01495693
Zhao Geping	01500567

Table of Contents

1. Introduction

1.1 Introduction & Motivation

In this project, we aim to use a mixture of Natural Language Processing techniques, technical indicators and the LSTM model to predict stock prices. This distribution of stock prices can be used for several purposes, where it can be for portfolio management, by estimating mean returns and covariance matrix of returns. It can also be used for market risk management, to calculate Value-at-Risk and Expected Shortfall using Parametric Methods, Historical Simulation and Monte Carlo Simulation. Technical Analysis has been used by traders to look at trade signals with respect to trading volume and prices, with the belief that past trading activity and price movements are determinants to future price movements. (Hayes, 2024)

We also wanted to include sentiments of retail traders, which had significant involvement in the GameStop and AMC Short Squeeze saga, where retail traders from Reddit /r/wallstreetbets forum had banded together to buy shares of AMC and GameStop stock, having originated by YouTuber “Roaring Kitty”, alias “DeepFuckingValue” in Reddit, who posted his positions in GameStop. More than 100% of the shares public float were sold short, and the actions by the retail users led to short squeeze as hedge funds taking short positions had to pay much higher price to buy them back to cover their positions. (Verlaine & Banerji, 2021) This incident would inspire us to include user sentiment analysis in the prediction process.

Natural Language Processing has gained significant traction in the 2010s onwards, especially with the growth of Neural Networks. In 2013, word2vec was developed at Google, to get vector representations of words, incorporating information from neighbouring words using cosine similarity. This significantly improved compared to one-hot encoding or simple counts of words, thanks to the use of Recurrent Neural Networks. (Mikolov et al., 2013). In 2017, the groundbreaking “Attention is All You Need” paper resulted in development of the Bidirectional Encoder Representations from Transformers (BERT) model in 2018, which learns bi-directional representations of text to significantly improve contextual understanding. (NVIDIA, n.d.) . The idea of using Transformers for Natural Language Processing tasks continued developing in 2022 with OpenAI launching the Large Language Model (LLM) ChatGPT. (OpenAI, 2022).

For the ease of time, we decided to predict AAPL stock using LSTM, and hence utilizing comments about the AAPL stock, but the project can be scaled similarly to more stocks. Ideally we would have preferred to predict the S&P Index and index ETFs which uses it as the underlying but the difficulty of crawling data would make us abandon this direction.

2. Data Collection Process

2.1 Data Sources

To build a strong dataset, financial data were obtained from various sources. We used Python's yfinance library to get the historical stock prices, FRED for index/currency data, as well as WRDS data. To add a qualitative dimension, we scraped Reddit using Python to capture investor sentiment and market discussions which will then be ingested into the NLP process.

2.1.1 Reddit - NLP Data

Python libraries such as ‘praw’ and ‘psaw’ were leveraged to interact with Reddit’s platform. Upon creating an application on Reddit’s backend to gain the required credentials, we then have the capability of accessing and manipulating Reddit's API and database. Popular finance Reddit subreddits like "stocks," "investing," and "wallstreetbets" were crawled and we then systematically collected and fetched the texts. The data is then narrowed down and filtered for posts and comments containing relevant keywords related to financial markets, extracting the raw text strings. This data ranging from 2016-2022, is then stored in CSV format which will then be ingested into the NLP process and modelled into investor sentiment, market trends etc.

2.1.2 Financial Data - yfinance/FRED

Examining the correlation matrix of log returns below, two stocks, GOOGL (Google) and MSFT (Microsoft), emerge as suitable candidates. Their correlations with other stocks predominantly remain in the moderate 0.2 to 0.4 range, thereby reducing redundancy and preserving model integrity. By selecting only these two stocks, we aim to minimise overfitting risks and enhance both the model’s robustness and interpretability. Additionally, the analysis was conducted in a timeframe prior to 2016, between 2010 and 2015, to ensure no forward bias. This approach captures relevant market behavior without introducing forward biases related to any single year’s conditions.

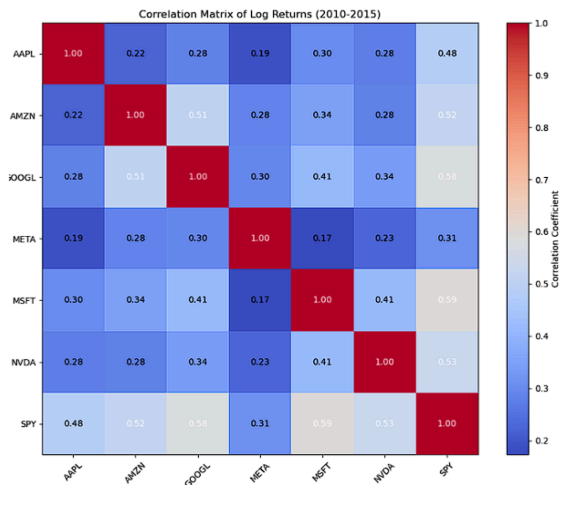

Figure 1: Correlation Matrix to determine correlation of MAANG stocks

To capture temporal dependencies and identify patterns in historical data, we incorporated lagged stock prices and returns. We will be using 1, 3, 6 , 12 day lagged AAPL stock data to capture and model momentum effects or mean reversion tendencies, and enhance our model's predictive accuracy. As Tsay (2010) highlights, historical data is crucial for forecasting future trends in time series analysis. To contextualize individual stock movements, we have also incorporated broad market indices like the DJIA. This introduces a base for our model to compare stock performance to a benchmark, so it can distinguish between company-specific factors and broader market trends, aligning with the asset pricing and risk factor framework of Fama and French (1993).

Thirdly, to account for global economic factors, we have included currency exchange rates, such as DEXJPYUS and DEXUSUK. Fluctuations in these rates, influenced by geopolitical events, monetary policy, and trade dynamics, can indirectly affect stock performance. As Pan, Fok, & Liu (2007) demonstrate, exchange rates play a significant role in understanding equity markets.

Lastly, to enhance our analysis, we incorporated technical indicators such as Simple Moving Averages (SMA), Exponential Moving Averages (EMA), and Relative Strength Index (RSI). These indicators expose the model to trends originating from momentum, market trends, and overbought/oversold conditions. By combining these technical signals with fundamental and macroeconomic factors used in the previous paragraphs, we aim to provide the model with a more comprehensive understanding of market dynamics.

2.1.3 Google Search Trends / WordSearch

Search frequency of the term "apple" in Google searches was also incorporated to gauge public sentiment and attention. This text-based feature, when combined with traditional financial metrics, allows us to assess whether increased public discourse correlates with stock movements. Research by Da, Engelberg, & Gao (2011) and Tumarkin & Whitelaw (2001) underscores the importance of investor attention in driving equity returns and hence we will be incorporating them into our model. In overall, the various features are encapsulated in the correlation matrix below.

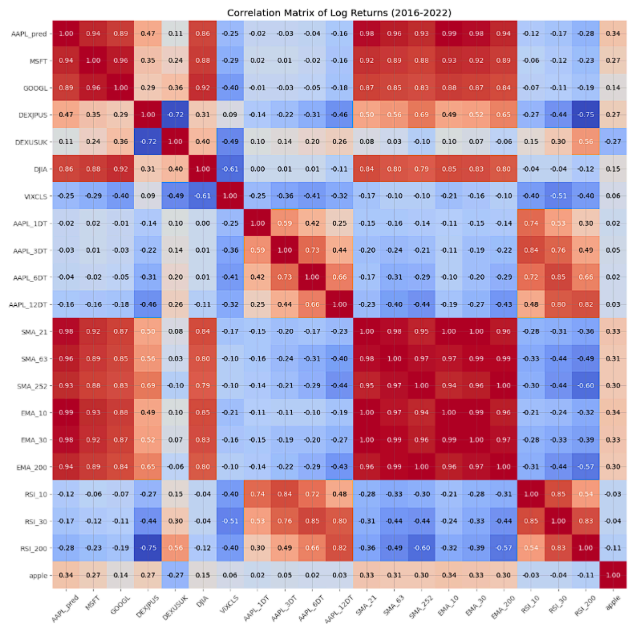

Figure 2: Correlation Matrix of the basic model's features excl. NLP features

2.2 Data Cleaning Process

To obtain granular daily search frequency data, we used Python's pytrends library to connect with Google Trends' API and retrieve search volume for specific keywords over our defined period. In this case, we aimed to analyze search trends for the keyword "apple" from January 1st, 2016, to December 31st, 2022. However, Google Trends API has limitations on the timeframe for a single request. To overcome this, we implemented a custom function fetch_daily_data. This function iteratively retrieves data in smaller chunks, specifically 180 days at a time. For each chunk, it constructs the appropriate timeframe string for the API request and utilizes pytrends.build_payload to build the necessary payload.

Following the payload construction, the function next calls pytrends.interest_over_time to retrieve the daily search data. Any retrieved data undergoes a cleaning step where the irrelevant "isPartial" column is dropped using pandas. Finally, the retrieved data for each chunk is concatenated into a single DataFrame named wordsearch. The code then checks for duplicate indices within the wordsearch DataFrame and removes them using pandas methods to ensure data integrity. Finally, we then save the wordsearch output into a CSV file for further analysis.

With respect to the Reddit comments about AAPL stock, we started by using Regex to convert all comments to lowercase, strip out the URLs (i.e. “i.e. http” and “www”), remove any mentions and subreddits, punctuation, special characters, line breakers and hashtags. We then used the emoji library to remove emojis. Finally, using the spaCy library, we removed the stop words and created word tokens which are then joined together. The word tokenization process breaks down the entire text into meaningful word tokens, which is effective for the English language, which has clear word boundaries. This is stored as a column cleaned_Comment_tokens_joined.

Next, we use another NLP preprocessing technique called lemmatization using the natural language toolkit nltk library, by reducing inflected forms of words into a common root word in the “lemma” form. For instance, the words “dancing”, “dances”, “dancer”, “dancer” are reduced to their lemma form “dance”. (Murel & Kavlakoglu, 2023), so that these inflected forms can be analyzed as a single item in the lemma form. We save this as a separate column cleaned_Comment_tokens_joined_lemmatized.

2.3 Feature Engineering Process

Numerical features such as financial data (e.g., index prices, forex) and word search data were logarithmically transformed using the np.log function. This helps us to effectively compress the range of values, especially for features with large variations and reduce the impact of outliers and to normalize the distribution of the data, making it more compatible.

Following that, with our lemmatized tokens in place, we then proceed to extract the sentiment score of it, using the nltk library. The SentimentIntensityAnalyzer class uses the pre-trained algorithm called VADER (Valence Aware Dictionary and sEntiment Reasoner), which has some of the below features useful to sentiment analysis. It can handle typical negations (e.g. “not good”), use of contractions as negations (e.g., "wasn't very good"), use of punctuation to signal increased sentiment intensity (e.g., "Good!!!"), using degree modifiers to alter sentiment intensity (e.g., intensity boosters such as "very" and intensity dampeners such as "kind of"), understanding of sentiment slang words which are prominent in social media posts, use of word-shape to signal emphasis (e.g., using ALL CAPS for words/phrases),   .polarity_score method, and sentiment-laden emoticons such as :) and :D (Hutto & Gilbert, 2014). where the corpura (cleaned_Comment_tokens_joined_lemmatized column) is passed through and returns the probability that each corpus (each observation in that column) belongs to a negative (neg) , neutral (neu) and positive sentiment (pos). Then a compound score (compound) is calculated, $\in [-1, 1]$ reflects how positive or negative the corpus is. They are then saved into 4 columns 'sia_cleaned_Comment_tokens_joined_lemmatized_neg',
'sia_cleaned_Comment_tokens_joined_lemmatized_neu',
'sia_Comment_tokens_joined_lemmatized_pos',
'sia_Comment_tokens_joined_lemmatized_compound' . And here, we have some numerical

representation of our corpus cleaned_Comment_tokens_joined_lemmatized which will be used as features for our regression problem. However a limitation of VADER is that when individual word tokens are stringed together in a sequence, the sentiment may actually change, for example in sarcasm where seemingly positive sentiments when broken down into word tokens, but stringing them together, it actually has a negative connotation.

Thus, we also wanted to explore the use of pre-trained transformers to extract sentiment score, which has been a huge success in NLP tasks, by utilizing the transformers library from Hugging Face, a community /hub where state of the art models are trained and published for public use. We chose the FinBERT library, which is a pre-trained NLP model to analyze sentiment of financial text. It is built by further training the BERT language model in the finance domain. First BERT was pre-trained on books and Wikipedia articles to capture general features of languages. FinBERT is then further pre-trained on financial news corpus to be more well suited for the features of the finance domain. And finally fine tuned with labeled data for sentiment classification using the Financial Phrasebank dataset. (Genc, 2020). The model provides softmax outputs, i.e. probabilities for 3 labels positive, negative or neutral. We do not perform any pre-processing for our comments in this case and instead make use of the transformers.AutoTokenizer class from_pretrained function to automatically select the original preprocessing techniques implemented by the author of the model, to get the 3 probabilities for the 3 different sentiments. Finally we took a simple weighted average of the sentiments and their probabilities to come up with a compound score, which is then stored under "negative_finbert_comment", "neutral_finbert_comment", "positive_finbert_comment". They will be used as features for our regression task.

3. Modelling Process

3.1 Classical LSTM Model

The classical LSTM model demonstrates significant potential in forecasting stock prices, leveraging its recurrent architecture to capture temporal dependencies in multivariate data effectively. In this analysis, the LSTM model was trained with four predictors – Opening Price, High Price, Low Price, and Volume Traded – and was tasked with predicting future price movements over a sequence of 50 days.

Model Configuration and Performance Metrics

The model was configured with the following hyperparameters:

- **Sequence length (n_steps_in):** 100 days
- **Prediction horizon (n_steps_out):** 50 days
- **Hidden units:** 16

- **Number of layers:** 1
- **Dropout:** 0.2
- **Learning rate:** 0.001
- **Epochs:** 1000

Training and test loss metrics were calculated using Mean Squared Error (MSE). Additionally, the Mean Absolute Percentage Error (MAPE) was computed to provide insight into the relative prediction error.

- **Training MSE:** 0.00027268
- **Test MSE:** 0.00217610
- **MAPE:** 430,082,720,000.0 (an anomalously large value indicative of computational or data preprocessing errors that need rectification)

The training MSE indicates that the model fits the training data with high precision, while the higher test MSE suggests challenges in generalizing unseen data.

<u>Temporal Dynamics and Accuracy</u>

Figure 14 in the PDF shows the LSTM model's predictions compared against actual Bitcoin prices for the test set. The model effectively tracks the overall downward trend, accurately predicting the general direction of price movements. However, deviations in the predictions indicate limitations in capturing high-frequency volatility, a characteristic typical of financial time series.

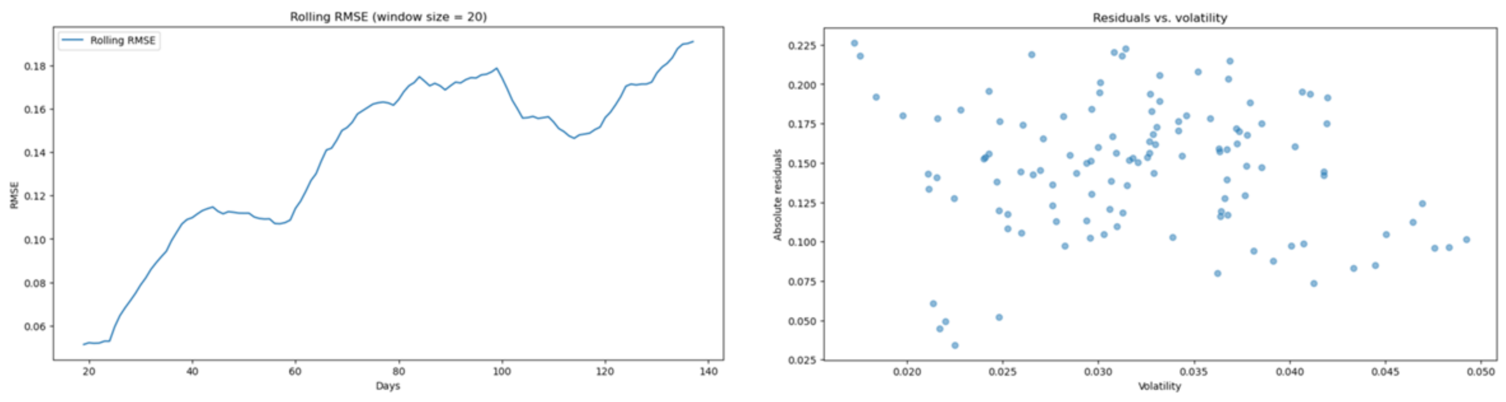

Figure 3: rolling RMSE (left) and residuals-volatility scatter plot (right)

Additionally, the final evaluation, conducted using a "one-shot multi-step prediction" approach, involved feeding the last 100 days of the test set as input and predicting the subsequent 50 days. As illustrated in Figure 16, the model successfully captured the overall decline in Bitcoin prices during this period. However, the absolute price levels exhibited slight underestimation.

Feature Representation and Processing Challenges

The sequence data was standardized for predictors (mean zero, unit variance) and scaled for the target variable (min-max scaling to [0, 1]). This preprocessing ensured consistent feature scaling, allowing the model to converge efficiently during training. Nonetheless, the observed anomaly in MAPE highlights the importance of verifying the preprocessing pipeline, particularly when converting tensor-based predictions back to their original scale.

Furthermore, the model's reliance on a single-layer architecture with limited hidden units might constrain its ability to model more complex temporal interactions, especially in volatile periods.

Model Interpretation and Data Leakage

Figure 5 highlights the risk of data leakage when historical predictors are overly aligned with the prediction targets. While the LSTM accurately fits the training data and closely follows test set trends, potential leakage due to improperly split sequences might overinflate its performance evaluation. This emphasizes the need for rigorous validation techniques to ensure fair assessments of model performance.

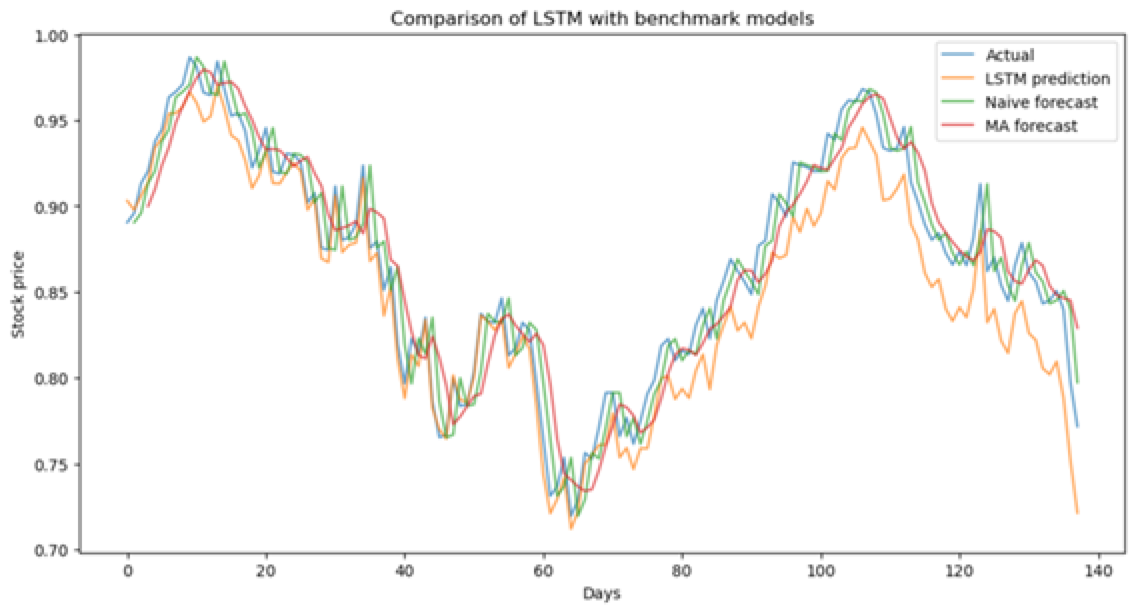

Figure 4: comparison of the LSTM model with benchmark models

Observations on Sequence Length and Volatility Sensitivity

Sensitivity analysis revealed that the sequence length of 100 days optimized the trade-off between training stability and prediction accuracy. Shorter sequences underutilized the model's memory, while longer sequences led to diminished returns on accuracy, increasing computational overhead.

Moreover, the residual analysis highlighted a negative correlation between residuals and price volatility. The model struggled in periods of heightened volatility, reflecting a limitation in its ability to capture extreme price swings. This provides an opportunity for incorporating advanced architectures (e.g., QLSTM or noisy QLSTM) to enhance robustness.

Comparative Performance

When benchmarked against simpler models like naive and moving average forecasts, the classical LSTM showed superior accuracy in capturing long-term trends but struggled with finer granularity. The naive forecast had an RMSE of 0.01862, while the moving average forecast scored 0.02207, both significantly worse than the LSTM's test RMSE of 0.00217610.

4. Discussions on Findings & Improvements

With respect to use of sentiment analysis in prediction of stock prices, there are 2 sides of an argument. One side argues that sentiments drive actions of retail investors, who then act which causes stock prices to move. Another side argues the other way round, where users bear strong sentiment and hence participate more actively in social media, after stock price movements, which can be generally found before and after major announcements e.g. earning calls, product launches and summits. We do believe both arguments to be true, which can explain volatility clustering, where volatility is likely to remain high for the next day given the previous volatility is high.

We could have also considered the use of other feature extraction techniques to create better word embeddings for corpus in NLP. For the case of sentiment analysis, it is a text classification task. For instance, TF-IDF (term frequency-inverse document frequency) can be considered, to quantify the weights of string representations (word tokens, lemmas) in a document amongst the corpus, by looking at the term frequency: the frequency of particular string relative to the entire document and the inverse document frequency: how common is the string representation is amongst the corpus. Alternatively, we can create word embeddings using pre-trained transformers as well. After creating word embeddings, we can then apply dimensionality reduction using techniques like Principal Component Analysis to reduce the feature space.

5. References

Genc, Z. (2020, July 31). *FinBERT: Financial sentiment analysis with BERT.* Medium. Retrieved December 12, 2020, from https://medium.com/prosus-ai-tech-blog/finbert-financial-sentiment-analysis-with-bert-b277a3607101

Hayes, A. (2024, July 4). *Technical analysis: What it is and how to use it in investing.* Investopedia. Retrieved December 11, 2024, from https://www.investopedia.com/terms/t/technicalanalysis.asp

Hutto, C. J., & Gilbert, E. (2014, May 16). *VADER: A parsimonious rule-based model for sentiment analysis of social media text.* Eighth International Conference on Weblogs and Social Media (ICWSM-14), Ann Arbor, MI. Retrieved December 12, 2024, from http://eegilbert.org/papers/icwsm14.vader.hutto.pdf

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013, September 7). *Efficient estimation of word representations in vector space.* arXiv. Retrieved December 11, 2024, from https://arxiv.org/pdf/1301.3781

Murel, J., & Kavlakoglu, E. (2023, December 10). *What are stemming and lemmatization?* IBM. Retrieved December 12, 2024, from https://www.ibm.com/topics/stemming-lemmatization

NVIDIA. (n.d.). *Google's BERT – What is it and why does it matter?* NVIDIA. Retrieved December 11, 2024, from https://www.nvidia.com/en-sg/glossary/bert/

OpenAI. (2022, November 30). *Introducing ChatGPT.* OpenAI. Retrieved December 11, 2024, from https://openai.com/index/chatgpt/

Simha, A. (2021, October 6). *Understanding TF-IDF for machine learning.* Capital One. Retrieved December 12, 2024, from https://www.capitalone.com/tech/machine-learning/understanding-tf-idf/

Verlaine, J. A., & Banerji, G. (2021, January 29). *Keith Gill drove the GameStop Reddit mania. He talked to the Journal.* The Wall Street Journal. https://www.wsj.com/articles/keith-gill-drove-the-gamestop-reddit-mania-he-talked-to-the-journal-11611931696

Da, Z., Engelberg, J., & Gao, P. (2011). In search of attention. *The Journal of Finance, 66*(5), 1461–1499. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=1364209

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. https://www.sciencedirect.com/science/article/abs/pii/0304405X93900235

Pan, M. S., Fok, R. C. W., & Liu, Y. A. (2007). Dynamic linkages between exchange rates and stock prices: Evidence from East Asian markets. *International Review of Economics & Finance, 16*(4), 503–520. https://doi.org/10.1016/j.iref.2005.09.003

Tsay, R. S. (2010). *Analysis of financial time series* (3rd ed.). Wiley. https://doi.org/10.1002/9780470644560

Tumarkin, R., & Whitelaw, R. F. (2001). News or noise? Internet postings and stock prices. *Financial Analysts Journal, 57*(3), 41–51. https://doi.org/10.2469/faj.v57.n3.2455