Stock Price Forecasting with a Temporal Fusion Transformer: Integrating Historical, Fundamental, and Sentiment Data for Improved Market Predictions

Yushan Yan [*1] **Kai Shen** [*1]

Abstract

We present a stock prediction framework using Temporal Fusion Transformers (TFT) that integrates historical market data with sentiment signals from financial news and social media. Our model demonstrates strong generalization capabilities when trained on multiple energy sector stocks. Starting with single-stock predictions, we expand to a dataset of 50 NYSE/NASDAQ-listed energy stocks with market caps exceeding 1000M. Results show that multi-stock training enables the model to capture broader sector trends and generate reliable predictions for both seen and unseen stocks. The framework processes daily price data, fundamental metrics, and sentiment analysis from Alpha Vantage and Reddit's WallStreetBets, providing probabilistic forecasts through an interactive interface.

1. Introduction

Modern financial markets are influenced by various factors, making stock price forecasting challenging. Beyond historical prices and fundamental indicators, investor sentiment reflected in news articles and discussions on social media also shapes price trends. This work integrates multiple data sources, including historical prices from yfinance and sentiment metrics from Alpha Vantage and QuiverQuantitative. By applying feature engineering and leveraging a Temporal Fusion Transformer (TFT) implemented with PyTorch Lightning, our approach captures complex temporal patterns and sentiment signals to improve stock trend predictions.

1.1. Motivation

The stock market, despite its inherent risks, continues to captivate a vast number of investors seeking substantial returns. As of 2023, over 61% of adults in the United States—amounting to more than 162 million people—own stocks. By 2024, more than 6,000 companies are expected to be listed on the NYSE or NASDAQ. Many of these stocks exhibit price fluctuations exceeding 5% within a single day, surpassing the annual interest rates offered by most banks. The allure of quick financial gains is strong, even as it carries significant risk. Therefore, developing a reliable method to predict stock price movements is highly desirable for mitigating investment risks.

1.2. Existing Approaches

Financial analysts have long favored straightforward approaches like Linear Regression and ARIMA models to predict stock prices (Bhuriya et al., 2017). These models became popular because they're easy to understand and explain to stakeholders. Linear Regression, in particular, works by looking for simple relationships between various market signals - from past price movements to trading volumes and technical indicators. While many traders still use these methods, they often fall short when dealing with the complex reality of today's markets. The fundamental problem is that Linear Regression assumes a straight-line relationship between these factors, but real market behavior is rarely so simple.

ARIMA models take a different approach, looking at how stock prices change over time by studying patterns in historical data (Ariyo et al., 2014). While this method can spot recurring trends and cycles, it has a major blind spot: it can't account for sudden market shifts driven by breaking news or changes in investor sentiment. Think of ARIMA as trying to predict tomorrow's weather by only looking at past temperatures, while ignoring incoming storm systems. Modern traders are increasingly turning to more sophisticated tools, particularly deep learning systems like LSTMs and Transformer models (Selvin et al., 2017). These advanced approaches can pick up on subtle market patterns that traditional methods miss, much like how modern weather forecasting combines multiple data sources to make more accurate predictions.

1.3. Related Works

The financial world has come a long way from simple forecasting tools. Modern approaches build on these early foundations while tackling their shortcomings. The first wave of AI models - RNNs and LSTMs - marked a significant step forward, as they could spot complex patterns that traditional methods missed (Selvin et al., 2017). But even these had

their limits, particularly when analyzing longer-term market trends or dealing with diverse types of market data.

Enter Transformer models, which revolutionized the field with their attention mechanisms. These models excel at connecting the dots across different time periods, much like an experienced trader who can link today's market movements to events from months ago. The Temporal Fusion Transformer (TFT) takes this even further - it not only makes predictions but also shows which factors drove those predictions, adding a layer of transparency that's crucial for real-world trading decisions (Hu, 2021).

Meanwhile, the explosion of digital media has opened up new frontiers in market analysis. Tools from companies like Alpha Vantage and Quiver Quantitative now scan everything from news headlines to social media chatter, measuring the market's pulse in real-time (Liu et al., 2021). By combining these sentiment indicators with traditional market metrics, traders can build a more complete picture of market dynamics.

Modern market analysis goes beyond just crunching numbers. Alpha Vantage's news sentiment tools scan financial media to gauge market mood, while Quiver Quantitative keeps its finger on the pulse of retail investor communities like WallStreetBets. These tools add a crucial human element to market analysis - after all, markets are ultimately driven by people's decisions and emotions.

By weaving together traditional market data with these sentiment indicators, modern forecasting systems paint a richer picture of market dynamics. It's like having multiple advisors - one watching the numbers, another monitoring news headlines, and a third tracking investor conversations online. This comprehensive approach is better equipped to navigate today's fast-moving and often unpredictable markets.

2. Method

This project employs a multi-source data integration and the Temporal Fusion Transformer (TFT) time series model to forecast stock price movements and their relationship with various sentiment signals. The overall process includes data acquisition and integration, feature engineering, dataset construction, model training and validation, as well as prediction result visualization. Detailed steps are described as follows:

2.1. Data Acquisition and Integration

2.1.1. Energy Stocks Selection:

We focus on analyzing the performance of energy sector stocks, a critical industry for the global economy. Energy stock price movements are driven by factors such as crude oil prices, geopolitical events, macroeconomic conditions, and renewable energy trends. These stocks are characterized by long historical records, high volatility, and significant profit potential, making them suitable for evaluating time series forecasting models.

For our analysis, we load a list of 50 energy stocks from a pre-prepared Excel file. This dataset enables the model to identify generalizable patterns across the historical and future performance of multiple stocks.

2.1.2. Data Sources:

Historical Stock Data: Using the `yfinance` interface, we obtain daily prices and trading volumes for the selected energy stocks over a specified date range. Additionally, we leverage `yfinance.Ticker` to retrieve fundamental indicators (e.g., EPS, MarketCap, Short Ratio, Price-to-Book ratio, Analyst Target Price) for these stocks. These metrics enrich the model's input features, thereby enhancing predictive accuracy.

News Sentiment Data (Alpha Vantage): Through Alpha Vantage's `NEWS_SENTIMENT` API, we acquire sentiment scores for financial news articles related to the target stocks. We aggregate these daily, calculating an average sentiment score per day to characterize the prevailing tone of media coverage over time.

Social Media Sentiment Data (Reddit WSB): Beyond official and media reports, we incorporate social media discussions. Using Quiver Quantitative's data services, we obtain sentiment scores from Reddit's WallStreetBets (WSB) community regarding the target stocks. These data are pre-compiled into an Excel file (`WSB_SENTIMENT_FILE`) and, upon running the code, we merge them with the stock's price and fundamental information. The simplified structure for data preprocessing and acquisition is illustrated in Figure 1.

This flowchart visually outlines the pipeline for data integration and processing:

- **Data Acquisition**: The process begins by loading the source file containing selected energy stock tickers.
- **Data Sources**:
  - **yfinance**: Obtains historical prices and fundamental indicators such as MarketCap, Price-to-Book ratio, and Short Ratio.
  - **Alpha Vantage**: Retrieves news sentiment scores through the `NEWS_SENTIMENT` API, aggregated as daily sentiment averages.
  - **Reddit WSB**: Gathers social media sentiment data pre-compiled into an Excel file, representing the sentiment scores of discussions on Reddit's WallStreetBets community.

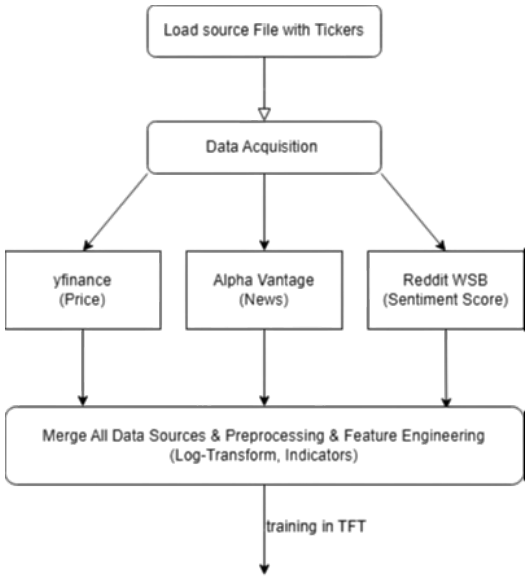

Figure 1. Data Processing Pipeline: Acquisition, Integration, and Feature Engineering

- **Data Preprocessing and Feature Engineering**: All acquired data sources are merged, preprocessed, and enhanced with feature engineering techniques, including logarithmic transformations, rolling volatility calculations, and additional technical indicators.
- **Training in TFT**: The preprocessed dataset is used as input to train the Temporal Fusion Transformer (TFT) model for stock price prediction.

2.2. Data Preprocessing and Feature Engineering

2.2.1. Data Consolidation:

We retrieved historical stock data, including Stock ticker and corresponding Open (*Target*), High, Low, Close, Volume, On-Balance Volume, Realized Volatility, Market Cap, Dividend rate, PE ratio, Price-to-Book ratio, Debt-to-Equity ratio, *Short* ratio and Analyst Target Price using Yahoo Finance API. For each stock, we merge the price data from yfinance with the associated fundamental indicators. Missing values are forward-filled as needed. To stabilize variance and approach normality, we apply log-transformation to price and volume data:

$$\bar{X}_t = \log(X_t) \text{ for X = Open, High, Low, Close} \quad (1)$$

And

$$\log(\text{Volume} + 1) \quad (2)$$

Additionally, we construct technical indicators such as OBV (On-Balance Volume). Other technical indicators are computed in corresponding forms. For example, Realized Volatility is computed as the rolling standard deviation of $\log(Ret)$ over a 10-day window, where $\log(Ret)$ is defined as $\log(Close_t/Close_{t-1})$.

2.2.2. Incorporating Sentiment Data:

The sentiment of public news is retrieved from Alpha Vantage, and the sentiment of social media is represented by the "wallstreetbets" Channel on Reddit (this could be further significantly enhanced by access to larger platform API such as Twitter. However, the costs of that are high for start-up projects). We match each day's stock prices with sentiment data from these two sources. For days without sentiment data, we use a neutral score of zero rather than discarding those entries.

To capture market dynamics, we add technical indicators like rolling volatility and log returns. We also include calendar features such as the day of the week, as trading patterns often show temporal regularities. All features are merged into a single DataFrame, indexed by date, and converted into a *TimeSeriesDataSet* suitable for the TFT model.

2.2.3. Final Feature Set:

Each daily record includes:

- Temporal Dimensions: date, day_of_week
- Market Price Features: log-transformed Open, High, Low, Close etc.
- Fundamental Features: EPS, MarketCap, ShortRatio etc.
- Sentiment Features: daily average sentiment from Alpha Vantage news and WSB social media sources

2.3. Dataset Construction

We use `TimeSeriesDataSet` to format the processed data for the Temporal Fusion Transformer (TFT) model.

- **Temporal Index:** We define *time_idx* (the day count from the start date) as the temporal index and group the data by *stock_id*.
- **Feature Division:** Features are categorized into:
  - *Static Features:* Attributes that remain constant over time, such as *stock_id* and *MarketCap*.
  - *Known Time-Varying Features:* Date-related categorical features that are known in advance.
  - *Unknown Time-Varying Features:* Variables that evolve over time, such as prices, volumes, and sentiment.
- **Target Variable:** The target variable is the log-transformed Open price.

- **Data Splitting:** Historical data is split into a training set (historical period) and a validation set (future forecast horizon).
- **Encoder-Decoder Setup:** The encoder length is set to 60 days, and the decoder length is user-defined based on the prediction horizon.

2.4. Model Training and Validation

To achieve robust and reliable forecasting results, we follow a systematic approach to model training and validation that emphasizes not only performance metrics but also generalization and interpretability.

2.4.1. Model Selection (Temporal Fusion Transformer):

The Temporal Fusion Transformer (TFT) is a time series forecasting model that combines recurrent neural networks, attention mechanisms, and gating layers to handle complex temporal dependencies and heterogeneous data sources. TFT can model global patterns across multiple time series and incorporate static, known, and unknown future covariates. The TFT is chosen as our primary forecasting architecture due to its ability to process multimodal time series data. Unlike traditional models that rely on linear assumptions, TFT incorporates both *static covariates* (e.g., time-invariant features like market capitalization) and *dynamic inputs* (e.g., price, volume, and sentiment data) in a unified framework (Lim et al., 2021). Mathematically, we consider a time series:

$$(y_t)_{t=1}^{T} \tag{1}$$

representing the log-transformed stock price. The TFT uses an LSTM-based encoder-decoder architecture to produce probabilistic forecasts:

$$\bar{y}_{T+1}, \bar{y}_{T+2}, ..., \bar{y}_{T+H} \tag{2}$$

Instead of outputting a single point estimate, the TFT model is capable of generating quantiles to quantify uncertainty. The TFT employs attention-based mechanisms to learn which variables and time steps are most critical, thereby improving both accuracy and interpretability. This modularity also allows easy incorporation of new features, such as sentiment data from Reddit WSB and Alpha Vantage. Figure 2 illustrates the architecture of TFT, highlighting the integration of historical targets, observed inputs, and covariates to generate point forecasts and uncertainty intervals.

2.4.2. Model Parameters and Optimization:

We employ a *QuantileLoss* function to predict multiple quantiles (e.g., 5%, 25%, 50%, 75%, 95%) of the future distribution of stock prices. This approach provides not just a single point forecast, but a range of plausible outcomes, enabling us to quantify uncertainty and better understand the risk associated with predictions. Such probabilistic forecasting is crucial in financial contexts, where decision-makers benefit from knowing the potential variability and confidence intervals of future price movements.

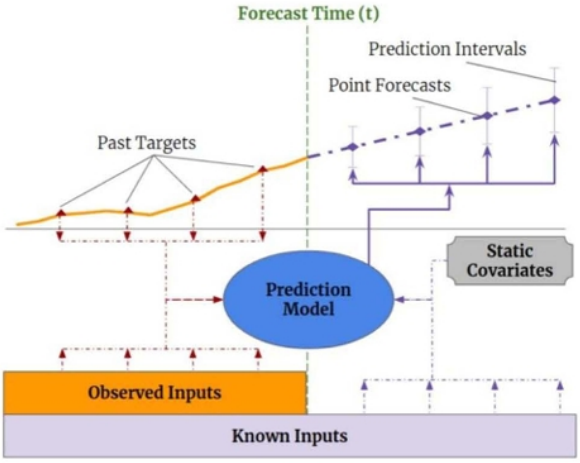

Figure 2. Temporal Fusion Transformer structure

The Temporal Fusion Transformer from PyTorch Forecasting is initialized with hidden size, attention head size 16, dropout 0.1, attention head size 4, learning rate 0.01, and max encoder length of 60. Key model parameters, including hidden layer size, attention head count, LSTM layers within the TFT architecture, and dropout probabilities, are tuned empirically. This tuning process may involve grid search, random search, or Bayesian optimization strategies. The model is then trained using PyTorch Lightning's Trainer. Early stopping and checkpointing ensure we select the best model based on validation loss. We monitor the validation loss to guide parameter selection, focusing on minimizing errors in out-of-sample predictions. Mathematically, given a set of quantiles:

$$Q = q_1, q_2, ..., q_M \tag{3}$$

the model predicts:

$$\bar{y}t + 1^{q_i}, \bar{y}t + 2^{q_i}, ..., \bar{y}_{t+H}^{q_i} \tag{4}$$

for each quantile q_i. The loss is computed using a quantile loss function described next.

2.4.3. Loss Function: Quantile Loss

The Quantile Loss for a single quantile q and prediction horizon is:

$$QL(y, \bar{y}^q) = \sum_{t=T+1}^{T+H} \max(q(y_t - \bar{y}_t^q), (q-1)(y_t - \bar{y}_t^q)) \tag{5}$$

This makes sure that if the actual value $y_t > \bar{y}_t^q$ and $q > 0.5$, the penalty is proportional to q-residual. For multiple quantiles, the final loss is the average across all quantiles. This allows the model to estimate the full distribution of future prices, not just the mean.

2.4.4. Training Procedure:

We structure our training using PyTorch Lightning's `Trainer`, which abstracts many of the common training loops and boilerplate code, ensuring a cleaner and more maintainable workflow. Through this framework, we streamline:

Batching and Dataloading: Training and validation datasets are fed into the model in batches, ensuring efficient GPU utilization and stable gradient estimation. Batch sizes are chosen based on memory constraints and performance considerations.

Callbacks and Early Stopping: To avoid overfitting and to promote generalization, we employ an `EarlyStopping` callback. This callback monitors the validation loss each epoch. If the validation loss stops improving after a certain patience threshold (e.g., 10 epochs), training is halted. EarlyStopping thus prevents the model from memorizing noise in the training set and conserves computational resources.

Model Checkpoints: We enable `ModelCheckpoint` callbacks to automatically save the model parameters (weights) whenever a new best validation score is achieved. This ensures that even if training is stopped prematurely or other issues occur, the best-known version of the model is securely stored.

Learning Rate Scheduling and Logging: A `LearningRateMonitor` callback tracks how the learning rate changes over time, especially when employing learning rate schedulers that adjust the step size to encourage convergence. Additionally, logging tools like TensorBoard are integrated to visually inspect training progress, monitor losses and metrics, and analyze trends in real-time.

2.5. Hyperparameter Tuning

Hyperparameter tuning is critical to the improvement of the model performance, helping find a balance between underfitting (not capturing patterns) and overfitting (memorizing the past without generalizing). In this study, we focus on one stock ticker, Exxon Mobil (XOM), to demonstrate the tuning process. The key parameters include:

2.5.1. Learning Rate Dropout Rate

Adjusting learning rate and dropout rate can stabilize training and improve generalization. A well-chosen learning rate reduces training time and achieves a lower validation loss. With a smaller learning rate (0.001), the model tends to give more sensitive predictions of near future. The dropout rate influences the model's ability to strike a balance between underfitting and overfitting. Demostration are shown in Figure 3 and Figure 4

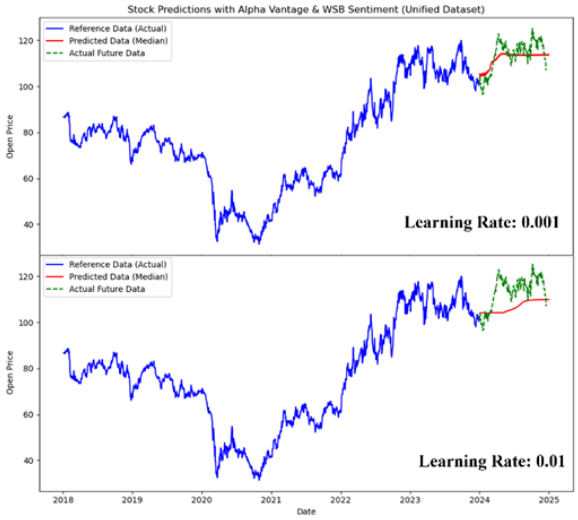

Figure 3. Demonstration of learning rate impact on the performance

2.5.2. Model Architecture Parameters

Hidden sizes and layers can influence the complexity of the TFT model. More complexity can fit more subtle patterns but also risk overfitting. A moderately sized network can perform as well as a larger one but trains faster. We adopt a network of 256 hidden sizes and 3 LSTM layers to predict energy stock prices in the near future with reasonable costs. Encoder/prediction lengths define the forecasting problem structure. Correct lengths aligned with known market cycles or data availability can significantly improve forecast accuracy. This real-time feedback allows us to make informed adjustments based on how the training evolves.

2.5.3. Validation and Performance Assessment:

During training, a portion of the time-indexed data is reserved as a validation set. The validation set is created by using data that chronologically follows the training data. The final time index is known, and we choose a cutoff so that the last portion of the dataset forms the validation set. The model's performance on this validation portion serves as an unbiased indicator of how well it generalizes to unseen data. We calculate validation losses at the end of each epoch,

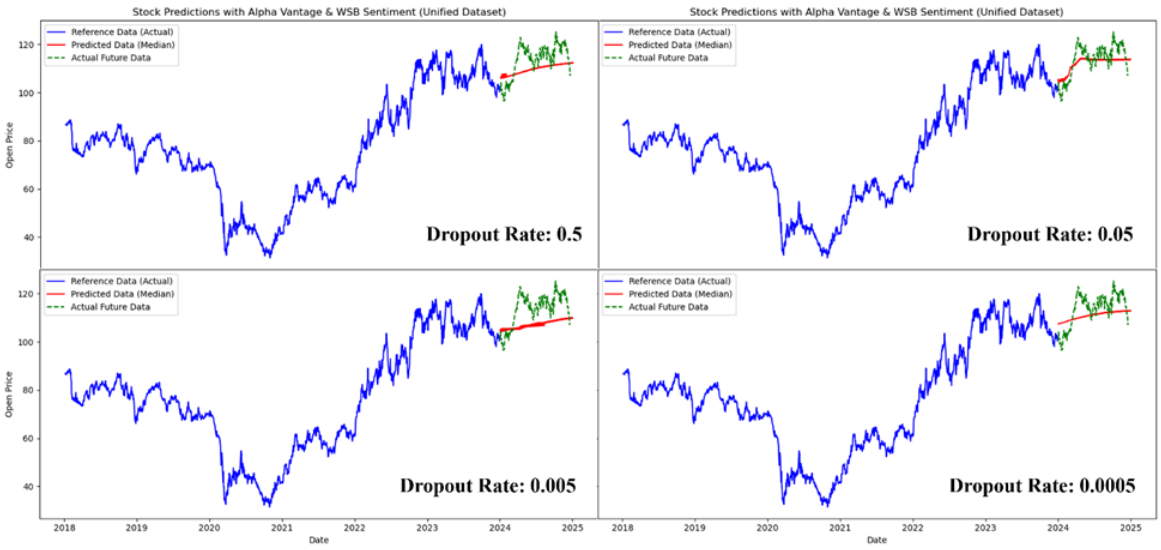

Figure 4. Demonstration of dropout rate impact on the performance

using the same *QuantileLoss* metrics as for the training data. Lower validation losses indicate stronger generalization. Decreasing validation loss over epochs indicates learning progress. Early stopping prevents overfitting.

If needed, additional evaluation metrics—such as MAE (Mean Absolute Error) or RMSE (Root Mean Squared Error)—can be computed to provide more interpretable error measures. By analyzing these metrics, we can gauge whether the model is capturing relevant patterns and if the incorporation of sentiment and fundamental data improves its predictive capacity.

3. Prediction and Visualization

3.1. Interface for Prediction

We visualize historical data, median predictions, and uncertainty intervals on the same plot. Utilizing PySimpleGUI, an interactive graphical interface allows users to dynamically select stocks and date ranges, and to view the resulting prediction curves and confidence intervals in real time. As shown in Figure 5, the PySimpleGUI offers extra convenience and flexibility to predict different range of stock trend.

3.2. One Stock Sample Results

We started training on the most basic example, i.e., one single stock, before extending the training to a large batch of stocks for a generalized model. When trained on a single stock (e.g., XOM), the model learns patterns specific to that stock's historical price movements. We used the methods illustrated in Method Section, the TFT can produce forecasts

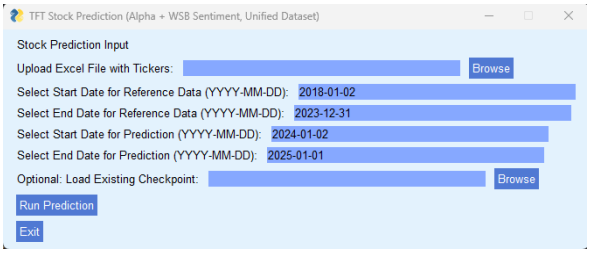

Figure 5. Example of GUI interface

that closely match future movements for short horizons. The quantile loss decreases over epochs, as shown in Figure 6, indicating improved predictions.

3.3. Multiple Stocks Results

When trained on multiple stocks (here, we use 50 as an example), the model treats all stocks as part of a large, multi-series dataset. In this report, we used 50 stocks as examples, randomly chosen from NSYE or NASDAQ listed energy-section stocks with more than 1000M market cap. By training a large batch of stocks, the TFT model can learn general market behaviors and sector-wide (energy section in this case) trends, leading to better generalization and improved forecasts, especially for less-liquid or more volatile stocks. In addition, training on many series can prevent overfitting to a single stock's idiosyncrasies and improve the robustness of predictions. As shown in Figure 7, after training on 50 stocks, the model gives a more generalized prediction of XOM future price (bottom figure) rather than the previously XOM-only trained result. (upper figure)

After training on data from 50 different stocks, the model

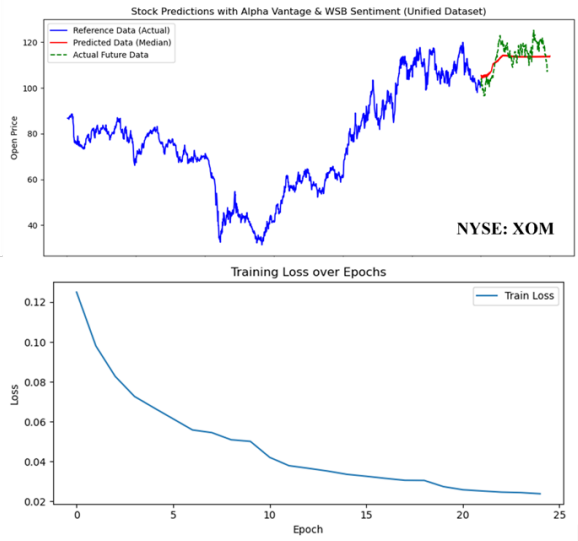

Figure 6. Results of XOM (Exxon Mobil) stock price prediction and corresponding training loss over epochs

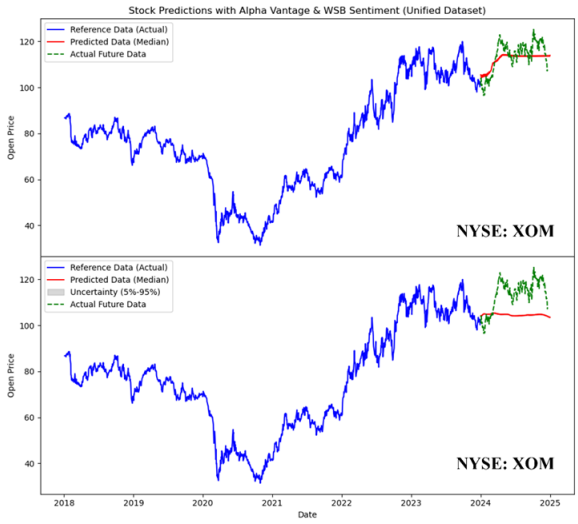

Figure 7. Results of XOM (Exxon Mobil) stock price prediction by TFT model trained on XOM historic training data (upper) and 50-stock-generalized training data (bottom).

demonstrates strong generalization capabilities, as illustrated in Figure 8. The left panels show stocks included in the training set, where the model's forecasts closely mirror actual future price movements. In contrast, the stocks depicted on the right were not part of the training data, yet the model still captures their overall trend and approximate price levels. These findings suggest that the TFT model effectively leverages its training on a diverse stock set to generate reasonable predictions for both known and previously unseen companies. While predictions for training-set stocks tend to align more closely with real outcomes, the model's performance on new stocks remains robust, reflecting its capacity for broader generalization.

However, a large training batch also leads to increased complexity of the model, which can increase training time and costs. The hyperparameters may also need to be further tuned to handle the complexity and ensure that the model capacity is sufficient to learn from all these stocks.

4. Conclusion

This report detailed a TFT-based model for stock price forecasting. By refining model architecture, tuning hyperparameters, and integrating multiple data sources, we demonstrated the model's potential to capture complex market dynamics and produce forward-looking price trajectories of energy section stocks.

5. Future Work

In this report, we have presented a methodology for forecasting stock prices using the Temporal Fusion Transformer (TFT) model with the integration of historical prices, fundamental metrics, and sentimental data from both financial news (Alpha Vantage) and social media (Reddit's Channel WallStreetBets). By preprocessing data and merging sentiment signals, the resulting Dataset supports more comprehensive information about our interested energy stocks.

As shown in Section 2.5, hyperparameter tuning plays an important role in improving model performance. Careful tuning of learning rate, dropout, hidden size, encoder/decoder lengths, and LSTM layer depth allows the model to strike a balance between complexity and generalization capabilities. Although initial tuning efforts were largely manual, future work would benefit from a more systematic approach, such as Bayesian optimization, grid search, or evolutionary algorithms.

In addition to model-level improvements, expanding the variety and granularity of input data offers substantial potential. Incorporating macroeconomic variables, such as interest rates, inflation, GDP, and commodity prices including oil and natural gas, may capture broader market influences. In the meantime, higher-frequency sentiment data and multilingual news analysis could better reflect short-term market reactions and global investor sentiment. While integrating data from non-traditional sources like Twitter or TikTok could increase costs (A LOT), the potential insights into rapid sentiment shifts may justify these investments.

Scaling the model to encompass thousands of tickers across multiple sectors and regions represents another promising direction. (Due to the time limitation of this current project, we could not finish the primary bold goal of making a generalized model for predicting all the tickers, but we feel very

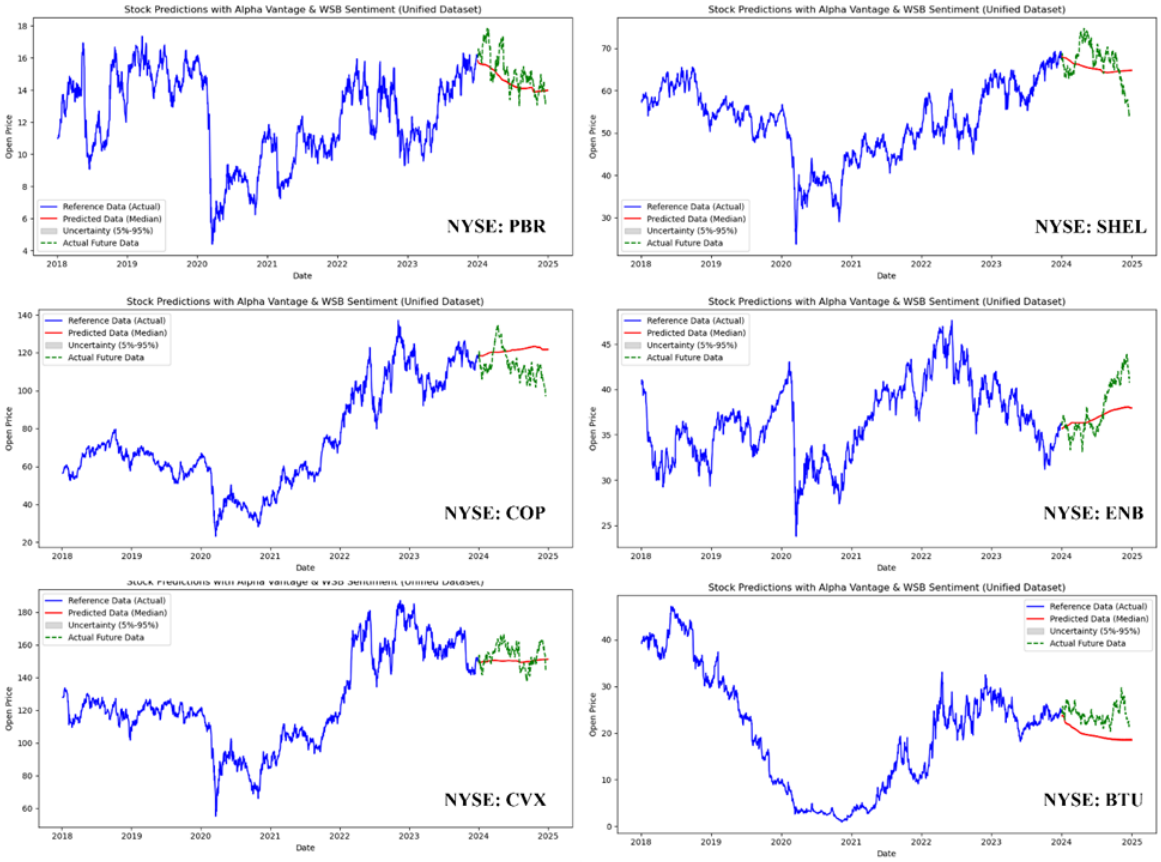

Figure 8. Predictive results for randomly selected stocks. The three stocks on the left were included in the set of 50 training stocks, while the three on the right were never seen during training, illustrating the model's ability to generalize to new, unseen stocks.

excited about it and will keep working on this; please keep tuned to our GitHub). Broader training sets could enable the TFT to discern global patterns, sector-wide behaviors, and cross-market dependencies, ultimately improving the robustness and relevance of its predictions. Such scalability could empower investors and analysts to better anticipate sector rotations, identify international market trends, and allocate capital more strategically.

6. Data Availability

The code with data is available at: https://github.com/kaishen526/stocktft.git

7. Acknowledgments

We gratefully acknowledge Yusheng Cai for insightful discussions and Voltage Park for providing GPU cloud resources, which were essential for completing this work.

References

Adebiyi A Ariyo, Adewumi O Adewumi, and Charles K Ayo. Stock price prediction using the arima model. In *2014 UKSim-AMSS 16th international conference on computer modelling and simulation*, pages 106–112. IEEE, 2014.

Dinesh Bhuriya, Girish Kaushal, Ashish Sharma, and Upendra Singh. Stock market predication using a linear regression. In *2017 international conference of electronics, communication and aerospace technology (ICECA)*, volume 2, pages 510–513. IEEE, 2017.

Xiaokang Hu. Stock price prediction based on temporal fusion transformer. In *2021 3rd International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI)*, pages 60–66. IEEE, 2021.

Bryan Lim, Sercan Ö Arık, Nicolas Loeff, and Tomas Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4):1748–1764, 2021.

Bowen Liu, Pawel Szalachowski, and Jianying Zhou. A first look into defi oracles. In *2021 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS)*, pages 39–48. IEEE, 2021.

Sreelekshmy Selvin, R Vinayakumar, EA Gopalakrishnan, Vijay Krishna Menon, and KP Soman. Stock price prediction using lstm, rnn and cnn-sliding window model. In *2017 international conference on advances in computing, communications and informatics (icacci)*, pages 1643–1647. IEEE, 2017.