Team07
Forecasting Gold Price Movements in India using Language Models and Sentiment Analysis

Adithya Jayan[a;*], Deepak Kumar[a;], Jaison Fernandez[a;***], Mohan P[a;****], Prayaga Nagasree Tejashwini[a;*****], Sai Yaswanth Divvela[a;******] and Tarun Vinjamuru[a;*******]**

[a]MTech Online, Indian Institute of Science, Bangalore

Abstract. This project presents a language model-based automation workflow for forecasting gold price movements in India, leveraging sentiment signals extracted from financial news, stock market data, and technical indicators such as moving averages and RSI. While sentiment-driven models are widely used in equity markets, their application in gold price movement prediction is underexplored.

The proposed methodology orchestrates a system capable of performing data collection, web scraping, preprocessing, sentiment analysis, LLM encoder-based sentiment extraction, and gold price movement assertion. The system includes a custom fine-tuned language model trained locally on domain-specific data, integrated into an orchestration framework for real-time inference. The performance of the model is also benchmarked against standard baseline models.

1 Introduction

Gold plays a critical role in the Indian economy as both a cultural commodity and a financial instrument. Its price is influenced by a complex interplay of global economic indicators, investor sentiment, and local market dynamics. Although traditional time-series models have been widely used for forecasting gold prices, they are not capable of capturing information embedded in financial news and understanding market sentiment. Using large language models (LLMs), we can extract key topics and sentiments from unstructured text data, which can improve prediction accuracy.

This project uses key topics and their sentiment signals derived from financial and stock market news to forecast short-term gold price movements in India. We propose an Orchestration framework that integrates structured financial data and unstructured news content using langchain to predict gold price movement.

The workflow contains English and Indic news scrapers, a price data scraper, and domain-specific technical indicators. News data is processed using supervised fine-tuned pre-trained LLMs to extract topic-specific sentiment features. These features are then fed into a custom set-transformer model trained to predict gold price direction. Additionally, we use traditional baseline statistical and ML/DL models that are trained solely on historical price data. The outputs of these models are combined using a meta-model to generate the final prediction.

This workflow enables the integration of market indicators with sentiment features, for efficient price forecasting. We benchmark our model against standard baseline predictors and demonstrate it's value.

* Email: adithyajayan@iisc.ac.in
** Email: deepakkumar6@iisc.ac.in
*** Email: jaisonf@iisc.ac.in
**** Email: mohanpanakam@iisc.ac.in
***** Email: prayagan@iisc.ac.in
****** Email: saiyaswanthd@iisc.ac.in
******* Email: tarunv@iisc.ac.in

2 Methodology

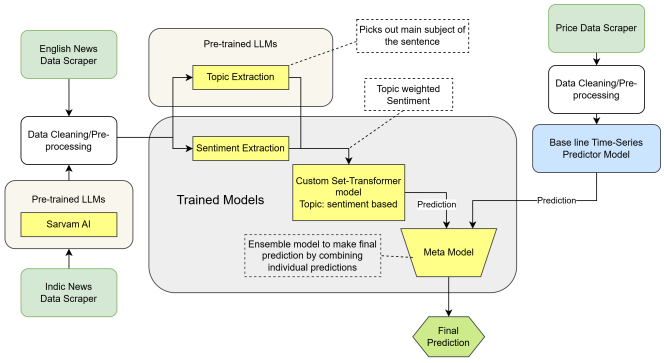

Figure 1. Chain Functional Block Diagram

The overall chain is designed to handle every step of the flow starting from news-data-scraping, processing, prediction, to fine-tuning for periodic self-improvement. Figure 1, provides a visual representation of the same. The principal components present in the chain are listed below.

2.1 News data scraping

We need a comprehensive dataset of financial news due to its relevance to gold price movements. We implemented a set of web scraping functions targeting multiple reputable sources to systematically collect, processes, and consolidate news articles from BullionVault, Yahoo Finance and Reuters. In addition to the English news, we are scraping local language news (like BBC Telugu) and using SARVAM

AI to translate local languages to english, thus seamlessly integrating this into our developed orchestration while ensuring linguistic consistency. This helps to capture the regional perspectives. Detailed information in Appendix A.

2.2 *Gold price database update*

To obtain historical gold price data for modeling and analysis, we retrieved the daily trading data of the GOLDBEES Exchange Traded Fund (ETF) listed on the Bombay Stock Exchange (BSE) under the ticker GOLDBEES.BO. Detailed information in Appendix C.

2.3 *Data cleaning and Pre-processing*

News articles and historical gold price data were sourced from public repositories, but inconsistencies required preprocessing. Irrelevant prefixes were removed, text was standardized, and incomplete entries were filtered. Dates were uniformly formatted to align news with price data. Articles were segmented into chunks preserving sentence boundaries, and directional price movements (up, down, unchanged) were computed to label each article. The resulting dataset pairs cleaned news content with corresponding price movement labels for model training and evaluation. Detailed information in Appendix B.

2.4 *Topic extraction*

We use a variation of 'Aspect based' sentiment in our flow to predict change in asset price. We opted to use a 'Universal Sentence Encoder' [10] for representing the topic associated with each input text vector. The UAN converts the text-vectors into 256-long encodings. In this encoded space, articles with similar and related content have similar representation. An example is shown in Figure 2. We can observe that similar contexts have similar representations, allowing for better generalization in impact of such articles.

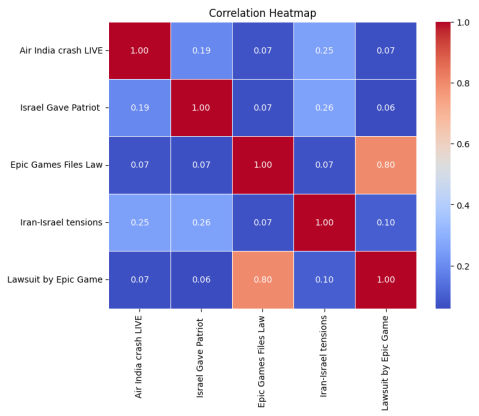

Figure 2. Correlation plot of various news encodings.

This allows the predictor model to learn meaningful relations and generalize on how different types of news will impact the overall asset price. This also helps the model decide relative importance and impact between different articles whose associated sentiments could have opposite impacts on the price. Detailed information in Appendix D.

2.5 *Model Finetuning and Sentiment extraction*

To adapt the `ProsusAI/finbert` model—a BERT-based model pretrained on financial texts. This finetuning specifically targets the **gold market domain(goldbert)**, allowing the model to learn sentiment patterns and context relevant to gold-related financial discussions. This specialization enhances its predictive accuracy compared to relying solely on the globally pretrained FinBERT. —for our domain-specific sequence classification task, we utilized **Parameter-Efficient Fine-Tuning (PEFT)** through **Low-Rank Adaptation (LoRA)**.

These adapters were injected into the attention projection layers of the model. During training, only the LoRA adapter weights were updated, while the original FinBERT parameters were frozen.

This PEFT strategy provided several practical benefits:

- Efficiency: Only a small fraction ($\sim$0.1–1%) of model parameters are trainable.
- Memory Savings: Lower GPU memory footprint compared to full fine-tuning.
- Modularity: LoRA adapters can be saved, reused, or merged across tasks.
- Performance: Achieves accuracy comparable to full fine-tuning with better generalization in low-resource settings.

This approach enabled robust fine-tuning of FinBERT on our specialized financial task while significantly reducing compute and memory overhead. The LoRA-based fine-tuning of FinBERT for financial sentiment classification yielded consistent, high-quality predictions across three folds, demonstrating:

- Stable cross-validation performance (Avg. Val F1: 91.51%)
- Efficient training with parameter-efficient tuning (LoRA)
- Minimal overfitting, as evidenced by matching train/val trends
- Scalable optimization verified through W&B runtime analytics

Detailed information as well as performance metrics are discussed in Appendix E.

2.6 *The prediction model*

The prediction-model is the final component on the sentiment based prediction branch before being combined with the time-series model predictions at the meta-model.

Here, we use the topic-weighted sentiment values obtained by the previous blocks in the chain to predict the relative price percentage change in gold close-price for the next day. We make use of the *Set Transformer architecture* [5] to perform this prediction.

The set transformer model being an encoder-decoder-based transformer architecture, takes variable length data for prediction, and it is specifically designed to be permutation invariant which is necessary as we use article data whose order has no relevance.

The final trained model displayed accuracy metrics as follows while predicting gold price for each day: MAE=0.46; RMSE: 0.63; R^2: 0.993; MAPE: 0.70. Overall performance was below expectation. We believe this can be attributed to a lack of sufficient scraped text corpus for training. Detailed information in Appendix F.

2.7 *Statistical, machine learning and deep learning prediction models*

To reinforce the **prediction capability of the Set Transformer-based model**, we employ a meta-ensemble approach that integrates

outputs from statistical, machine learning, and deep learning models—including SARIMAX, XGBoost, Random Forest, LSTM—as well as the Set Transformer itself. Detailed information in Appendix G.

2.8 *Meta-Ensemble (Ridge Regression) Model*

This meta-model aggregates the outputs of multiple base models: SARIMAX, XGBoost, Random Forest, LSTM and set transformer using a regularized linear regression approach(used RidgeCV). This ensemble strategy leverages the strengths of individual models and mitigates their weaknesses. Detailed information in Appendix H.

2.9 *Feedback loop*

The meta-ensemble model weights are re-evaluated daily using rolling windows of the most recent 100 days of predictions, including backtesting data. This dynamic updating process serves as a feedback mechanism for the base models.

Possible future improvements to the feedback loop to improve performance overtime is discussed in Section 5.

3 Results and analysis

Comprehensive results and outputs from each of the pipeline stages are provided in the appendices at the conclusion of the paper. A summary is provided below.

3.1 *Set-Transformer vs Statistical, Machine Learning, and Deep Learning Prediction Models*

The bar chart in Figure 3 presents the predicted Gold ETF prices generated by multiple models—ARIMAX, XGBoost, Random Forest, LSTM, and Set-Transformer—alongside the aggregated prediction produced by a meta-ensemble model.

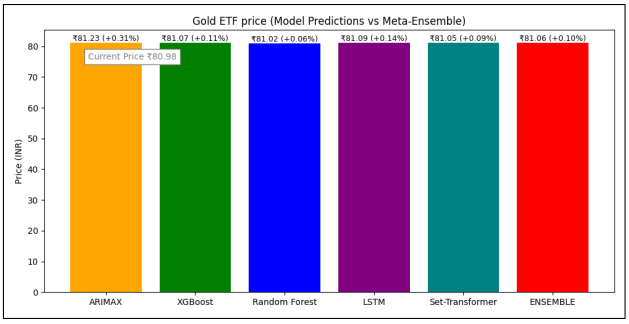

Figure 3. Gold ETF price prediction across models and meta-ensemble

3.2 *SARIMAX Model: Actual vs. Predicted (300-Day Backtest)*

Table 1 presents a comparative analysis of four gold price prediction models—ARIMAX, LSTM, Random Forest, and XGBoost—over a 300-day backtest period. More details on the evaluation metrics can be found in Appendix I.

More details on the SARIMAX Model: Actual vs. Predictedcan be found in Appendix J.

Table 1. Performance Comparison of Gold Price Prediction Models (300-Day Backtest)

Model	Cur. ()	Pred. ()	%Chg	MAE	RMSE	DirAcc (%)
ARIMAX	82.64	83.01	+0.45	0.27	0.39	81.21
LSTM	82.23	81.38	-1.03	0.97	1.38	49.16
Random Forest	82.64	82.63	-0.01	0.60	0.80	50.50
XGBoost	82.23	82.91	+0.82	0.62	0.81	52.51

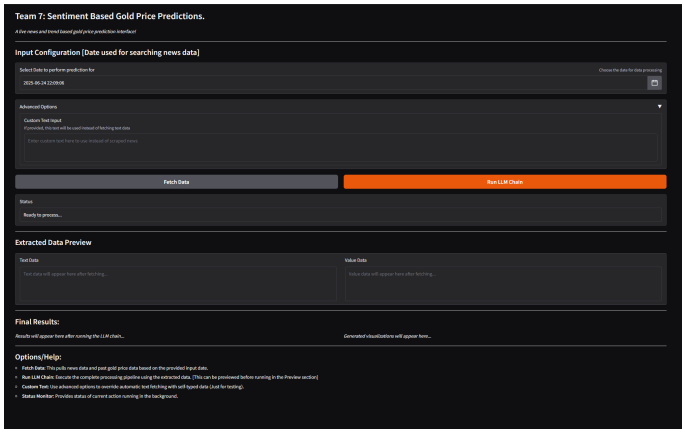

Figure 4. GUI User Interface

3.3 *Performance*

An interactive browser based GUI interface to facilitate running and usage was implemented as seen in Figure 4. Final integration with the overall lang-chain framework was partially completed.

3.4 *Implementation and Code*

All code and resources related to the project can be found in the github repository: [https://github.com/dsygnt11-iisc/DL-7-25].

4 Conclusion

This work presents a comprehensive framework for forecasting short-term gold price movements in India by leveraging sentiment signals extracted from financial news and technical indicators. By combining web scraping, sentiment analysis using large language models, and structured time-series data, overall chain augments sentiment information with quantitative financial trends.

Overall, this study highlights the potential of language model-driven approaches in commodity forecasting and opens possibilities for further exploration of multilingual sentiment fusion, model explainability, and real-time market applications.

5 Future scope

Potential future scope of this work can be include

1. Expanding data sources to incorporate multilingual news, social media sentiment, and macroeconomic indicators for better diversity.
2. Enhancing the model architecture with temporal attention could better capture interactions between sentiment and price trends
3. Regression-based forecasting can provide finer-grained predictions.

4. Real-time deployment, backtesting within trading simulations and use of Broker's MCP for order placement would increase practical utility.
5. Targeted news scraping based on performance analysis and correlation studies on keywords used for scraping vs performance. This can be incorporated as part of the feedback loop.

6 Individual Contributions

All team members contributed actively throughout each phase of the project, including model development, training, evaluation, and documentation. Additionally, the following individuals demonstrated exceptional initiative and made significant contributions in specific areas:

Adithya Jayan Dataset preparation for training, research and implementation of news-topic-context informatio extraction from news data. Architecture design to combine sentiment and context encodings for price-prediction and it's implementation and training. Gradio based GUI-template for integration with overall functional chain.

Deepak Kumar In Data Cleaning: Used pandas to load, inspect, and clean raw .csv datasets. Applied regular expressions (re module) to remove unwanted text like "Title:" prefixes. Used string methods to normalize case, spacing, and remove null or non-string content. Converted dates into datetime format using pd.to_datetime(). Integrated cleaning inside automated pipelines (e.g., LangChain).

In Data Pre-processing: Defined sentence segmentation and chunking logic to split long texts into 1500-character segments for transformer models (approx. 512 tokens). Applied heuristic sentiment labeling via keyword matching. Merged cleaned text with historical gold price data using pandas.merge() and aligned by date. Created a final labeled dataset with "date", "content", and "label" columns.

Jaison Fernandez The overall system architecture includes pathfinding (using the Filbert sample test), gold price action extraction, and technical analysis-based feature generation. It incorporates statistical, machine learning, and deep learning models—including SARIMAX, XGBoost, Random Forest, and LSTM—for predictive modeling. Each model is defined, trained, and backtested against actual price data to evaluate performance. An ensemble model is also constructed and trained to combine individual predictions for enhanced accuracy.

Mohan P Sentiment extraction from news and provided sentiment, probabilities of classes and confidence. Fine tuned Finbert-model using PEFT (LoRa) method, tuned hyper-parameters and extracted required metrics for analysis. Trained GoldBert model can be used to generate sentiments by giving gold news as an input and based on gold price data from scraped news data.

Prayaga Nagasree Tejashwini In Data Cleaning: Used pandas to load, inspect, and clean raw .csv datasets. Applied regular expressions (re module) to remove unwanted text like "Title:" prefixes. Used string methods to normalize case, spacing, and remove null or non-string content. Converted dates into datetime format using pd.to_datetime(). Integrated cleaning inside automated pipelines (e.g., LangChain).

In Data Pre-processing: Defined sentence segmentation and chunking logic to split long texts into 1500-character segments for transformer models (approx. 512 tokens). Applied heuristic sentiment labeling via keyword matching. Merged cleaned text with historical gold price data using pandas.merge() and aligned by date. Created a final labeled dataset with "date", "content", and "label" columns.

Sai Yaswanth Divvela Developed a suit of web scrapping tools to extract and consolidate English and Regional NEWS for training and validation data and real-time inference.

Tarun Vinjamuru Implemented the LangChain-based workflow for a multi-model gold price prediction system. Integrated independently developed model components into an execution pipeline using SequentialChain and TransformChain. I handled all environment setup, model loading, and dynamic path generation, ensuring smooth end-to-end execution. I optimized data handling, maintained modularity, and ensured compatibility across inputs and outputs.

Acknowledgements

We would like to thank Prof. Deepak Subramani for guidance through the course.

References

[1] Dogu Araci, 'FinBERT: Financial sentiment analysis with pre-trained language models', (August 2019).

[2] Rohit Arora. Fun with Topic Extraction — medium.com. https://medium.com/analytics-vidhya/fun-with-topic-extraction-8aa11e0437d0, 2021.

[3] Dávid Javorský, Ondřej Bojar, and François Yvon. Assessing word importance using models trained for semantic tasks, 05 2023.

[4] Juho Lee. GitHub - juho-lee/set_transformer: Pytorch implementation of set transformer — github.com. https://github.com/juho-lee/set_transformer.

[5] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R Kosiorek, Seungjin Choi, and Yee Whye Teh, 'Set transformer: A framework for attention-based permutation-invariant neural networks', (October 2018).

[6] Daniel Melchor. A detailed guide to Pytorch's nn.Transformer() module. | Towards Data Science — towardsdatascience.com. https://towardsdatascience.com/a-detailed-guide-to-pytorchs-nn-transformer-module-c80afbc9ffb1/, 2021.

[7] Seongsik Park and Harksoo Kim, 'Effective sentence-level relation extraction model using entity-centric dependency tree', *PeerJ Comput. Sci.*, **10**, e2311, (September 2024).

[8] Ankur Sinha. Sentiment analysis of commodity news. https://www.kaggle.com/code/ankurzing/sentiment-analysis-of-commodity-news/input. Kaggle.

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin, 'Attention is all you need', (2017).

[10] Georg Wiese. Enhancing Intent Classification with the Universal Sentence Encoder — medium.com. https://medium.com/webknossos/enhancing-intent-classification-with-the-universal-sentence-encoder-ecbcd7a3005c, 2018.

[11] Haolun Wu, Ye Yuan, Liana Mikaelyan, Alexander Meulemans, Xue Liu, James Hensman, and Bhaskar Mitra, 'Learning to extract structured entities using language models', (February 2024).

[12] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola, 'Deep sets', in *Advances in Neural Information Processing Systems*, eds., I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, volume 30. Curran Associates, Inc., (2017).

A News Data scraping

News articles are scraped from BullionVault, Yahoo Finance, Reuters, and BBC Telugu. For each source, a dedicated scraping function was developed to extract article titles, publication dates, and full article text.

The get_latest_bullionvault_articles function retrieves the most recent articles from BullionVault's gold news section. Similarly, get_latest_yf_articles uses Selenium to dynamically load and parse Yahoo Finance news, handling JavaScript-rendered content. For Reuters, the get_latest_yf_articles and its sub-functions crawl the site's search results for gold-related news, following links to individual articles to get the data.

To incorporate regional perspectives, the fetch_bbc_telugu_news function scrapes popular news from BBC Telugu. Since these articles are in regional language (here Telugu), we employ the Sarvam translation API via the sarvam_translate_text function to automatically translate content into English, ensuring linguistic consistency across the dataset.

All scraped articles are structured into pandas DataFrames with standardized columns for date and content. Dates are normalized to a consistent format.

B Data Cleaning and Pre-processing

News Data Cleaning Raw news data included inconsistencies such as static prefixes (e.g., "Title:"), erratic spacing, and casing mismatches. The following steps were applied:

- Removed fixed prefixes using regular expressions.
- Normalized all whitespace characters (e.g., tabs, multiple spaces).
- Converted all text to lowercase.
- Discarded empty, null, or non-string entries.
- Converted date fields into standardized datetime objects using the pandas library.

Text Chunking Given the input token limitation of transformer models (typically 512 tokens), news articles were split into chunks of up to 1500 characters. Sentence segmentation was performed using regex patterns to ensure semantic coherence, and short sentences were grouped together to retain context.

Price Data Matching Historical gold price data was matched with news articles by date. In cases where price data was missing for certain article dates, either forward-filling or row removal was used, depending on how continuous the data was. Price movements were computed by comparing closing prices across consecutive days. Articles labeled with price direction (increase, decrease, unchanged) were used for training models.

Sentiment Labeling Two methods were employed to perform sentiment annotation:

- If a dataset had an existing sentiment column (e.g., *Price Sentiment*), the labels in this column were mapped to numerical values: positive $\rightarrow$ 0, neutral or none $\rightarrow$ 1, negative $\rightarrow$ 2.

- The heuristic rule-based approach was used for unlabeled data. Sentiment was determined by the occurrence of specific financial trading keywords such as *"bullish"*, *"surge"*, and *"safe-haven"* (positive), or *"fall"*, *"bearish"*, and *"decline"* (negative). Content lacking these indicators was labeled as neutral.

This ensured that every processed text chunk was uniformly labeled and suitable for input to sentiment classification models.

Final Dataset Structure Each record in the final processed dataset contained three key elements:

- A cleaned and chunked portion of a financial news article,
- Its corresponding sentiment label, encoded as an integer (0, 1, or 2),
- A directional gold price movement label, indicating whether the market moved upward, downward, or remained unchanged on the corresponding date.

This structured format ensured consistency and alignment across all records, enabling effective supervised learning for both sentiment classification and financial trend prediction tasks.

C Gold price database update

This extraction was accomplished using the yfinance Python library, which provides programmatic access to Yahoo Finance data. The dataset includes key financial indicators such as Open, High, Low, Close, and Volume for each trading day within the specified date range. The raw data is preprocessed, formatted into a flat structure if necessary, and saved locally in CSV format for further use in the modeling pipeline.

D Topic extraction

Problem In traditional aspect-based sentiment analysis, sentiments related to a particular 'aspect' is extracted from the text corpus. In our project, our aspect of interest is 'Impact on gold price'. This collection of extracted sentiment values alone lacks information on the relative impact of different news-topics on our prediction-target. Furthermore, having a pre-decided set of *High impact topics*, to extract sentiment based on - would restrict the insights from uncommon or un-covered areas and topic.

Our Solution To help form a better representation of the context associated with each extracted gold price sentiment, we perform *Topic extraction* on each input text vector. This *topic*, in combination with it's related sentiment is then combined to give an encoded context-sentiment encoding. A collection of such encodings corresponding to a single day can then be used for prediction.

Representation method Our initial approach of extracting topic or context in terms of tokenized words [2] proved inefficient as words-tokens belong to a high-dimensional space which would reduce model generalization in categorizing topic impact. Furthermore, words alone often lack context, and using phrases instead of words to improve context would only further increases the dimensionality and hence reduce the generalization.

To overcome this, we opted to use a 'Universal Sentence Encoder'[10].

E Sentiment extraction

LoRA Configuration. We applied LoRA using the Hugging Face `peft` library with the following settings:

- `r = 8`
- `lora_alpha = 16`
- `target_modules = ["query", "key", "value"]`
- `lora_dropout = 0.1`
- `bias = "none"`
- `task_type = TaskType.SEQ_CLS`

Training Hyperparameters. Fine-tuning was performed using the Hugging Face `Trainer` API with the following configuration:

- `per_device_train_batch_size = 4`
- `per_device_eval_batch_size = 2`
- `num_train_epochs = 5`
- `learning_rate = 5e-5`
- `warmup_steps = 500`
- `weight_decay = 0.1`
- `logging_steps = 500`
- `eval_steps = 1500`
- `save_steps = 1500`
- `evaluation_strategy = "steps"`
- `save_strategy = "steps"`
- `load_best_model_at_end = True`
- `metric_for_best_model = "f1"`
- `fp16 = False` (disabled for MPS compatibility)

K-Fold Cross-Validation Performance To assess the generalizability of the LoRA-fine-tuned `ProsusAI/finbert` model, we conducted a 3-fold cross-validation on our labeled sentiment classification dataset. The evaluation was carried out for both training and validation sets, and performance metrics were recorded at each fold.

Table 2. Fold-wise Validation Metrics

Fold	Accuracy (%)	F1 Score (%)	Precision (%)	Recall (%)
Fold 1	90.64	89.52	89.80	89.27
Fold 2	92.03	91.15	91.32	90.99
Fold 3	94.35	93.86	94.05	93.68
Average	**92.34**	**91.51**	**91.72**	**91.31**

These results demonstrate consistent and strong performance across all folds. The model achieves an average validation accuracy of **92.34%** and F1 score of **91.51%**, confirming high generalizability. Despite a slight drop in the *neutral* class, overall performance remains stable across sentiment categories.

Final Average Metrics Summary After aggregating performance across all folds, we report the following consolidated results:

- Average Training Accuracy: 93.59%
- Average Training F1 Score: 92.93%
- Average Validation Accuracy: 92.34%
- Average Validation F1 Score: 91.51%

These metrics confirm strong agreement between training and validation performance, with minimal overfitting and well-maintained precision-recall tradeoffs.

W&B Training Dynamics To monitor training behavior and optimization progress, key metrics were logged using Weights & Biases (W&B). The following insights were derived from the visualizations:

- Training and Evaluation Loss: Both training and validation loss decreased over time, indicating effective convergence without significant overfitting.
- F1 and Accuracy Trends: Evaluation F1 and accuracy consistently improved with training steps, peaking in the final epochs.
- Learning Rate Schedule: A linear decay schedule was applied, decreasing from 5×10^{-5} to zero across 5 epochs.
- Gradient Norm Behavior: Spikes in gradient norm indicate optimizer steps, but values remained bounded, reflecting stable optimization under LoRA tuning.
- Throughput: Metrics such as `eval/steps_per_second` and `samples_per_second` remained steady, confirming computational efficiency and no hardware bottlenecks.

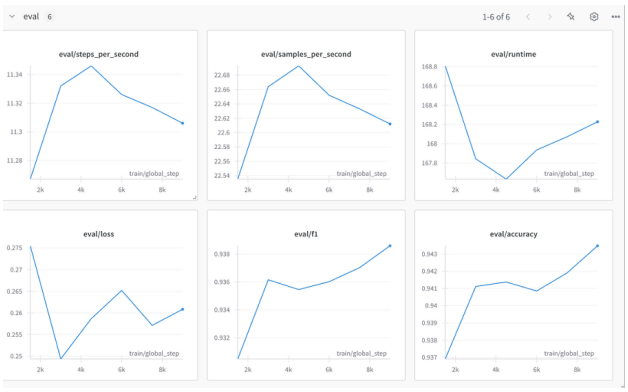

Figure 5. W&B - Evaluation Metrics vs. Global Steps

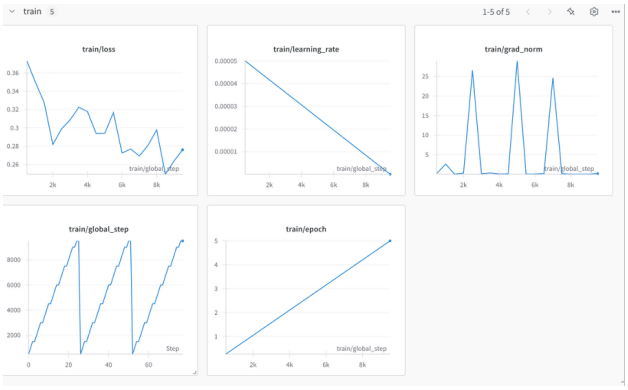

Figure 6. W&B - Training Metrics vs. Global Steps

These results support the claim that LoRA is an effective strategy for fine-tuning large transformer models with minimal compute cost, especially in low-resource or domain-specific scenarios.

F Prediction Model

Set-Transformer architecture The choice of this architecture is influenced by two main reasons.

1. *The number of news snippets per day is variable*: We want to make use of all the information available in one day to perform the day's prediction.
2. *The order of the articles-snippets has no relevance*: Traditional auto-regressive models capable of taking sequential data are not order-agnostic.

The set transformer model solves both of the above issues. Being an encoder-decoder-based transformer architecture, it can take variable length data for prediction. Further, the set-transformer is specifically designed to be permutation invariant.

Training Training of the model was done by first annotating the overall training news corpus with it's associated sentiment and context/topic encoding. These were then combined and grouped by date - to produce sets of 256 length sentiment-weighted-topic-encoding values.

These sets being of variable length, were then padded with uniformity. The original model [4] was updated to accept an additional masking argument - to mask out these padding during the forward pass. Additionally, the ground truth is also obtained for the corresponding dates, and relative price change is calculated for adjacent days in terms of percentage change. This is important as the sentiment based predictor model generally has no idea of the current gold price while making a prediction, hence all prediction made is relative-change.

We use early-stopping to prevent overfitting of the model on the data by monitoring the validation loss. Figure 7 shows the training and validation losses over 30 epochs. Our final model training trained to only 10 epochs to prevent overfitting.

Performance The overall final predicted price vs true price using just the sentiment-based chain is provided in Figure 7.

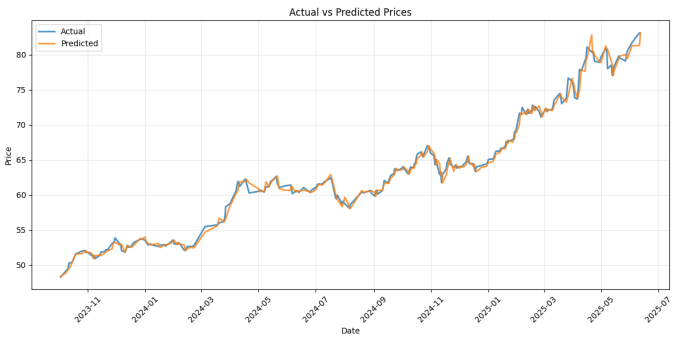

Figure 7. Pure sentiment based price prediction.

However, the value predicted by the model is the relative-price-percentage change, which when combined with the very slow rate of chaange in gold - leads to near tracking ever if the model underperforms as observed in **??**.

Performance can potentially be improved by scraping a larger news-text dataset for training. The current model uses a data-corpus that is a combination of scraped-news data, combined with annotated news text dataset obtained from Kaggle [8]. We observe that the model over fits easily on our current training dataset - hence it is likely that it is not generalizing enough due to lack of data. This can be taken as a future scope for exploration and study.

G Statistical, machine learning and deep learning prediction models

- **SARIMAX:** A classical statistical time series model that captures linear trends, seasonality, and temporal dependencies. It extends the ARIMA framework by incorporating exogenous variables (denoted as X_t), such as sentiment scores. The general form is:

$$y_t = \mu + \sum_{i=1}^{p} \phi_i y_{t-i} + \sum_{j=1}^{q} \theta_j \epsilon_{t-j} + \sum_{k=1}^{K} \beta_k X_t^{(k)} + \epsilon_t,$$

  where y_t is the gold price at time t, ϕ_i and θ_j are the autoregressive (AR) and moving average (MA) coefficients, $X_t^{(k)}$ represents the kth exogenous feature (e.g., sentiment), and ϵ_t is the error term.
- **XGBoost:** An efficient and scalable machine learning algorithm based on gradient boosting of decision trees. It builds an ensemble of weak learners sequentially, where each tree learns to correct the errors of the previous one. The prediction for gold price $\hat{y}_t$ is given by:

$$\hat{y}_t = \sum_{k=1}^{K} f_k(x_t), \quad f_k \in \mathcal{F},$$

  where x_t is the feature vector at time t (e.g., technical indicators and sentiment scores), f_k denotes the k-th regression tree, and $\mathcal{F}$ is the space of all possible trees. The model optimizes a regularized objective function:

$$\mathcal{L} = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k),$$

  where l is a differentiable loss function (e.g., squared error), and $\Omega(f_k) = \gamma T + \frac{1}{2}\lambda \sum_j w_j^2$ penalizes model complexity to avoid overfitting.
- **Random Forest:** A machine learning ensemble method that builds multiple decision trees using bootstrapped subsets of the data and aggregates their outputs to improve prediction accuracy and reduce overfitting. For gold price prediction, the model outputs the average of K independently trained trees:

$$\hat{y}_t = \frac{1}{K} \sum_{k=1}^{K} T_k(x_t),$$

  where x_t is the input feature vector at time t (e.g., historical prices and sentiment scores), T_k represents the kth decision tree, and $\hat{y}_t$ is the predicted gold price. Random feature selection at each split introduces decorrelation between trees, enhancing generalization performance and robustness to noisy inputs.
- **LSTM:** A type of recurrent neural network (RNN) designed to capture long-term dependencies in sequential data, making it well-suited for modeling time series such as gold prices influenced by historical trends and sentiment dynamics. At each time step t, the LSTM updates its internal memory using a series of gating mech-

anisms:

$$
\begin{aligned}
f_t &= \sigma(W_f[h_{t-1}, x_t] + b_f) && \text{(forget gate)} \\
i_t &= \sigma(W_i[h_{t-1}, x_t] + b_i) && \text{(input gate)} \\
\tilde{C}_t &= \tanh(W_C[h_{t-1}, x_t] + b_C) && \text{(candidate state)} \\
C_t &= f_t \odot C_{t-1} + i_t \odot \tilde{C}_t && \text{(cell state update)} \\
o_t &= \sigma(W_o[h_{t-1}, x_t] + b_o) && \text{(output gate)} \\
h_t &= o_t \odot \tanh(C_t) && \text{(hidden state)}
\end{aligned}
$$

Here, x_t represents the input features at time t (e.g., gold price and sentiment), h_t is the hidden state, C_t is the memory cell state, and σ is the sigmoid activation function. The final prediction $\hat{y}_t$ is typically obtained through a dense output layer applied to h_t.

H Meta-Ensemble (Ridge Regression) Model

This meta-model uses RidgeCV, a cross-validated ridge regression model, to learn optimal weights to combine predictions. Given a feature matrix $X \in \mathbb{R}^{n \times m}$, where each column corresponds to a predicted gold price from a base model, and a target vector $y \in \mathbb{R}^n$ containing actual prices, the model solves:

$$\hat{y} = X\beta + \epsilon, \quad \text{with} \quad \beta = \arg\min_{\beta} \left\{ \|y - X\beta\|_2^2 + \lambda \|\beta\|_2^2 \right\},$$

where λ is the regularization parameter selected via cross-validation.

I Performance Evaluation Metrics

The following metrics were used to evaluate the predictive performance of the models:

% Change: Captures the relative percentage difference between the predicted and actual price, reflecting directional deviation.

MAE (Mean Absolute Error): Quantifies the average magnitude of absolute errors in prediction, offering a straightforward measure of model accuracy.

RMSE (Root Mean Squared Error): Emphasizes larger prediction errors by squaring deviations, providing insight into overall prediction reliability.

DirAcc (%)(Directional Accuracy): Reflects the percentage of times the model correctly predicts the direction of price movement, which is particularly useful for trading strategy evaluation.

J SARIMAX Model: Actual vs. Predicted (300-Day Backtest)

J.1 Meta-ensemble model weight training

Figure 8 below illustrates the meta-model weight distribution derived from 100 backtested predictions across various individual models. These weights represent the contribution of each base model and are utilized to compute the final ensemble prediction output.

Figure 9 shows the ARIMAX model's 300-day backtest performance, comparing predicted prices against actual Gold ETF prices. The model closely tracks the actual price trajectory, capturing both short-term fluctuations and long-term trends with high accuracy. The visual alignment between red (predicted) and blue (actual) markers demonstrates the effectiveness of ARIMAX in modeling temporal dependencies in gold price movements.

The plot in Figure 10 illustrates the day-wise prediction errors (Predicted − Actual) over a 300-day backtest period, highlighting

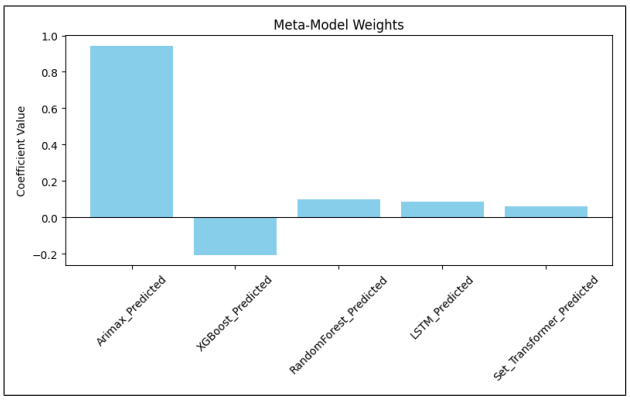

Figure 8. Meta-model weight distribution across individual models

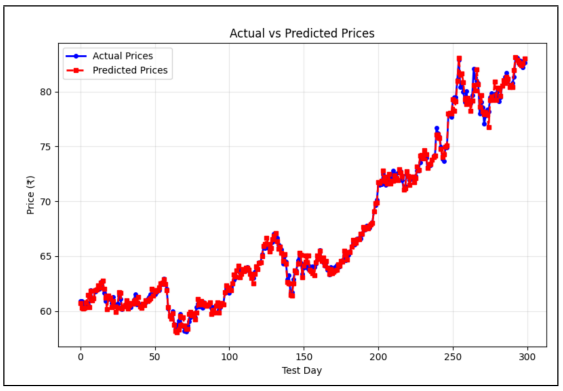

Figure 9. Actual vs. Predicted Gold ETF Prices Using the ARIMAX Model (300-Day Backtest)

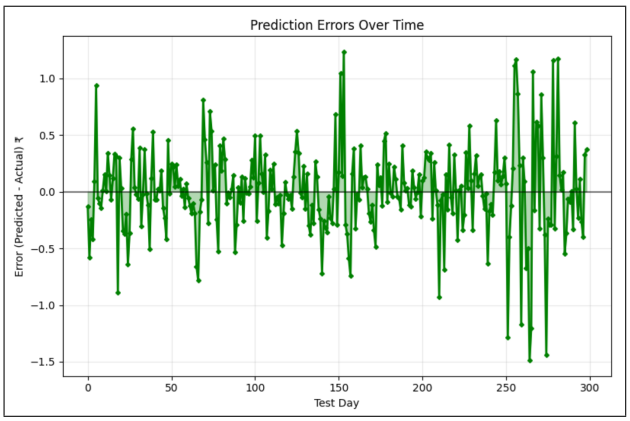

Figure 10. Prediction error over a 300-day backtest period using the ARIMAX model

the fluctuations and occasional spikes in SARIMAX model prediction deviations.

Figure 11 compares actual gold prices with predictions from four different models—ARIMAX, XGBoost, Random Forest, and LSTM—over a 300-day backtest period. All models demonstrate strong alignment with the actual price trend, although the LSTM model shows slightly higher volatility in short-term predictions.

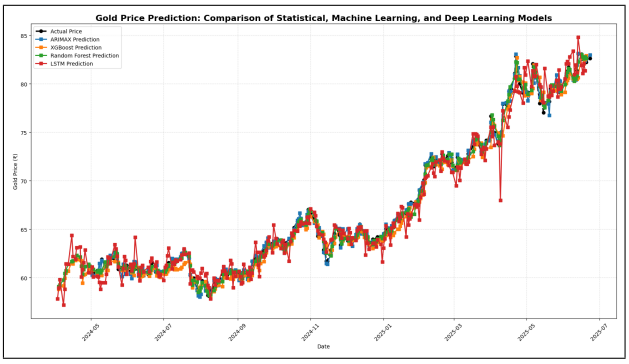

Figure 11. Comparison of actual gold prices with predictions from ARIMAX, XGBoost, Random Forest, and LSTM models over a 300-day backtest period