

Predicting Next-Trading-Session Stock Movements from Arabic Corporate Announcements: A Hybrid Transformer-Boosted Ensemble

Abdullah Almusned*, Mohammad Mehedi Hassan, Muhammad Al-Qurishi, Bader Alkhamees

Department of Information System, College of Computer and Information Sciences, King Saud University, Saudi Arabia

Abstract—Predicting next-day stock movements from corporate announcements is a complex task, especially in markets where financial text is published in low-resource languages such as Arabic. In this study, we develop a hybrid, multimodal forecasting framework designed to capture both the semantic depth of Arabic announcements and the quantitative structure of market behavior. The system combines three complementary components: a fine-tuned multilingual Transformer for extracting text representations, gradient-boosted trees for modeling financial features, and a temporal Transformer capable of learning sequential market patterns. Each model contributes regression and classification outputs, which are then integrated through a stacked ensemble. To mitigate noise from small price changes, the framework employs a gray-zone weighting scheme that reduces the impact of near-zero returns. Additionally, we calibrate predicted probabilities using isotonic regression to ensure stable confidence estimates and enable effective threshold tuning. We evaluate the proposed approach on more than 46,000 Arabic corporate announcements from the Saudi Stock Exchange (Tadawul). The ensemble consistently surpasses all individual models, achieving 0.868 AUC, 76.9% accuracy, and an F1-score of 0.834 for directional prediction, representing a substantial gain over standalone text-based, tabular, or temporal models. For magnitude forecasting, the ensemble reduces RMSE from 0.829 to 0.718. To our knowledge, this is the first large-scale study to fuse Arabic financial text, structured market data, and temporal patterns into a stacked ensemble model for event-driven forecasting. The results highlight how combining language signals with market indicators can improve prediction quality in Arabic-language markets, where reliable automated tools remain limited.

Keywords—Arabic corporate announcements; ensemble learning; financial forecasting; stock movement prediction; temporal transformer; text embeddings

I. INTRODUCTION

Forecasting short-term stock movements from unstructured textual announcements remains a challenging problem at the intersection of finance and natural language processing. Corporate announcements often contain market-moving information, such as earnings updates, mergers, and leadership changes, that can shape investor expectations and influence price dynamics [1], [2]. In markets where announcements are published in low-resourced languages such as Arabic, the need for predictive frameworks capable of combining multilingual text understanding with quantitative market signals becomes even more pronounced. Recent advances in deep learning have shown

that Transformer-based architectures [3] effectively capture semantic content in text, while ensemble techniques such as gradient boosting [4] remain highly competitive for structured financial features. Yet, each modality on its own is incomplete: text-only models ignore numerical context, whereas purely quantitative models cannot capture the linguistic signals embedded in corporate communication.

Corporate announcements are known to drive price discovery both immediately and over subsequent trading days [5], [6], making next-day movement prediction particularly relevant for event-driven strategies [7] and risk management applications [8]. However, Arabic financial markets remain underexplored, and several challenges complicate this predictive task. Announcements vary widely in tone and granularity; small price changes introduce label ambiguity near zero returns; and market responses are shaped by momentum, liquidity, and calendar effects. The non-stationary nature of financial data further complicates model stability. In this setting, probability calibration becomes essential for threshold selection and confident decision-making. An effective forecasting system must therefore integrate semantic interpretation, structured features, temporal dependencies, and calibrated confidence estimates.

Prior research in text-based financial prediction has progressed from sentiment analysis to event-driven and deep sequence models, but most work focuses on English-language corpora, leaving low-resource settings largely unaddressed. Tabular models such as boosted trees remain strong baselines for numerical features [9], [10], yet they cannot represent unstructured semantics. Sequence models capture temporal patterns but often underperform when data is limited. Methodological issues further hinder prior work, including: 1) treating near-zero returns as meaningful signals; 2) relying on uncalibrated probabilities; 3) performing late fusion without resolving conflicts between text-driven and momentum-driven signals; and 4) using random splits that risk look-ahead bias.

To address these gaps, this study introduces a hybrid multimodal architecture that combines multilingual text embeddings with tabular and temporal representations through complementary expert models and a stacked meta-learner. The framework incorporates calibrated probability estimation, gray-zone weighting for ambiguous near-zero returns, and time-aware evaluation. Its main contributions are:

1) *Joint two-task temporal model*: A Transformer-based temporal encoder predicts return magnitude and direction simultaneously using recent sequences of text-numeric features,

*Corresponding author.

with imbalance controlled via positive-class weighting.

2) *Gray-zone weighting*: A return threshold (τ) attenuates the influence of ambiguous near-zero movements, reducing classification noise.

3) *Stacked fusion*: A meta-learner integrates predictions from boosted trees, the temporal model, and a text-only classifier.

4) *Calibration and thresholding*: Probabilities are calibrated using isotonic regression, and thresholds are optimized via validation sweeps for metrics such as F1.

5) *Time-aware evaluation*: Chronological splitting ensures realism and prevents look-ahead bias.

Applied to more than 46,000 Arabic corporate announcements paired with market features, the hybrid ensemble improves directional Area Under the Receiver Operating Characteristic Curve (AUC) from 0.654 (tabular baseline) to 0.868, and accuracy from 0.605 to 0.769. For return magnitude, Root Mean Squared Error (RMSE) decreases from 0.829 to 0.718, with further gains from the sequence model and stacking. These results highlight the complementary value of combining textual, tabular, and temporal signals.

The remainder of the paper is organized as follows. Section II reviews prior research. Section III introduces the hybrid architecture. Section IV describes the training and calibration procedure. Section V details empirical findings. Section VI provides discussion, and Section VII closes the paper.

II. RELATED WORK

Early approaches to stock prediction from news and textual data focused on sentiment analysis and event detection [11], [12]. Researchers have shown that news sentiment and keyword-based indicators can influence stock movements, motivating the use of text mining for financial prediction [7]. Traditional machine learning models (e.g., support vector machines, logistic regression) have been used to classify news or tweets as bullish/bearish signals, often relying on manual feature engineering for text [13], [14]. With the advent of deep learning, more sophisticated architectures have been applied. Ding et al. [7] proposed an event-driven stock prediction model that first extracts structured events from news text and then uses a deep neural network to predict market reaction. Other works have employed recurrent neural networks (RNNs) or LSTMs to incorporate sequences of news or reports over time, capturing temporal dependencies in how information impacts prices [15], [16]. More recently, Transformers [3] and pre-trained language models like BERT [17] have been leveraged for financial text understanding, yielding improved performance in tasks such as stock movement classification from news headlines and earnings calls [18].

On the quantitative side, statistical time-series models (ARIMA, GARCH) [19] and machine learning models (random forests, gradient boosting) [20] have been extensively used for price forecasting based on historical prices and technical indicators. Gradient-boosted tree models, such as XGBoost [4], have gained popularity due to their strong performance on tabular financial data and ease of handling heterogeneous features [21]. These models have been successful in capturing

non-linear relationships in financial indicators and have won numerous forecasting competitions [10], [22].

Our work situates itself at the intersection of these two lines of research by constructing a hybrid model that combines textual and numerical inputs. Prior studies have noted that combining multiple sources of information can improve prediction robustness. For example, recent ensemble approaches use stacking or hybrid architectures to integrate signals from technical models and text models [23]. However, deploying such systems for Arabic markets remains under-explored. To the best of our knowledge, this is one of the first studies to incorporate Arabic textual corporate announcements into a predictive model using modern Transformer embeddings, and to blend this with traditional quantitative features in an ensemble. We draw on standard methods in each domain (e.g., multilingual BERT for text, XGBoost for tabular data, stacking for ensembling) and extend them with a novel training scheme tailored for joint classification-regression forecasting.

III. HYBRID MODEL ARCHITECTURE

Fig. 1 provides an overview of our hybrid forecasting architecture, which consists of three main components: a text embedding stream, a tabular feature stream, and a prediction ensemble that combines multiple learned models. In this section, we describe each component in detail and present the mathematical formulation of the models and training objectives.

A. Text Embedding Layer

Formally, the Transformer encoder implements a function $f_T : \mathcal{X}_T \rightarrow \mathbb{R}^d$ where $\mathcal{X}_T$ is the space of text sequences. It is composed of self-attention layers that enable context-aware encoding of each token in the announcement [3]. We use the pretrained BAAIbge-m3 multilingual embedding model as the Transformer encoder. We do not modify the pre-trained weights extensively; rather, we fine-tune the encoder on our dataset of announcements (with task-specific supervision) so that the resulting embeddings are optimized for the downstream prediction task. Fine-tuning ensures that e_i emphasizes aspects of the text that correlate with stock jumps or drops (e.g., optimistic language about earnings, or warnings of losses). The announcements are tokenized with a maximum sequence length of 512, and the CLS token representation from the final hidden layer is used as the text embedding.

We denote an announcement i at date t with cleaned text T_i and tabular feature vector $x_i \in \mathbb{R}^p$ summarizing recent returns, volatilities, and aggregates. A multilingual encoder maps text to $e_i \in \mathbb{R}^d$. The classification label is $y_i^{(c)} \in \{0, 1\}$ (up/down next day), and the regression target is $r_i \in \mathbb{R}$ (next-day fractional return).

B. Tabular Feature Stream

In parallel with text processing, our system ingests a set of structured numerical features for each instance. These features capture recent market dynamics and company-specific context. Specifically, for each announcement i associated with a trading day t , we construct a feature vector x_i that includes:

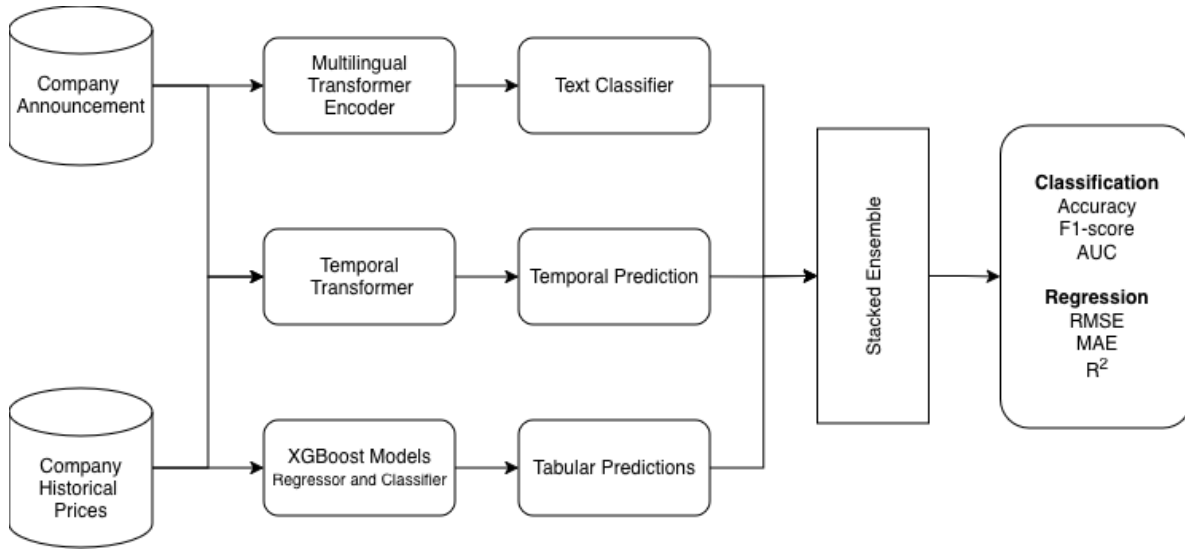


Fig. 1. Hybrid architecture of the proposed forecasting system. The text of each announcement is encoded by a Transformer encoder into a dense vector, while recent tabular features (returns, volatility, and related quantities) are processed by dedicated models (XGBoost and a temporal Transformer). Predictions from these components are combined by a stacked ensemble to produce final forecasts of return magnitude and direction.

1) *Prior returns*: e.g., the stock's daily returns over the past k days (a short history leading up to the announcement).

2) *Rolling volatility*: statistical measures such as the standard deviation of returns over a recent window, to gauge market uncertainty or momentum.

3) *Company financials*: relevant fundamental indicators (if available) such as market capitalization, recent revenue growth, or debt ratios.

4) *Market aggregates*: broader market indices or sector indices returns, to situate the company's movement in the market context.

All features are computed from publicly available data prior to or on the day of the announcement, so that the prediction is genuinely forward-looking (predicting the next-day outcome). We apply standard preprocessing (normalization, winsorization of outliers) to $\mathbf{x}_i$ and optionally augment it with interaction terms based on domain knowledge (for example, a momentum indicator interacting with a volatility index) (see Algorithm 1).

Algorithm 1 Time-Aware Feature Construction and Split

Require: Records $\{(T_i, \mathbf{x}_i, r_i, y_i^{(c)}, t_i)\}_{i=1}^N$, window L , valid ratio ρ

- 1: Compute text embeddings $\mathbf{e}_i \leftarrow f_T(T_i)$
- 2: Build per-day descriptors $\mathbf{q}_t$ from recent returns and volatilities
- 3: For each firm and date, form sequence $\mathbf{z}_{t-L+1:t} \leftarrow [\mathbf{e}_{t'}, \mathbf{q}_{t'}]_{t'=t-L+1}^t$
- 4: Sort by time; choose cut date t^* so that $\lfloor (1-\rho)N \rfloor$ samples lie before t^*
- 5: **return** training set $\{i : t_i \leq t^*\}$, validation set $\{i : t_i > t^*\}$

The tabular feature vector $\mathbf{x}_i$ serves as input to two of our predictive models: the XGBoost models and the temporal Transformer model (described next). It has dimensionality p , with p typically on the order of tens to a few hundreds of features. Importantly, some components of $\mathbf{x}_i$ are lagged time-series values (like past returns), which means there is inherent

temporal structure that the temporal model can exploit beyond what a static model like XGBoost might capture.

C. Gradient-Boosted Tree Models

We train gradient-boosted decision tree models [4] on the tabular features for both regression and classification tasks. XGBoost is a powerful ensemble-of-trees method that iteratively fits new decision trees to the residuals of the current model, minimizing a specified loss. It has proven effective for structured financial data due to its ability to handle non-linear feature interactions and varying scales.

For regression, the XGBoost regressor $f_{\text{xgb}}^{(\text{reg})}(\mathbf{x})$ is trained to predict the next-day stock return r_i (we define r_i as the fractional price change from the previous close to the next day's close for announcement i). The model optimizes a regularized objective:

$$L_{\text{xgb-reg}} = \sum_i (r_i - \hat{r}_i)^2 + \sum_m \Omega(T_m), \quad (1)$$

where, $\hat{r}_i = f_{\text{xgb}}^{(\text{reg})}(\mathbf{x}_i)$ is the predicted return, and $\Omega(T_m)$ is a complexity penalty on each tree T_m in the ensemble (to control overfitting) as defined in [4]. The loss is the mean squared error (MSE) augmented with regularization terms for tree depth, leaf weights, and related quantities. The output of this model is a continuous prediction $\hat{r}_i$.

For classification, we train a separate XGBoost classifier $f_{\text{xgb}}^{(\text{clf})}(\mathbf{x})$ to predict the direction label $y_i^{(c)}$, which is 1 if the stock price goes up the next day and 0 if it goes down. This model optimizes the logistic loss:

$$L_{\text{xgb-clf}} = - \sum_i \left[y_i^{(c)} \log \hat{p}_i + (1 - y_i^{(c)}) \log(1 - \hat{p}_i) \right] + \sum_m \Omega(T'_m), \quad (2)$$

where, $\hat{p}_i = f_{\text{xgb}}^{(\text{clf})}(x_i)$ is the predicted probability of an upward move and T'_m are the trees in the classifier model. We apply a sigmoid transformation to the raw ensemble score to obtain $\hat{p}_i \in [0, 1]$. Class imbalance is handled by a scale weight ω applied to positive examples in the loss if needed. In our dataset, the fraction of positive (upward) moves is roughly 55%, while negative (downward) moves account for approximately 45%, so we set a weight ω slightly above 1 for negative examples to balance the classes (the exact value is tuned; in our case, $\omega \approx 1.2$, corresponding to `pos_weight` ≈ 0.83 for positives).

Notably, we incorporate a **gray-zone weighting** scheme into the classifier training to account for days with very small price changes. We define a threshold τ for negligible movements (the “gray zone”). If the absolute return $|r_i| < \tau$, the training example is considered ambiguous noise (the price essentially stayed flat). We down-weight these examples in the classification loss to reduce their influence. Formally, each instance i is given a weight $w_i^{(c)}$ in $L_{\text{xgb-clf}}$ such that:

$$w_i^{(c)} = \begin{cases} \lambda, & |r_i| < \tau \\ 1, & |r_i| \geq \tau \end{cases} \quad \text{with } 0 \leq \lambda < 1. \quad (3)$$

In our experiments, we set $\lambda = 0.5$ and $\tau = 0.05$ (i.e., a 5% relative price change) to de-emphasize borderline cases. The gray-zone weighting does not introduce a separate no-movement class; these observations retain their original binary labels. This heuristic encourages the model to focus on instances where the stock had a meaningful move, thus learning more distinct signals for clear upward versus downward movements. We will show in Section V that this leads to improved precision and recall by avoiding random fluctuations. (See Algorithm 2.)

Algorithm 2 Tabular Boosted Trees with Gray-Zone Weighting (Classification) and Regression

Require: Train indices $\mathcal{I}_{\text{tr}}$, features x_i , targets $(r_i, y_i^{(c)})$, class weight `pos_w`, gray-zone (τ, λ)

- 1: Fit regressor $f_{\text{tab}}^{(\text{reg})}$ on $\{(x_i, r_i)\}_{i \in \mathcal{I}_{\text{tr}}}$ by minimizing squared loss with tree regularization
- 2: Compute $w_i^{(c)} \leftarrow \lambda$ if $|r_i| < \tau$ else 1 for $i \in \mathcal{I}_{\text{tr}}$
- 3: Fit classifier $f_{\text{tab}}^{(\text{clf})}$ on $\{(x_i, y_i^{(c)}, w_i^{(c)}, \text{pos_w})\}_{i \in \mathcal{I}_{\text{tr}}}$ by minimizing weighted logistic loss with tree regularization
- 4: **return** $f_{\text{tab}}^{(\text{reg})}, f_{\text{tab}}^{(\text{clf})}$

D. Transformer-Based Temporal Model

While XGBoost treats each instance independently (using summary features of recent history), we also employ a Transformer-based *temporal* model that can directly ingest a sequence of recent daily data to capture temporal patterns. The motivation is that certain patterns in how news and stock prices evolve over time (e.g., a series of positive announcements, or a divergence from market trends) might be better recognized by a sequence model with long-range dependency capture.

Our temporal model is implemented as a Transformer encoder that operates on a sequence of L days leading up to and including the announcement day. For each day $t - L + 1$

through t (where day t has the announcement T_i for company i), we create a combined embedding of that day’s information:

$$z_{t'} = \left[e_{t'}^{(\text{text})} \parallel q_{t'}^{(\text{num})} \right], \quad t' = t - L + 1, \dots, t \quad (4)$$

where, $e_{t'}^{(\text{text})}$ is the text embedding of any announcement on day t' (if multiple announcements occur on that day for the company, they could be concatenated or the most relevant chosen; if no announcement occurs on a given past day, we use a zero or placeholder vector), and $q_{t'}^{(\text{num})}$ is a vector of numerical features for that day (such as the daily return, volatility, and related quantities). The two parts are concatenated into $z_{t'}$, giving a representation for day t' that includes both that day’s text (if any) and quantitative state. These $z_{t'}$ are then fed as a sequence into a Transformer encoder (with positional encoding to indicate the ordering in time).

Let $\{z_{t-L+1}, \dots, z_t\}$ denote the input sequence to the temporal model for announcement i on day t . The Transformer processes this sequence with self-attention, effectively allowing the model to attend to relevant days in the past when making predictions for day $t+1$. We take the Transformer’s output corresponding to the last day t (or a special classification token that we prepend to the sequence) as the basis for prediction. We denote the temporal Transformer model as f_{temp} , which outputs two values:

$$\left(\hat{r}_i^{(\text{temp})}, \hat{p}_i^{(\text{temp})} \right) = f_{\text{temp}} \left(\{z_{t'}\}_{t'=t-L+1}^t \right) \quad (5)$$

where, $\hat{r}_i^{(\text{temp})}$ is the predicted return magnitude (regression output) and $\hat{p}_i^{(\text{temp})}$ is the predicted probability of an upward move (classification output) for the next day $t+1$. Thus, the temporal model is a multi-task model that addresses both regression and classification jointly.

We optimize it using a combined loss:

$$L_{\text{temp}} = \frac{1}{N} \sum_{i=1}^N \left[(r_i - \hat{r}_i^{(\text{temp})})^2 + \alpha \cdot \ell_{\text{BCE}}(y_i^{(c)}, \hat{p}_i^{(\text{temp})}) \right], \quad (6)$$

where, ℓ_{BCE} is the binary cross-entropy loss for the classification output and α is a weighting hyperparameter to balance the regression and classification tasks. In practice, we set α such that the losses are on comparable scales (e.g., $\alpha \approx 1$ after normalizing r_i to a similar range as the probability scale). The temporal model is trained on the same training set as the other models, with supervision for both tasks. By learning both tasks together, the model can develop internal representations that are useful for understanding how textual events affect not only the direction but also the magnitude of price changes. We found that multi-task training acts as a regularizer: it prevents the Transformer from overfitting to one aspect (say, the exact regression target) by also requiring it to predict the directional class (Algorithm 3).

Algorithm 3 Temporal Multi-Task Training with Class Weight and Gray-Zone Weighting

Require: Train indices $\mathcal{I}_{tr}$, sequences $\mathbf{z}_{t-L+1:t}$, targets $(r_i, y_i^{(c)})$, weights $(\alpha, \text{pos_w}, \tau, \lambda)$

- 1: Initialize temporal encoder f_{temp}
- 2: **for** epochs $= 1, \dots, E$ **do**
- 3: **for** mini-batch $\mathcal{B} \subset \mathcal{I}_{tr}$ **do**
- 4: $(\hat{r}_i^{(\text{temp})}, \hat{p}_i^{(\text{temp})}) \leftarrow f_{\text{temp}}(\mathbf{z}_{t_i-L+1:t_i})$ for $i \in \mathcal{B}$
- 5: $w_i^{(c)} \leftarrow \lambda$ if $|r_i| < \tau$ else 1 (gray-zone)
- 6: $\ell_{\text{reg}} \leftarrow \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} (r_i - \hat{r}_i^{(\text{temp})})^2$
- 7: $\ell_{\text{clf}} \leftarrow \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} w_i^{(c)} \cdot \text{BCE}(y_i^{(c)}, \hat{p}_i^{(\text{temp})}; \text{pos_w})$
- 8: Update f_{temp} to minimize $\ell_{\text{reg}} + \alpha \ell_{\text{clf}}$
- 9: **end for**
- 10: **end for**
- 11: **return** trained f_{temp}

The architecture of the temporal Transformer uses H self-attention heads and D layers (we used $H = 8$ and $D = 4$ in our experiments), allowing it to capture complex temporal patterns. This model in effect serves as a learned time-series feature extractor that complements the static XGBoost approach. It is particularly useful for capturing trends or reversals: for example, if a company's stock has been trending up but an announcement is negative, the contrast can be learned by the Transformer attending to prior upward momentum and the current negative language, adjusting the prediction accordingly.

E. Stacked Ensemble

Finally, we integrate the outputs of the various models through a stacked ensemble. Stacking (stacked generalization) [24] is an ensemble technique where a meta-learner is trained to combine the predictions of base learners, usually using a hold-out set or cross-validation to avoid overfitting. In our system, the base learners are:

- XGBoost regressor ($\hat{r}_i^{(\text{xgb})}$) and classifier ($\hat{p}_i^{(\text{xgb})}$),
- Temporal Transformer regressor ($\hat{r}_i^{(\text{temp})}$) and classifier ($\hat{p}_i^{(\text{temp})}$),
- A text-only classifier ($\hat{p}_i^{(\text{text})}$) that uses the Transformer text embedding e_i directly to predict the movement direction (this can be a simple logistic regression or feed-forward network trained on the text embedding to output a probability).

The inclusion of a text-only classifier in the ensemble is to ensure that pure textual signals (which might be attenuated or differ from the patterns picked up by the temporal model) are not lost. For instance, the language in an announcement might strongly indicate a direction even if recent numeric trends suggest otherwise; the text-only model can capture this specialized signal.

The outputs from these models for each instance i form a set of features for the meta-learner. For classification, we take as meta-features the predicted probabilities (and optionally some transformations like log-odds) from XGBoost, the temporal model, and the text model: $\{\hat{p}_i^{(\text{xgb})}, \hat{p}_i^{(\text{temp})}, \hat{p}_i^{(\text{text})}\}$. For regression, we take the predicted returns: $\{\hat{r}_i^{(\text{xgb})}, \hat{r}_i^{(\text{temp})}\}$

(the text-only model was used only for classification in our case). We then train a simple meta-learner on top of these. For classification, a logistic regression or shallow neural network works well as the meta-learner, as it can learn weights to optimize the final classification decision. For regression, a linear regression on the predictions or another XGBoost regressor can be used as the meta-learner.

The meta-learner training is done using a two-stage approach to avoid overfitting: we perform K -fold cross-validation on the training data to get out-of-fold predictions from each base model, then train the meta-learner on those predictions. Alternatively, we set aside a portion of the training data as a validation set to train the stacker. In our experiments, we used a hold-out validation set for stacking due to data volume constraints. The meta-learner learns, for example, that the text model's prediction is very valuable in certain situations (e.g., when the text contains clearly positive or negative content), whereas in other cases the numeric models might be more reliable.

At prediction time, the stacked ensemble produces the final outputs: $\hat{p}_i^{(\text{ens})}$ for the probability of an upward movement and $\hat{r}_i^{(\text{ens})}$ for the predicted return. These are the outputs we use for decision-making and evaluation. The ensemble effectively leverages the strengths of each component: XGBoost provides a strong baseline using well-understood features (capturing, e.g., general market uptrends or mean-reversion effects), the temporal Transformer contributes pattern recognition over time and joint text-numeric interactions, and the text-only model injects pure NLP-driven signals. This combination substantially improves performance metrics as we show in the results.

IV. TRAINING, CALIBRATION, AND THRESHOLDING

In this section, we outline our model training process and the additional calibration steps taken to ensure reliable probability outputs. We also discuss the selection of a decision threshold for classification optimized for our target metric (Algorithm 4).

Algorithm 4 Stacking, Calibration, and Threshold Selection

Require: Validation indices $\mathcal{I}_{va}$; base predictors $f_{\text{tab}}^{(\text{reg})}$, $f_{\text{tab}}^{(\text{clf})}$, $f_{\text{text}}^{(\text{clf})}$

- 1: **for** $i \in \mathcal{I}_{va}$ **do**
- 2: $\hat{p}_i^{(\text{tab})} \leftarrow f_{\text{tab}}^{(\text{clf})}(\mathbf{x}_i)$
- 3: $\hat{p}_i^{(\text{temp})} \leftarrow f_{\text{temp}}(\mathbf{z}_{t_i-L+1:t_i})$
- 4: $\hat{p}_i^{(\text{text})} \leftarrow f_{\text{text}}^{(\text{clf})}(e_i)$
- 5: $\hat{r}_i^{(\text{tab})} \leftarrow f_{\text{tab}}^{(\text{reg})}(\mathbf{x}_i)$
- 6: $\hat{r}_i^{(\text{temp})} \leftarrow$ temporal regression output
- 7: **end for**
- 8: Fit meta-classifier on $\{(\hat{p}_i^{(\text{tab})}, \hat{p}_i^{(\text{temp})}, \hat{p}_i^{(\text{text})}), y_i^{(c)}\}_{i \in \mathcal{I}_{va}}$ to produce $\hat{p}_i^{(\text{ens})}$
- 9: Fit isotonic calibrator g mapping $\hat{p}_i^{(\text{ens})} \mapsto \tilde{p}_i^{(\text{ens})}$ using $\mathcal{I}_{va}$
- 10: Select threshold $\theta^* \in [0, 1]$ by maximizing a chosen metric (e.g., F1) on $\{(\tilde{p}_i^{(\text{ens})}, y_i^{(c)})\}_{i \in \mathcal{I}_{va}}$
- 11: Fit meta-regressor on $\{(\hat{r}_i^{(\text{tab})}, \hat{r}_i^{(\text{temp})}), r_i\}_{i \in \mathcal{I}_{va}}$ to produce $\hat{r}_i^{(\text{ens})}$
- 12: **return** calibrated classifier (g, θ^*) and meta-regressor

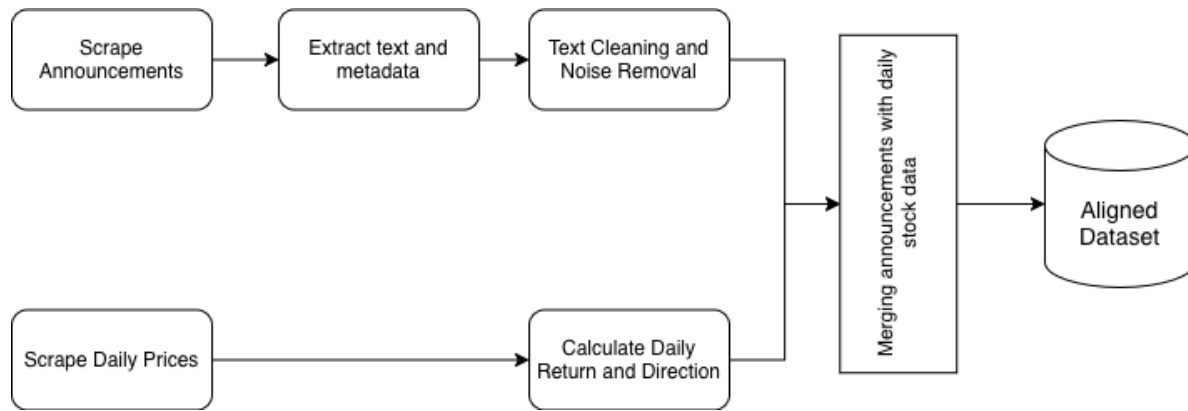


Fig. 2. Data collection and preparation workflow. Corporate announcements are scraped, cleaned, and paired with their corresponding metadata, while daily stock prices are processed to compute returns and directional labels. Both sources are merged chronologically to form the aligned dataset used for model training.

A. Dataset

The Saudi Stock Exchange (Tadawul) serves as the primary data source for this research. As the only securities exchange in Saudi Arabia and one of the largest in the Middle East, Tadawul operates under the supervision of the Capital Market Authority (CMA) and is publicly listed, owned by the Public Investment Fund (PIF). It provides a regulated environment for stock trading and transparent access to corporate announcements, official reports, and financial disclosures. From this platform, two main datasets were collected: daily stock prices and official company announcements.

Fig. 2 illustrates the overall data collection workflow. Announcement data were gathered directly from the Tadawul portal using a three-phase web-scraping pipeline. The first phase captured announcements transmitted in JSON format, while the second extracted key metadata such as the announcement date, time, and company name. The final phase involved text cleaning and transformation, including renaming fields for clarity, removing irrelevant attributes, filtering out incomplete records, and preprocessing the announcement body using standard NLP techniques. HTML and JavaScript tags, special characters, and other noise were removed, and URLs or email addresses were masked with placeholders to produce a consistent Arabic textual dataset suitable for modeling. This process yielded 46,670 cleaned announcements. In parallel, daily stock price data—consisting of open, close, high, and low values—were retrieved and used to compute daily returns and price direction.

For labeling, the cleaned announcements were merged with corresponding stock price data based on the publication date and time. The prediction target was defined as the stock price movement in the next applicable trading session following the announcement. The percentage change in closing price was categorized into two labels: 0 for a decrease and 1 for an increase. This labeling approach ensured a consistent time-aware mapping between corporate announcements and subsequent market reactions, forming the foundation for predictive modeling.

B. Model Training Procedure

All models were trained on a common dataset of stock announcements. The data were chronologically divided into training, validation, and testing sets using a 70:15:15 ratio. Of the 46,700 announcements, 32,700 were used for training, 7,000 for validation, and 7,000 for testing. The chronological order was maintained, with the earliest observations used for training, followed by validation, while the most recent observations were reserved for testing. This approach prevents look-ahead bias and allows the models to be evaluated on unseen future data. For each company, input features were constructed in chronological order using only information from previous announcements and available historical data. The XGBoost models were trained using the XGBoost library with early stopping on the validation set to prevent overfitting (using validation loss and AUC as criteria). The hyperparameters (number of trees, maximum depth, learning rate, L2 regularization, and related settings) were tuned via grid search on the validation set.

The Transformer-based models (text encoder fine-tuning and temporal model) were trained using PyTorch. We initialized the text encoder with pretrained weights (a multilingual BERT model [17]) and fine-tuned it together with a small classification head (for the text-only classifier) on the training data for a few epochs, using binary cross-entropy loss for direction prediction. We used a relatively low learning rate for the encoder (2×10^{-5}) to avoid catastrophic forgetting of language features. The temporal model was trained from scratch (aside from the text embeddings, which were fed from the frozen or semi-frozen encoder) using the multi-task loss L_{temp} . We applied dropout and early stopping (monitoring classification AUC on validation) to regularize this model, as it has a large capacity relative to our data size. The `pos_weight` for the classification loss in the temporal model was set to the same value as in XGBoost to handle the slight class imbalance.

During training, we employed the gray-zone weighting described earlier for both the XGBoost classifier and the temporal model's classification loss. Empirically, we found that setting $\tau = 0.05$ (5%) to define negligible moves helped; approximately 10% of training instances fell in this gray zone

and had their loss weight scaled down. This led to a small but consistent improvement in F1-score, as the models focused more on getting the direction correct for larger moves (which are of greater practical significance).

The temporal Transformer was configured with a sequence length of $L = 60$, an embedding dimension of 192, four Transformer layers, and six attention heads, with a dropout rate of 0.1. The model was trained using the AdamW optimizer with a learning rate of 0.001, a batch size of 256, and a maximum of 20 epochs.

After obtaining trained base models, we generated predictions on the validation set to train the ensemble meta-learner. For each instance in the validation set, we collected $\hat{p}^{(\text{xgb})}$, $\hat{p}^{(\text{temp})}$, $\hat{p}^{(\text{text})}$, $\hat{r}^{(\text{xgb})}$, and $\hat{r}^{(\text{temp})}$. We then trained the logistic regression meta-classifier and linear regression meta-regressor on this validation data. This stacking approach was chosen to ensure the meta-learner sees a realistic performance of base models (since they were not trained on the validation set, their behavior on validation simulates test-time behavior). Finally, we evaluated the entire pipeline on the held-out test set, which the models never saw during training or stacking.

C. Probability Calibration

While our models output probabilities (especially the ensemble classifier, which produces $\hat{p}_i^{(\text{ens})}$ for each instance), these probabilities are not guaranteed to be calibrated; that is, the predicted probability may not reflect the true empirical frequency of an upward move. Calibration is important in financial contexts if we want to interpret the output as confidence or feed it into a decision strategy (for example, to decide not to trade if the probability of a correct prediction is below a threshold). We therefore applied an isotonic regression calibration [25] on the validation set outputs of the ensemble classifier.

Isotonic regression is a non-parametric calibration method that fits a monotonic piecewise-constant function to map raw scores to calibrated probabilities. Using the validation data, we obtained the ensemble's uncalibrated probabilities and the true outcomes, and then learned an isotonic regression function $g(\cdot)$ such that $g(\hat{p})$ best matches the true fraction of positive outcomes for instances with score $\hat{p}$. This function was then applied to all ensemble outputs. Because isotonic regression is monotonic, it does not scramble the ranking of predictions; it only adjusts the confidence level. Post-calibration, if the model reports a probability of 0.7, ideally about 70% of those predictions should be correct.

We found that the ensemble's raw scores were slightly over-confident prior to calibration (for instance, it might assign 95% probability to some points, but only approximately 90% of those were actually positive). After isotonic calibration, the probabilities became well-aligned with observed frequencies. Notably, calibration does not change the AUC metric (which is threshold-invariant), but it can improve the interpretability of probabilities and the choice of threshold for a desired precision-recall trade-off.

D. Threshold Tuning

The final step in our pipeline was selecting a decision threshold for classifying an instance as upward movement

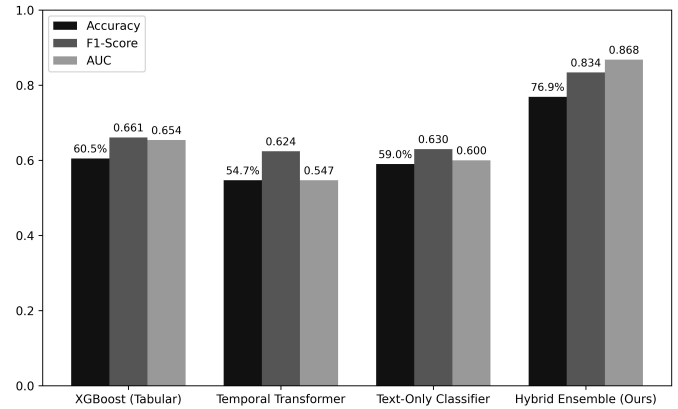


Fig. 3. Classification performance across base models and the hybrid ensemble. Metrics shown include accuracy, F1-score, and AUC. The hybrid ensemble outperforms the tabular-only, temporal, and text-only models across all evaluation measures.

(positive) or downward (negative). The default threshold is 0.5 for a balanced binary classification, but depending on our objective (maximizing F1-score, for example), a different threshold might yield better results. On the validation set, we swept the classification threshold from 0 to 1 and evaluated the F1-score and accuracy of the ensemble. The best F1 was achieved at a threshold of approximately 0.4, significantly lower than 0.5. This indicates that the classifier needed to be more sensitive in predicting the positive class (up movement) to balance precision and recall.

We fixed the threshold at $\theta^* = 0.4$ for the test evaluation, as this θ^* maximized F1 on validation. This threshold, combined with the calibrated probabilities, effectively creates a decision rule: predict up if $\hat{p}_i^{(\text{ens})} > 0.4$, else predict down. In Section V, we examine the impact of this choice. We also report performance at $\theta = 0.5$ for reference, but all primary metrics are given for the tuned threshold. The use of a tuned threshold improved the classification F1 from about 0.78 (at the 0.5 threshold) to 0.834 at the chosen threshold, with a slight increase in accuracy as well.

V. RESULTS

We evaluate our hybrid system on a test set of Arabic corporate announcements, comparing the performance of individual model components and the final ensemble. Key evaluation metrics for the classification task are accuracy, F1-score (for the positive “up” class), and AUC (area under the ROC curve), as shown in Fig. 3. For the regression task, we report the root mean squared error (RMSE), mean absolute error (MAE), and R^2 (coefficient of determination) of the predicted returns, as shown in Fig. 4.

Table I summarizes the classification performance of various models. The hybrid ensemble achieves the highest performance across all metrics. In particular, it attains an accuracy of 76.9%, an F1-score of 0.834, and an AUC of 0.868, substantially outperforming the standalone models. By contrast, the XGBoost classifier (using only tabular features) reaches about 60.5% accuracy and 0.654 AUC. The Transformer temporal model by itself did not perform as well in classification (around

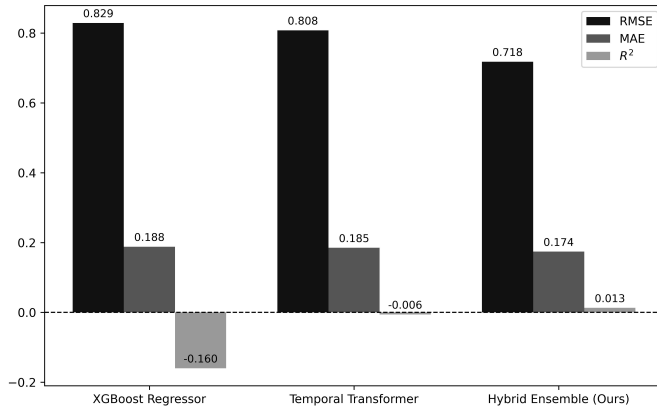


Fig. 4. Regression performance across base models and the hybrid ensemble. Metrics reported include RMSE, MAE, and R^2 . The hybrid ensemble achieves the lowest RMSE and MAE and the highest R^2 , indicating improved accuracy and explanatory power over both individual models.

TABLE I. DIRECTIONAL PREDICTION PERFORMANCE OF DIFFERENT MODELS

Model	Accuracy (%)	F1-score	AUC
XGBoost (tabular)	60.5	0.661	0.654
Temporal Transformer	54.7	0.624	0.547
Text-only classifier	59	0.63	0.60
Hybrid ensemble (ours)	76.9	0.834	0.868

a. All metrics are computed on the held-out test set.

54.7% accuracy, AUC 0.547), possibly due to the difficulty of learning from limited data and the multi-task training trade-off. Interestingly, a simple text-only classifier achieved intermediate performance (we observed around 0.60–0.65 AUC in validation), indicating that text alone has predictive power but not enough to surpass the tabular model. The ensemble, however, combines these signals effectively: its AUC of 0.868 represents an increase of over 20 percentage points versus the temporal model and over 21 points versus the XGBoost baseline. This highlights the complementarity of text and numeric features in this domain. The F1-score improvement is similarly large: from 0.661 for XGBoost to 0.834 for the ensemble. The high F1 indicates that the ensemble achieves a good balance of precision and recall for the “up” class after threshold tuning. Indeed, with $\theta^* = 0.4$, the ensemble’s precision and recall for upward moves were both in the low 80s percentage-wise, whereas at a 0.5 threshold the recall was lower, underscoring the importance of threshold optimization.

The superior performance of the ensemble can be attributed to how it leverages each component: the XGBoost model contributes a reliable baseline of structured signals (achieving decent precision on more predictable scenarios), the text model contributes information on events that deviate from historical trends (e.g., surprise announcements), and the temporal model captures sequential patterns (though its standalone performance was lower, it likely provides useful features to the stacker that are not available from XGBoost alone). We also note that the ensemble’s accuracy of 76.9% is quite high for a stock movement prediction task, reflecting that the task is somewhat easier in this setting than random guessing (which

TABLE II. REGRESSION PERFORMANCE FOR PREDICTING NEXT-DAY RETURN MAGNITUDE

Model	RMSE	MAE	R^2
XGBoost regressor	0.829	0.188	−0.160
Temporal Transformer	0.808	0.185	−0.006
Hybrid ensemble (ours)	0.718	0.174	+0.13

Error metrics are expressed in fractional units, where 1.0 corresponds to a 100% price change.

would be approximately 50%). This may be due to the nature of corporate announcements, which often indeed cause a clear directional move. The AUC of 0.868 suggests the ranking of positive versus negative instances by the model is very good, which is important for decision-making.

Next, Table II presents the regression performance. We compare the RMSE and MAE of predicting the actual next-day return for the XGBoost regressor, the temporal model’s regression output, and the ensemble regressor. The XGBoost regression had an RMSE of 0.829 (which is quite high relative to typical daily returns, indicating difficulty in pinpointing exact magnitudes) and an R^2 that was slightly negative ($R^2 = -0.16$), meaning it underperformed a trivial mean-prediction baseline. The temporal model’s regression did better, with RMSE 0.808 and $R^2 \approx -0.006$ (essentially zero). This suggests that incorporating recent sequence information and text allowed it to match the baseline and slightly outperform XGBoost in RMSE. Finally, the ensemble regressor achieved RMSE = 0.718. This is a meaningful improvement (over 13% reduction in RMSE compared to XGBoost). While the R^2 for the ensemble is still modest ($R^2 = 0.13$), it indicates that the ensemble can explain some variability in returns that single models cannot. In practical terms, an RMSE of 0.718 on next-day returns implies that, although precise magnitudes are hard to get perfectly, the model is capturing enough signal to reduce error significantly relative to naively assuming no change.

It is worth noting that predicting exact return percentages is inherently noisy and challenging, especially with unexpected market factors. The negative R^2 values for the base models reflect that challenge—a constant prediction (e.g., always predict 0% change) could sometimes beat those models in MSE. However, the ensemble bringing R^2 into positive territory is an encouraging sign that combining text and data does add explanatory power. We observed that the biggest errors for the base models occurred on days of extreme moves (large positive or negative jumps), which is expected since those are hard to predict. The ensemble was relatively better at those, likely because the announcements that led to extreme moves contained clear signals (e.g., catastrophic news or extremely good news) which the text model picked up, and thus the ensemble could anticipate a large magnitude better. Nonetheless, even the ensemble had difficulty with a few outliers, which is reflected in the still-high RMSE.

From a calibration perspective, after isotonic calibration, the probabilities output by the ensemble were well-aligned. We plotted the calibration curve (predicted probability versus actual frequency of up moves) on the test set and found it to be nearly diagonal for the ensemble, whereas the raw XGBoost probabilities were flatter (indicating under-confidence for high

probabilities and over-confidence for low probabilities). This calibration helped in selecting the 0.4 threshold. In fact, the optimal threshold $\theta^* = 0.4$ roughly corresponded to an event where the calibrated probability of an upward move was about 50% (since the model was slightly conservative). The net effect is that we treat some cases as “up” even if the raw model probability is slightly below 0.5, because those turned out to be beneficial for maximizing F1. This strategy is reminiscent of moving the decision boundary to compensate for class imbalance or different error costs.

VI. DISCUSSION

The hybrid nature of the system is crucial. If we remove the text channel and use only numeric features, the AUC drops from 0.868 to 0.654 (XGBoost only)—a large degradation. This shows that in our dataset, the textual information in Arabic announcements carries unique predictive signal not captured by past prices or other features. Conversely, using text alone (without numeric context) was also suboptimal, demonstrating that recent stock behavior and market conditions still matter. By fusing the two, the model can, for example, recognize when a positive announcement might already be “priced in” by a prior run-up (a case where numeric features temper the text’s optimism), or when a negative announcement breaks an otherwise positive trend (a case where text overrides momentum). The temporal Transformer was an attempt to have a single model learn these interactions, but its performance lagged, perhaps due to limited data or the difficulty of training a complex sequence model. The stacking approach allowed us to sidestep this by combining simpler models, and it clearly paid off in performance.

Additionally, the gray-zone weighting had an interesting effect: the models became slightly biased towards predicting no movement unless there was a strong signal. We observed that without gray-zone down-weighting, the XGBoost classifier would sometimes attempt to classify every tiny fluctuation (leading to many false positives on essentially flat days). With the weighting, it learned a small “dead band” where it would not confidently predict up or down unless other features (or text sentiment) suggested a real move. In practice, this is useful because it reduces over-trading on days when the stock might not move significantly.

In terms of error analysis, most of the ensemble’s misclassifications involved either very small moves (which we do not mind as much, since they are near the gray zone) or unexpected external events (e.g., an announcement was released but then separate macroeconomic news drove the stock in the opposite direction). There were a few cases where the text of an announcement was misleading (e.g., a very optimistic tone but an actual outcome that worried investors, or vice versa), which points to the limitations of textual sentiment and the need for possibly more complex NLP (such as understanding negation, sarcasm, or reading between the lines of corporate phrasing). However, these cases were relatively rare in our data.

VII. CONCLUSION

We presented a hybrid Transformer-boosted ensemble model for forecasting stock price movements from Arabic corporate announcements. By combining a multilingual Transformer text encoder with gradient-boosted trees on structured

data and a temporal sequence model, our system exploits diverse sources of predictive signals. The stacked ensemble achieved better performance than any individual model, demonstrating the value of integration. We also implemented calibration and a gray-zone weighting strategy to improve the practical decision-making utility of the model. Our results on real-world data show that the model can accurately predict the direction of next-day moves with high precision and recall, and provide reasonable estimates of magnitude.

This work highlights the benefit of modern NLP techniques in financial forecasting, especially for languages and markets where automated text analysis is less developed. It also provides a blueprint for how to architect complex forecasting systems that blend multiple modalities and algorithms. For practitioners, an immediate takeaway is that when dealing with market-moving text like earnings releases, one should not rely solely on either text sentiment or historical price trends; a combination is far more powerful.

VIII. FUTURE WORK

Future work could explore richer text representations (e.g., incorporating domain-specific contextual embeddings or even generative pre-training on financial news in Arabic). The evaluation could also be extended to additional conventional baselines, such as logistic regression, random forest, and SVM, to provide a broader comparison across modeling approaches. Another avenue is to improve the temporal model, possibly using more advanced architectures or training on a larger corpus of sequence data, to further boost sequential pattern recognition. Moreover, expanding the dataset to include announcements from multiple markets or languages would test the generality of the approach. Finally, in a live trading setting, one could integrate this model into a portfolio or risk management framework, where calibrated probabilities help decide not just the direction to trade but also the confidence (and hence position sizing). We believe the interdisciplinary approach demonstrated here—melding NLP, time-series analysis, and ensemble learning—will be increasingly important in developing the next generation of market forecasting tools.

REFERENCES

- [1] A. M. Sarea, S. Subramanian, and B. Alareeni, “Web-based financial disclosures by using machine learning analysis: evidence from bahrain,” A. Hamdan, A. E. Hassanien, A. Razzaque, and B. Alareeni, Eds. Cham: Springer, 2021, pp. 357–371.
- [2] F. Li, “Textual analysis of corporate disclosures: a survey of the literature,” *J Account Lit*, vol. 29, 2011.
- [3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” 2017, arXiv:1706.03762 [Preprint]. [Online]. Available: <https://arxiv.org/abs/1706.03762>
- [4] T. Chen and C. Guestrin, “XGBoost: a scalable tree boosting system,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: Association for Computing Machinery, 2016, pp. 785–794.
- [5] S. Feuerriegel and J. Gordon, “Long-term stock index forecasting based on text mining of regulatory disclosures,” *Decis Support Syst*, vol. 112, pp. 88–97, 2018.
- [6] M. Kraus and S. Feuerriegel, “Decision support from financial disclosures with deep neural networks and transfer learning,” *Decis Support Syst*, vol. 104, pp. 38–48, 2017.

- [7] X. Ding, Y. Zhang, T. Liu, and J. Duan, "Deep learning for event-driven stock prediction," in *Proceedings of the 24th International Joint Conference on Artificial Intelligence*. Buenos Aires, Argentina: AAAI Press, 2015, pp. 2327–2333.
- [8] K. K. V, "Applications of machine learning in predictive analysis and risk management in trading," *Int J Innov Res Comput Sci Technol*, vol. 11, no. 6, pp. 18–25, 2023.
- [9] P. H. Vuong, T. T. Dat, T. K. Mai, P. H. Uyen, and P. T. Bao, "Stock-price forecasting based on XGBoost and LSTM," *Comput Syst Sci Eng*, vol. 40, no. 1, pp. 237–246, 2022.
- [10] Y. Zhang, "Stock price prediction method based on XGBoost algorithm," 2022, pp. 595–603.
- [11] B. J. Vanstone, A. Gepp, and G. Harris, "Do news and sentiment play a role in stock price prediction?" *Appl Intell*, vol. 49, no. 11, pp. 3815–3820, 2019.
- [12] Y. E. Akdogan and A. Anbar, "More than just sentiment: using social, cognitive, and behavioral information of social media to predict stock markets with artificial intelligence and big data," *Borsa Istanbul Rev*, vol. 24, pp. 61–82, 2024.
- [13] W. Long, J. Gao, K. Bai, and Z. Lu, "A hybrid model for stock price prediction based on multi-view heterogeneous data," *Financ Innov*, vol. 10, no. 1, p. 48, 2024.
- [14] Y. Shi, Y. Zheng, K. Guo, and X. Ren, "Stock movement prediction with sentiment analysis based on deep learning networks," *Concurr Comput Pract Exp*, vol. 33, p. e6076, 2020.
- [15] N. Das, B. Sadhukhan, T. Chatterjee, and S. Chakrabarti, "Effect of public sentiment on stock market movement prediction during the COVID-19 outbreak," *Soc Netw Anal Min*, vol. 12, no. 1, p. 92, 2022.
- [16] P. Chen, Z. Boukouvalas, and R. Corizzo, "A deep fusion model for stock market prediction with news headlines and time series data," *Neural Comput Appl*, vol. 36, no. 34, pp. 21 229–21 271, 2024.
- [17] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," 2019, arXiv:1810.04805 [Preprint]. [Online]. Available: <https://arxiv.org/abs/1810.04805>
- [18] H. Lee, J. H. Kim, and H. S. Jung, "Deep-learning-based stock market prediction incorporating ESG sentiment and technical indicators," *Sci Rep*, vol. 14, no. 1, p. 10262, 2024.
- [19] A. A. Ariyo, A. O. Adewumi, and C. K. Ayo, "Stock price prediction using the ARIMA model," in *2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation*, 2014, pp. 106–112.
- [20] M. A. Lubis and S. Samsudin, "Using the random forest method in predicting stock price movements," *J Dinda Data Sci Inf Technol Data Anal*, vol. 5, no. 1, pp. 28–35, 2025.
- [21] L. Yan and Y. Xu, "XGBoost-enhanced graph neural networks: a new architecture for heterogeneous tabular data," *Appl Sci*, vol. 14, no. 13, p. 5826, 2024.
- [22] Z. Wang, Z. Hu, F. Li, and S.-B. Ho, "Learning-based stock market trending analysis by incorporating social media sentiment analysis," 2021, research Square [Preprint].
- [23] S. Zhang, Y. Chen, W. Zhang, and R. Feng, "A novel ensemble deep learning model with dynamic error correction and multi-objective ensemble pruning for time series forecasting," *Inf Sci*, vol. 544, pp. 427–445, 2021.
- [24] D. H. Wolpert, "Stacked generalization," *Neural Netw*, vol. 5, pp. 241–259, 1992.
- [25] B. Zadrozny and C. Elkan, "Transforming classifier scores into accurate multiclass probability estimates," in *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: Association for Computing Machinery, 2002, pp. 694–699.