Fiza Noor\textsuperscript{1} and Inam Ullah Khan\textsuperscript{2}

\textsuperscript{1}Department of Mathematics, COMSATS University Islamabad
\textsuperscript{2}Department of Computer Science, Edge Hill University

March 04, 2024
ADAPTEN: Adaptive Ensembles Leveraging Feature Engineering for Real-Time Market Analysis

Fiza Noor¹, and Inam Ullah Khan²

¹Department of Mathematics, COMSATS University Islamabad, Islamabad Campus, Pakistan.
²Department of Computer Science, Technology Hub, Edge Hill University, Ormskirk, United Kingdom.
Correspondence: inamullah@ieee.org

Abstract—In an era of significant economic volatility, time series forecasting is widely used to predict stock prices and guide investors in trading decisions. Nevertheless, existing data-driven techniques are unable to effectively handle the vast amount of financial data due to big data constraints such as nonlinearity, non-stationarity, heteroskedasticity, and unsynchronicity. A cohesive framework is also required for ensuring the smooth integration and synchronization of varied methodologies in time-series financial prediction tasks. To address this problem, this paper introduces a novel framework that investigates three ensemble strategies: blending, stacking, and voting, and selects the best method to perform the stock trend prediction task. Specifically, we deploy four distinct machine learning algorithms as the base learning model, each of which is uncorrelated and proficient in a different way depending on the task. The outputs of the basis classifiers are then combined using the adaptive boosting algorithm, a meta classifier, to give the final prediction results. To augment predictive models’s accuracy and generalization capabilities, we put forward strategies like feature engineering and Ridge regularization, which optimize the pertinence of data and curb overfitting. Our examination of five distinct case studies on Toronto Stock Exchange data reveals that the proposed multi-model ensemble method has superior performance compared to others.

Index Terms—Stock Trend Prediction, Feature Engineering, Big Data Analytics, Machine Learning, Stacking Ensemble.

I. INTRODUCTION

A. Background

The stock markets serve as the backbone of the contemporary global economy [1]. Present-day stock markets are tasked with the allocation of resources, the investment in high-quality businesses, and the efficient administration of savings. Trading represents merely a fraction of the multifarious purposes fulfilled by the stock markets. Furthermore, it serves as an indicator of the prevailing economic conditions in a country [2]. A considerable proportion of investors perceive it as a method through which they can accumulate wealth, attain financial independence, and potentially generate returns surpassing inflation. All these factors are indispensable for the long-term preservation of economic stability. From a broader perspective, the condition of the stock market can significantly influence the confidence of consumers and businesses, thereby affecting investment and spending decisions throughout the entire economic cycle [3].

In recent years, the stock markets around the globe have experienced a paradigm shift due to the fast-paced developments in technology, regulatory frameworks, and investor behaviors [4]. The markets have also been overwhelmingly impacted by digitalization and globalization. During the COVID-19 period, millions of people stuck at home took advantage of the new zero-commission mobile apps to trade shares, indexes, and options. The degree of interdependence between international market players has also increased. An instance in one region of the world has rapid implications for other regions and the overall global market, which highlights how closely interconnected the modern world is in terms of financial matters [5]. The advent of online trading platforms and investing applications has further helped in diminishing barriers, which pushed an unprecedented group of individual investors to participate in stock market operations [6]. Stock trading apps generated $22.8 billion in revenue in 2021, with over 130 million individuals used stock trading apps in 2021, a 49% increase on 2020 statistics [7]. These statistics show that the demographic composition of investors has shifted as younger and more diverse populations are engaging in stock trading like never before.

The forecasting of stock market fluctuations has developed into a significantly complex domain [8]. The stock-related databases contain a wide range of information, comprising historical stock prices, financial reports, as well as other data sources like social media sentiment, economic indicators, and geopolitical events [9]. However, this road has many bumpy potholes on its way. The chaotic nature of pricing data, non-linear dependencies among features, volatility, limited information, and unequal dominance between institutional and retail investors are all constraints towards accurate prediction [7], [10].

B. State of Research

The literature on stock trend prediction (STP) can be broadly classified into three categories: technical analysis, fundamental analysis, and innovative data-driven techniques.

Technical analysis is the prevailing methodology described in the literature, employing stock prices or indicators as inputs [11]. Technician analysts contend that stock prices incorporate every piece of newly acquired information, including macroeconomic variables and news. In order to forecast the stock market, it is thus sufficient to examine the patterns of price trends. These techniques make use of statistical models that have garnered significant attention for their capacity to forecast market trends and volatility, including ARCH, GARCH, and
ARIMA [12]. These models, often used by traders from a short-term perspective, undergo rigorous research to refine their predictive prowess with various indicators and oscillators like moving averages, MACD (moving average convergence divergence), and Bollinger bands [13]. Fundamental analysis, conversely, assesses the intrinsic value of a security by examining related economic, financial, and other qualitative and quantitative factors to improve understanding of market prediction intricacies [14]. Fundamental analysis involves economic indicators, industry factors, and company-specific factors and examines metrics like price-to-earnings (P/E) ratio, earnings per share (EPS), return on equity (ROE), and others for long-term growth and profitability potential.

While technical analysis is more concerned with the "when" of buying and selling based purely on market data, fundamental analysis focuses on the "why" based on a company’s actual business performance and economic factors. Many investors and traders use a combination of both techniques to inform their investment decisions, taking advantage of the insights each provides. Both methodologies, as underscored by [11], [13], [15], [16], are instrumental in advancing the field, albeit with inherent limitations that spur ongoing research and methodological innovation.

The emergence of ML and DL methods is gradually surpassing conventional methodologies. In particular, these models fall into three categories: supervised, unsupervised, and semi-supervised [17]. The supervised learning methods involve labeled data to predict future outcomes. The process is usually leveraged by techniques such as support vector machines (SVM), random forests (RF), neural networks, recurrent neural networks (RNN), especially long short-term memory networks (LSTM), etc. [18]. With the financial sector’s ongoing evolution, the amalgamation of advanced computational techniques with traditional financial analysis methods is anticipated to gain prominence. This convergence is composed of enhancing the sophistication and precise predictions of market movements, marking a significant leap in the art and science of financial prediction and analysis using data-driven modeling and control [19]. Although ML and DL techniques hold enormous potential for financial analytics, their implementation is fraught with difficulties. These include a tendency to overfit, substantial computational resource requirements, and the ‘black box’ nature of many ML algorithms, which present significant obstacles and often render financial decision-making processes less transparent [20], [21].

Research by Behera et al. [22] and Pulido et al. [23] demonstrates the effectiveness of ensemble techniques, which integrate multiple models to emphasize the enhanced accuracy of these approaches. Ensemble methods can capture a broader range of data patterns, leading to more robust forecasts for stock trends. Nonetheless, specific ensemble methods, particularly stacking and blending, remain underexplored in financial time series data prediction tasks, with the literature focusing mainly on bagging and boosting. This gap presents a research opportunity, as these lesser-studied techniques could offer additional benefits.

C. Contributions

This work delves into different STP issues, specifically focusing on binary classification tasks that aim to make precise
predictions about the upward and downward movements of various stocks on the Toronto Stock Exchange (TSE). The flowchart of the proposed ADAPTEN (adaptive ensemble) framework is shown in Algorithm 1. The framework explores a suite of well-known ensemble strategies and incorporates powerful ML models. Specifically, extra trees (EXT), quadratic discriminant analysis (QDA), naive bayes (NB), and XGBoost (XGB) are used as base learning models. During the classification procedure, each algorithm endeavors to segregate distinct data points and elucidate a class value. While each base model shows great potential in specific classification and regression tasks, they may surpass each other or exhibit flaws in other scenarios when used independently. Hence, the subsequent obstacles need to be tackled in order to achieve a precise forecast between the two patterns.

• **Challenge 1 (Limited Generalization):** One of the primary obstacles in deploying sophisticated ML models is their propensity to overfit on training data. This is mainly because of their deep and complex structures, which frequently identify noise as patterns and prevent the models from generalizing to new datasets. Xu’s [24] work draws attention to this problem when complex models learn very specific rules that only apply to the training data. The learning specificity makes models less effective when exposed to fresh, unseen data.

• **Challenge 2 (Model Complexity):** The ML techniques are slow to train [25]. Ensemble learning models such as EXT, QDA, and XGB are constrained by processing uncertain pieces of information. Also, the EXT and XGB algorithms have high computational costs due to adding additional randomness to the model while growing the trees. QDA, a statistical method, models the distribution of the features in each class using a Gaussian distribution with its own mean and covariance matrix [27]. As a result, its computational complexity gets intensive due to inverting multiple matrices and the requirements for more parameters to be estimated, especially in the context of financial datasets with many features.

To address the above, we propose a novel framework, ADAPTEN, that explores a trio of ensemble strategies: blending, stacking, and voting, as shown in Fig. 1. The ADAPTEN consists of four primary-level classifiers, namely EXT, QDA, NB, and XGB. These classifiers are unrelated and proficient at solving the problem using distinct approaches, and it is reasonable to anticipate that the combination of various methods would result in improved performance. The AdaBoost algorithm, which is a versatile and powerful meta classifier, combines the outputs of the four base classifiers to get the final prediction results. Our latest paper [20] is based on a stacked generalization to accurately predict electricity theft patterns using big data in smart grids. This research, however, expands upon the idea but explores two other ensemble techniques and applies feature engineering to pinpoint the most effective ensemble strategy to perform the final classification task. The main contributions of this paper are as follows:

- We present an ADAPTEN framework that evaluates stacking, voting, and blending methods to make accurate big data forecasts in STP. To the best of our knowledge, this is the first attempt in financial time series data analysis where such a tripartite approach is employed for accurate stock trend predictions.
- To achieve this framework, four ML models are deployed at the base level, and their predictions culminate in a higher-level AdaBoost algorithm for augmentation. It is important to note that the base classifiers are executed simultaneously, and there is a minimal disparity in processing time between them. We also enriched the proposed framework by crafting six innovative features and incorporating Ridge regularization to amplify its analytical power and generalization capabilities.
- The practicality and effectiveness of our approach are demonstrated through rigorous simulations and validation utilizing real-world data from stock exchange sources. The empirical evidence confirms that our proposal has superior performance compared to benchmark approaches.

The subsequent sections of this work are structured in the following manner: Sec. II and Sec. III discuss feature engineering and investigate ensemble techniques. Sec. IV presents the AdaBoost algorithm and its enhancements. Sec. V verifies the suggested framework using experimental results, and Sec. VI concludes this work.

### II. FEATURE ENGINEERING

The process of feature engineering, including extraction and selection, is delineated in this section. The original dataset
comprises only five basic features, such as open, low, adjusted close (Adj Close), volume, and high prices. In order to improve the model’s ability to make accurate predictions, we augmented the feature set by creating six more features, also known as technical indicators. The new features facilitate the detection of hidden patterns and trends for a profound comprehension of market dynamics [28]. Among the features that have been extracted are simple moving averages (SMA) and exponential moving averages (EMA), which are used to determine the direction of trends; moving average convergence divergence (MACD), which is used to analyze momentum; and weighted moving averages (WMA), which are used to strike a balance between their responsiveness and smoothing properties. The random forest (RF) technique is applied after the extraction of a feature set in order to pick the features that are most essential for the accuracy of the final classification. The subsequent two subsections will be devoted to providing a detailed description of these modules.

A. Feature Extraction

The EMA assigns greater weight to recent price data. This feature is very useful for identifying initial indications of trend shifts. The mathematical expression of this trend is given by,

\[
EMA_t = \left( \frac{2}{n + 1} \right) \times (Price_t - EMA_p) + EMA_p,
\]

where, \( EMA_t \) and \( EMA_p \) are the EMAs for the current and past periods, respectively. The expression \( \frac{2}{n+1} \) represents smoothing factor to determine the weight given to the most recent price, and \( n \) refers to the total number of periods.

Another important technical indicator is SMA, which offers a stable trend line to smooth daily price spikes. The 20-day SMA feature calculates the average price of a security over a 20-day period. It offers a consistent baseline to assess the trend. The mathematical formula for the 20-day SMA is given below,

\[
SMA_{20} = \frac{\sum_{i=1}^{20} \text{Close}_i}{20}
\]

Another important feature is WMA, which gives more importance to recent prices, therefore capturing the market’s prompt response to fresh information. This responsiveness to the latest market trends is crucial for making accurate predictions in a timely manner. The mathematical expression to calculate WMA is as follows,

\[
WMA = \frac{\sum_{i=1}^{n} (P_i \times w_i)}{\sum_{i=1}^{n} w_i}
\]

The MACD measures the interaction between short-term and long-term price movements and provides a dynamic gauge of market momentum. It is calculated by subtracting the 26-day EMA from the 12-day EMA of closing stock prices. The resulting figure serves as an indicator of overall market sentiment and trend strength. It is computed as,

\[
MACD = EMA_{12} - EMA_{26}
\]

Complementing the MACD, the signal line is the 9-day EMA of the MACD that acts as a smoother counterpart. It is often used to generate transaction signals when it crosses the MACD line and can be mathematically expressed as,

\[
Signal = EMA_{MACD9}
\]

The MACD histogram is another critical feature that represents the difference between the MACD and the signal line to provide a visual depiction of momentum oscillation. It is expressed as,

\[
\text{Histogram} = \text{MACD} - \text{Signal}
\]

The inclusion of extended features helps develop a model that not only effectively captures the fluctuations in the market but also performs well across various time periods and market situations.

B. Feature Selection

The feature selection process involves training an RF model on the expanded dataset and evaluating the feature importance score. Features with the highest importance scores are deemed most predictive of stock movement and are retained to balance the accuracy vs. computational complexity trade-off [29]. The importance of features is assessed by their contribution to the reduction of classification error in the constructed ensemble of decision trees. The importance of a feature \( j \) is quantified by the decrease in Gini impurity when the feature is used for node splitting. The mathematical calculation is as follows,

\[
I(j) = \frac{1}{N} \sum_{t=1}^{N} \sum_{i \in S_{t,j}} \Delta i(t, i),
\]

where, \( N \) represents total number of trees, \( I(j) \) is feature importance, \( S_{t,j} \) is the set of nodes in tree \( t \) that split on feature \( j \) and \( \Delta i(t, i) \) represents the decrease in Gini impurity for node \( i \) in tree \( t \) as a result of splitting on feature \( j \).

For a node \( i \) splitting on feature \( j \), the change in impurity can be articulated as the difference between the impurity of the node before the split and the weighted sum of the impurities of the subsequent child nodes. This is mathematically represented as,

\[
\Delta i(t, i) = G_{node} - \sum_{c \in \text{Children}(i)} \frac{N_c}{N_{node}} \times G_c,
\]

where, \( N_{node} \) and \( N_c \) represent the total number of samples and number of samples in child node \( c \), respectively. \( G_{node} \) is the Gini impurity of the node before the split, \( \text{Children}(i) \) is the child nodes resulting from the split of node \( i \). \( G_c \) is the Gini impurity of child node \( c \).

The impurity decrease \( \Delta i(t, i) \) for each split node is aggregated across all trees, and the feature importance scores are derived by averaging these decreases. Features are then ranked according to their importance scores to highlight those that are most influential in the prediction process.

\[
\text{Gini } \rightarrow \text{Importance Score} \rightarrow \text{Feature Selection}
\]
III. ENSEMBLE STRATEGIES

Ensemble learning seeks to enhance predictive performance by combining wisdom from two or more models. Based on a wide range of predictive modeling problems, ensemble modeling strategies can be broadly classified into three main categories: bagging, boosting, and stacking. In this framework, we assess the efficacy of each strategy to manage the idiosyncrasies of financial time series data and curb problems like overfitting, variance, and bias to achieve better prediction results (Nti, I. K.). In the next sections, we will examine the advantages and disadvantages of each technique, followed by the deployment of the most powerful ensemble variant to perform the final STP task.

1) Blending: Blending is an ensemble technique that involves training multiple models separately and then combining their predictions by taking an average or a weighted average [31]. For instance, if \( P_1, P_2, P_3, \) and \( P_4 \) represent the predictions from the base classifiers, the final prediction \( P \) using a simple average is computed as follows,

\[
P = \frac{1}{4}(P_1 + P_2 + P_3 + P_4)
\]

In weighted blending, the final prediction is determined by calculating the weighted total of the output from each classifier. In mathematical terms, this is represented as,

\[
P = w_1 \cdot P_1 + w_2 \cdot P_2 + w_3 \cdot P_3 + w_4 \cdot P_4
\]

Subject to the constraint \( \sum_{i=1}^{4} w_i = 1 \), the weights allocated to each classifier’s prediction are \( w_1, w_2, w_3, \) and \( w_4 \).

2) Stacking: Stacking, or stacked generalization, involves training a new model, known as a meta-learner, to aggregate the predictions of multiple base learners [20]. Each base learner is trained on the full dataset, and their predictions form a new dataset on which the meta-learner is trained. This approach can potentially capture complex relationships between base learners’ predictions and the target variable. In our investigation, we employ stacking to determine whether a meta-learner can enhance predictive accuracy by learning an optimal combination of base model predictions for the STP. Mathematically, if \( P_1, P_2, P_3, \) and \( P_4 \) represent the predictions from four base classifiers. The final prediction \( P \) using stacking can be expressed as,

\[
P = \text{AdaBoost}(P_1, P_2, P_3, P_4)
\]

Stacking aims to capture the complementary strengths of different models, which could potentially lead to improved predictive performance. However, it also introduces additional complexity and the risk of overfitting if the meta-classifier is too complex.

3) Voting: Voting is a simpler ensemble strategy where the final prediction is determined by a majority vote among the base classifiers. Each classifier votes for a class label, and the label with the most votes is selected as the final prediction. If \( V_1, V_2, V_3, \) and \( V_4 \) represent the votes from base classifiers, the final prediction \( P \) using voting can be expressed as,

\[
P = \text{mode}(V_1, V_2, V_3, V_4)
\]

Voting is robust and less prone to overfitting compared to more complex ensemble methods. We explore both hard voting, where the prediction with the majority count is chosen, and soft voting, where the probability estimates are averaged to make the final prediction. Both methods are mathematically encapsulated below,

\[
\hat{y} = \text{mode}\{M_1(x), M_2(x), M_3(x), M_4(x)\}
\]

\[
\hat{y} = \sum_{i=1}^{n} P_i(M_i(x))
\]

Voting is particularly effective when the base models are diverse. This allows automation in decision-making and, therefore, mitigates the impact of individual model biases.

IV. CLASSIFIER DESIGN AND OPTIMIZATION

A. Primary Classifiers

The STP framework utilizes a diverse set of primary classifiers, each possessing distinct attributes and capabilities, to guarantee a thorough examination of financial data. A review of these methods is as follows [7], [15], [27], [32].

EXT is a technique that aggregates the predictions made by several decision trees. It is closely related to other ensembles of decision tree methods, such as bootstrap aggregation (bagging) and random forest. The algorithm deployment for prediction tasks is straightforward because of the requirement of very few hyperparameters and smart heuristics to configure them. The underlaying principle for EXT is based on generating several decision trees without pruning from the training dataset, and then predictions are produced using majority voting. The stochastic selection of split points in the decision trees of the ensemble decreases their correlation, hence diminishing bias and offering a faster and more robust alternative for classification tasks.

The name “Naive Bayes” is derived from the fact that the algorithm computes the probability of a sample point for a particular class and allocates it to the class with the highest probability. Instead of trying to compute the probability of
each attribute value, it is assumed that they are conditionally independent given the class value. This assumption, which posits that the qualities do not interact, is highly improbable in real-world data. However, the technique exhibits unexpectedly good performance on data that does not adhere to this assumption.

QDA is a popular approach to handle data heteroscedasticity in most of the intricate classification problems. Unlike linear discriminant analysis (LDA), the algorithm probabilistically models the data and makes classification decisions based on the likelihood of a data point belonging to a particular class. This feature makes the algorithm more flexible in capturing the underlying complex structure of the data, which, otherwise, cannot be adequately modeled by linear decision boundaries. Nevertheless, the performance of QDAs can degrade in high-dimensional settings or when the sample size is not sufficient to estimate the covariance matrices accurately.

XGBoost is a popular ensemble technique that combines multiple weak learners (typically decision trees) to create a strong learner. The algorithm’s great prediction performance and ability to handle a variety of data sources make it a preferred choice for many classification, regression, ranking, and recommendation tasks. The high predictive strength, especially for structured or tabular data, stems from its gradient-boosting architecture. The architecture allows optimization of a customizable loss function and uses clever penalization of trees to prevent overfitting.

B. Secondary Classifier

AdaBoost is deployed as a meta classifier to ensemble the outputs of the base classifier and perform the final binary classification task of STP. The algorithm’s simplicity, coupled with the capability of reducing both bias and variance, makes it exceptionally effective for improving classification accuracy [16]. AdaBoost rectifies the misclassifications of its predecessors (base classifiers), thereby progressively refining the model’s predictive capabilities. Specifically, it incorporates the predictions of the major classifiers and assigns varying amounts of relevance (weights) to each prediction. At each iteration, the weights are modified, and a greater emphasis is placed on instances that were previously categorized incorrectly. To further optimize model performance, we use Ridge regularization, which effectively handles multicollinearity among predictors. The regularization stabilizes coefficient estimates and balances the bias-variance trade-off to boost model performance in intricate data scenarios.

1) Problem Formulation: We assume the TSE data is in a structured matrix format, denoted by A,

\[
A = \begin{pmatrix}
  a_{11} & a_{12} & \cdots & a_{1n} \\
  a_{21} & a_{22} & \cdots & a_{2n} \\
  \vdots  & \vdots  & \ddots & \vdots \\
  a_{m1} & a_{m2} & \cdots & a_{mn}
\end{pmatrix} = \begin{bmatrix}
  \tilde{t}_1 \\
  \tilde{t}_2 \\
  \vdots \\
  \tilde{t}_m
\end{bmatrix}
\]  

(14)

where each row represents a unique stock record and each column corresponds to a feature index. Specifically, \(a_{ij}\) denotes the \(j\)-th attribute of the data, representing the state at the \(i\)-th hour ahead of the trend that is to be predicted. The term \(\tilde{t}_k\) is a row vector representing the \(k\)-th stock record, and consists of features \([a_{k1}, a_{k2}, \ldots, a_{kn}]\) \(k \in [1, m]\). To aid the training process, the target feature \(y\) is assumed as follows,

\[
y = \begin{pmatrix}
  y_1 \\
  y_2 \\
  \vdots \\
  y_m
\end{pmatrix},
\]

(15)

where, \(y_k\) is a binary indicator (0 or 1). Specifically, \(y_k = 0\) signifies a downward movement, and \(y_k = 1\) indicates an upward movement in stock price. AdaBoost, as a meta-classifier, iteratively trains a series of weak classifiers on input data that is fed by base classifiers. At each iteration, it assigns higher weights to the instances that were misclassified by the previous weak classifiers, thereby focusing more on difficult-to-classify examples. The final classification decision is made by combining the predictions of all weak classifiers, and each weak classifier’s contribution is weighted by its accuracy. The ensemble prediction \(E(a_i)\) for each data point \(a_i\) is given by,

\[
E(a_i) = \sum_{t=1}^{T} \alpha_t \cdot b_t(a_i),
\]

(16)

where, \(a_i\) represents the feature set of the \(i\)-th instance, \(b_t\) is the prediction from the \(t\)-th weak learner, and \(\alpha_t\) is the weight assigned to this learner, reflecting its accuracy. The error rate \(\epsilon_t\) of the \(t\)-th weak learner is calculated as,

\[
\epsilon_t = \sum_{i=1}^{m} w_{i,t} \cdot I(y_i \neq b_t(a_i)),
\]

(17)

where, \(I(\cdot)\) is the indicator function, returning 1 if the condition is true and 0 otherwise, and \(w_{i,t}\) is the weight of the \(i\)-th instance at iteration \(t\).

The weight \(\alpha_t\) for each weak learner is determined by,

\[
\alpha_t = \frac{1}{2} \ln \left( \frac{1 - \epsilon_t}{\epsilon_t} \right)
\]

(18)

The weights of the training samples are updated to emphasize misclassified instances as,

\[
w_{i,t+1} = w_{i,t} \cdot \exp(\alpha_t \cdot I(y_i \neq b_t(a_i)))
\]

(19)

Weights are normalized after each iteration to ensure they sum to 1, maintaining a proper distribution over the instances. The model’s training is guided by minimizing the binary cross entropy loss (BCE) loss defined as,

\[
L_{BCE}(\hat{y}, y) = -\frac{1}{m} \sum_{i=1}^{m} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)],
\]

(20)

where, \(y_i\) is the true label and \(\hat{y}_i\) is the predicted probability of \(i\)-th instance.

To ensure the model’s effectiveness and generalizability, we also employ various other metrics such as accuracy, precision, recall, and the F1 score for evaluation. To counter potential
The AdaBoost algorithm constructs its robust classifier complexity and generalization capability [33]. Fundamentally, making robustness while maintaining a balance between model poses a structured constraint on the model, thereby promoting robustness while maintaining a balance between model complexity and generalization capability [33]. Fundamentally, the AdaBoost algorithm constructs its robust classifier \( F(x) \) by aggregating \( M \) individual base classifiers \( f_m(x) \) through a weighted sum. This relationship is mathematically expressed as,

\[
F(x) = \sum_{m=1}^{M} \alpha_m f_m(x),
\]

where, \( \alpha_m \) represents the weight assigned to the \( m \)-th base classifier. The original objective function in Eq. 20 is modified to add a regularization term to penalizes the square of the weights. This is mathematically expressed as,

\[
\min_{\alpha_1, \alpha_2, \ldots, \alpha_M} \left[ \frac{1}{N} \sum_{i=1}^{N} L(y_i, F(x_i)) + \lambda \sum_{m=1}^{M} \alpha_m^2 \right],
\]

where, \( N \) denotes the total number of training samples, \( L(y_i, F(x_i)) \) is the loss function, and \( \lambda \) represents the regularization parameter. Eq. 22 still aims to minimize prediction errors (the first term). However, it now includes a penalty on the magnitude of the weights (the second term). The regularization term, \( \lambda \sum_{m=1}^{M} \alpha_m^2 \), introduces a penalty term to the loss function proportional to the square of the magnitude of the coefficients. This not only helps in reducing the model’s complexity by shrinking parameter estimates but also stabilizes the predictions made by the ensemble. The weight update rule is mathematically expressed as,

\[
w_{i,t+1} = w_{i,t} \cdot \exp \left( -\alpha_t \cdot y_i \cdot h_t(x_i) + \lambda \alpha_t^2 \right).
\]

V. EXPERIMENTAL SETUP

A. Dataset Description and Preparation

The evaluation of the proposed framework was carried out on the Google Collaboratory (Cloab) in conformity with the framework architecture delineated in Section IV. The investigation centered on the Toronto Stock Exchange. The dataset spans from 2001 to 2023 [34] and contains 5579 daily logs to capture the ebb and flow of the market. Specifically, it is characterized by 3034 instances indicating a market upswing and 2545 instances marking a downturn. The attributes of the data are already explained in Section II. For the purposes of the study, the dataset was strategically split, allocating 75% to the training phase and the remaining 25% to testing to validate the predictive prowess of the model.

B. Evaluation Metrics

In order to assess the accuracy of predictions, the framework utilizes the confusion matrix (CM) as its primary instrument to capture TP, TN, FP, and FN values. TP stands for true positive values to indicate instances where both the actual and predicted positive value (upward trend) are correctly identified. In the matrix, this is represented as (1, 1). TN stands for true negative values, indicating that the classifier accurately predicted downward trend values, represented as (0, 0). The abbreviation FP stands for false positive, which refers to a situation when the classifier wrongly predicts a negative value (downward trend in our case) as positive. This is represented as (0, 1). FN is an abbreviation for false negative, indicating when the model incorrectly predicts a negative trend despite the actual trend being positive, denoted as (1, 0). The values in the CM are used to calculate performance metrics such as:

\[
\text{Precision} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN}, \quad \text{F1-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}, \quad \text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}.
\]
C. Results and Analysis

1) **Experiment 1: Feature Engineering Process Evaluation:** This case study evaluates how feature engineering process influences the accuracy of classification directly. Initially, the TSE dataset contained five features, which are Adj close, volume, open, low, and high. To enhance the model’s predictive capability, we enriched the dataset with six additional features known for their relevance in similar analytical contexts. The new crafted features are MACD, MACD signal, MACD histogram, WMA, 20-day SMA, and 20-day EMA. We employed the random forest (RF) algorithm to rank all features according to their importance in the predictive process. Notably, the MACD, both the histogram and signal components, stands out as key features when contrasted with the existing features such as open, low, and high values, as shown in Fig. 3. The practical application of these enhancements is exhibited in Figure 4, where the MACD curve’s crossover with the signal line provides actionable insights about buy or sell opportunities. By adjusting the threshold value of features, the number of features can be increased or decreased to train the model. The threshold value allows us to fine-tune the balance between model accuracy and computational efficiency. With more features, a model’s accuracy is potentially elevated, but at the cost of higher computational demand. By judiciously controlling the feature selection process, an optimal blend of high predictive accuracy and manageable computational requirements can be simultaneously achieved.

2) **Experiment 2: Ensemble Models Evaluation:** This case study compares the performance of three ensemble methods such as blending, stacking, and voting (soft and hard). According to the results in Fig. 5, the stacked method demonstrates a notable AUC score, registering an 85% value, compared to its counterparts, which incurred 80% (hard voting) and 81% (blending) scores, respectively. Additionally, when accessing the computational time in Table I, the stack method is more efficient, requiring only 13s, while the blending and voting methods require 30s and 45s, respectively. The stacked model’s superior predictive accuracy is due to the integration of diverse models that help it capture various aspects of data. In contrast, blending and voting methods employ simple or weighted averages, which potentially lead to suboptimal performance outcomes on complex problems.

3) **Experiment 3: Base Models Evaluations:** For our analysis, we applied four distinct classifiers in the first stage individually, which were EXT, NB, QDA, and XGB. Notably, the performance of the meta classifier is entirely contingent

### Table I: Comparative Analysis of Ensemble Method Efficiencies

<table>
<thead>
<tr>
<th>Methods</th>
<th>Acc</th>
<th>Prec</th>
<th>AUC</th>
<th>F1</th>
<th>MCC</th>
<th>kappa</th>
<th>Spe</th>
<th>J</th>
<th>MDA</th>
<th>H-L</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vote-Soft</td>
<td>0.78</td>
<td>0.77</td>
<td>0.81</td>
<td>0.81</td>
<td>0.62</td>
<td>0.62</td>
<td>0.73</td>
<td>0.71</td>
<td>0.67</td>
<td>0.18</td>
<td>42</td>
</tr>
<tr>
<td>Vote-Hard</td>
<td>0.73</td>
<td>0.76</td>
<td>0.80</td>
<td>0.75</td>
<td>0.59</td>
<td>0.59</td>
<td>0.77</td>
<td>0.68</td>
<td>0.65</td>
<td>0.20</td>
<td>45</td>
</tr>
<tr>
<td>Blend</td>
<td>0.78</td>
<td>0.74</td>
<td>0.81</td>
<td>0.81</td>
<td>0.64</td>
<td>0.63</td>
<td>0.70</td>
<td>0.27</td>
<td>0.68</td>
<td>0.18</td>
<td>30</td>
</tr>
<tr>
<td>Stack</td>
<td>0.81</td>
<td>0.82</td>
<td>0.84</td>
<td>0.83</td>
<td>0.69</td>
<td>0.69</td>
<td>0.82</td>
<td>0.75</td>
<td>0.73</td>
<td>0.15</td>
<td>13</td>
</tr>
</tbody>
</table>

### Table II: Performance Comparison of Base Classifiers

<table>
<thead>
<tr>
<th>Classifiers</th>
<th>Acc</th>
<th>Prec</th>
<th>AUC</th>
<th>MAR</th>
<th>F1</th>
<th>MCC</th>
<th>kappa</th>
<th>Spe</th>
<th>Sen</th>
<th>J</th>
<th>MDA</th>
<th>H-L</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXT</td>
<td>0.69</td>
<td>0.70</td>
<td>0.72</td>
<td>0.69</td>
<td>0.69</td>
<td>0.71</td>
<td>0.37</td>
<td>0.74</td>
<td>0.72</td>
<td>0.52</td>
<td>0.42</td>
<td>0.33</td>
<td>1.45</td>
</tr>
<tr>
<td>NB</td>
<td>0.53</td>
<td>0.77</td>
<td>0.55</td>
<td>0.51</td>
<td>0.51</td>
<td>0.64</td>
<td>0.01</td>
<td>0.01</td>
<td>0.23</td>
<td>0.77</td>
<td>0.47</td>
<td>0.42</td>
<td>0.46</td>
</tr>
<tr>
<td>QDA</td>
<td>0.82</td>
<td>0.93</td>
<td>0.79</td>
<td>0.81</td>
<td>0.81</td>
<td>0.85</td>
<td>0.65</td>
<td>0.54</td>
<td>0.69</td>
<td>0.93</td>
<td>0.74</td>
<td>0.70</td>
<td>0.17</td>
</tr>
<tr>
<td>XGB</td>
<td>0.75</td>
<td>0.79</td>
<td>0.77</td>
<td>0.75</td>
<td>0.75</td>
<td>0.78</td>
<td>0.50</td>
<td>0.70</td>
<td>0.70</td>
<td>0.63</td>
<td>0.60</td>
<td>0.24</td>
<td>4.21</td>
</tr>
</tbody>
</table>

Fig. 5: Performance Assessment of Ensemble Models

Fig. 6: AUC Score Assessment of Standalone Meta Classifier
upon the performance of the basis classifiers. Therefore, the selected set of base classifiers is founded on distinct underlying principles, namely ensemble methods, probabilistic modeling, and gradient boosting, to capture a broader spectrum of data intricacies in financial data. From the performance results shown in Table II, it can be seen that the NB classifier, despite being the fastest with a processing time of 1.06 units, showed limited accuracy (0.53) and precision (0.55), likely due to its simplistic assumption of feature independence. The EXT classifier offered moderate performance with an accuracy of 0.69, but its method of introducing randomness did not fare well against data with interdependent features. The XGB proved its mettle in model enhancement through iterative tree addition and achieved a notable accuracy of 0.75; however, it also incurred the highest computational cost at 4.21 units. Standing above the rest, the QDA classifier combined efficiency with high accuracy (0.82), recall (0.93), and precision (0.79), excelling especially in defining decision boundaries crucial for detecting shifts in stock market trends.

4) **Experiment 4: Meta Models Evaluation:** We investigates six different ML algorithms, such as AdaBoost, DT, KNN, LR, LGBM, and SVC, to act as meta learner and perform the next level of effective prediction. As shown in Fig. 6, both the DT and KNN algorithms underperform with highly correlated features and fail to accurately differentiate between trend signals. KNN is known to be a lazy learning approach that solely relies on stored outcomes and the 'k' closest samples for predictions. LR and SVC, both linear models, fall short of AdaBoost’s performance, particularly due to their inability to handle data non-linearity and being sensitive to the optimal values of certain hyperparameters. The AdaBoost classifier excels, closely mirrors actual trends, and iteratively enhances weak learners to minimize total error, effectively managing the ensemble and avoiding overfitting issues. AdaBoost’s superior accuracy, owing to its ability to learn and discover hidden structures, automatically solidifies its position as the preferred metaclassifier for our framework.

5) **Experiment 5: R-Stack’s Adaptiveness Comparison with Benchmark Algorithms:** This case study presents a comparative analysis of Regularized Stack’s (R-Stack) performance against the standard stacking model, as illustrated in Fig. 7. Without employing Ridge regularization, Fig. 7(a) displays a severe performance loss when classifying upward and downward movements, whereas the values of TN, FN, FP, and TP are 37.28%, 7.31%, 7.81%, and 47.60%, respectively. In STP, the TP value needs to be improved to capitalize on all potential profitable opportunities. To address this challenge, we utilize Ridge regularization, which effectively curbs overfitting problems and improves model generalization and function fitting capabilities on unseen data, as seen in Fig. 7(b). The bar chart in Fig. 8 presents that the R-stack model outperforms all other competing models in other well-known performance matrices, including accuracy, precision, and F1 score, and requires very little average computational time to perform the final classification task.

The volume of training data plays a pivotal role in achieving the ideal balance between underfitting and overfitting. To explore this, we conducted a series of experiments, varying the training ratio between 60%, 70%, and 80% to determine the R-Stack model’s performance with different training set sizes. Table V presents the results, which clearly indicate that the R-Stack model consistently surpasses other ensemble methods across all training data sizes. Notably, the R-Stack model reaches its peak performance at a 70% training ratio, with a maximum AUC value of 0.86, thereby demonstrating superior predictive capability compared to the benchmark algorithms. Traditional ensemble methods also exhibited performance improvements proportional to the increase in training data size.

VI. CONCLUSIONS

In this paper, we have investigated different types of ensemble strategies for stock trend prediction problems. An adaptive ensemble (ADAPTEST) framework that explores blending, stacking, and voting ensemble strategies for accurate prediction of stock trends using realistic data from the Toronto Stock Exchange. Our methodology consists of four base-learning ML models and an adaptive boosting algorithm to ensemble the outputs of base-learning machine learning models. We also performed feature engineering to first craft six new features from existing data. Afterwards, a random forest algorithm is
used to select the six best features and make the prediction task computationally efficient. To curb problems like overfitting in stacked methods, Ridge regularization is employed. The numerical results have demonstrated that the proposed ADAPTEN framework based on stacked generalization is more accurate and computationally efficient than other ensemble strategies and base models when used independently.

REFERENCES


