Predicting Stock Actions Using Machine Learning

Rafi Biswas
College of Engineering
UC Davis
ribiswas@ucdavis.edu

Stephanie Espinoza Gutierrez
College of Engineering
UC Davis
skespinozagutierrez@ucdavis.edu

Jennifer Liu
College of Engineering
UC Davis
zijliu@ucdavis.edu

Jashan Singh
College of Engineering
UC Davis
jdsing@ucdavis.edu

Victoria Zhang
College of Engineering
UC Davis
vynzhang@ucdavis.edu

***Abstract*—We propose a binary classification framework to predict whether S&P 500 stocks will exhibit trading activity (action) versus inactivity over a 21-day horizon. Our model leverages sixteen features drawn from technical indicators, financial fundamentals, and ESG risk scores. We compare linear regression, weighted multinomial logistic regression, and a 200-tree Random Forest; the Random Forest achieves 0.88 accuracy. Feature analysis reveals that price-based technical indicators dominate predictive power. Although the model captures most actionable signals, elevated false positives motivate future work on probability calibration, ensemble methods, and the integration of sentiment data.**

I. Introduction and Background

The stock market has traditionally been viewed through the lens of the Efficient Market Hypothesis (EMH), which hypothesizes that all publicly available information is already reflected in stock prices—rendering consistent outperformance an impossibility. For many decades, this idea restricted both academic speculation and institutional strategy. However, over the past ten years, this theory has been increasingly opposed by the increase of high-frequency data, scalable compute infrastructure, and exponential innovations in machine learning. Today, sophisticated models—ranging from reinforcement learning agents to Convolutional Neural Networks (CNN's) and Long Short-Term Memory (LSTM) networks—have shifted from academic speculation to industry standard, enabling traders to exploit subtle, short-lived inefficiencies at microsecond granularity. In such a competitive landscape, even a 1% improvement in the level of precision can translate into millions in profits for institutional portfolios.

Our work is enveloped within this modern landscape, with a goal of not disproving EMH wholesale, but constructing a meticulous, data-driven approach to choosing economically actionable stock movements over realistic time frames. Instead of chasing next-day predictions—a typical habit in research literature—we focus on medium-horizon signals (21-day returns) that are consequentially enough to overcome transaction costs, spreads, and slippage, and therefore, achievable in real-world trading.

In the early 21st century, stock prediction heavily relied on classical machine learning algorithms such as Support Vector Machines (SVMs) and boosted trees, commonly utilizing hand-made technical indicators. Around 2015, the machine learning field experienced a shift to deep learning methods, including CNN's and LSTM's, which ensured more optimal performance through automatic feature extraction from price series "images." Yet these models often introduced other complications: interpretability declined, compute costs soared, and look-ahead bias became a persistent risk. Recent studies have started embracing richer data representations—incorporating company fundamentals, Environmental, Social, and Governance (ESG) scores, and market sentiment—via advanced architectures like tabular transformers and embedding-based learners. Still, the trade-off between model complexity and interpretability remains unresolved.

Our project deliberately targets this middle ground. Rather than adopting opaque deep-learning models with limited explainability, we use a Random Forest classifier trained on a diverse yet interpretable feature set spanning technical indicators, fundamentals, and ESG risk factors. This choice enables us to model nonlinear interactions while retaining transparency through feature importance analysis—a critical aspect for both academic evaluation and practical deployment.

We frame the prediction task as a binary classification problem: given the information at time t, predict whether the 21-day return will exceed ±1%, a threshold motivated by real-world trading heuristics and risk controls. This framing also addresses a significant challenge in financial prediction: class imbalance. Most trading days see minimal price changes and are thus classified as "Hold," while substantial price movements—those warranting a Buy or Sell—are much rarer. As a result, naive models tend to overfit the majority class. To mitigate this, we implement class-weighted versions of logistic regression and Random Forests, ensuring both recall and precision remain meaningful across classes.

The foundation of our model is a cleaned and integrated dataset composed of over 800,000 daily entries from S&P 500 stocks, combining split-adjusted price data, annual financials, and ESG risk scores. These were standardized and filtered, with additional technical features created via rolling calculations (e.g., momentum, volatility, RSI). Exploratory analysis revealed skewed distributions, collinearities, and class imbalance, all of which guided our modeling decisions.

Three models were trained and evaluated: a linear regression (as a baseline), a class-weighted logistic regression (to introduce probability-based modeling), and a Random Forest classifier (to capture complex interactions). The Random Forest ultimately achieved the best performance—94–96% recall on actionable days, 52% on hold days, and a balanced accuracy of 0.74—demonstrating strong real-world viability while retaining interpretability. Feature importance plots confirmed that while technical indicators dominate, fundamental and ESG metrics meaningfully complement predictive performance.

By balancing practicality, performance, and explainability, our project contributes a grounded, replicable framework for medium-horizon stock action prediction, potentially bridging the gap between academic research and professional trading systems.

II. Dataset Description

The machine learning models were trained on three different datasets. Each dataset provides unique and necessary information about the S&P 500 companies. The dataset contains information about the daily stock price after adjusting for stock splits, company fundamentals, and ESG (Environmental, Social, Governance) risk scores. All of this dataset was cleaned, normalized, and adjusted to reduce the influence of outliers.

A. Price Data — `prices_cleaned.csv`

This dataset contains daily stock market data after adjusting for stock splits for all the S&P 500 companies. The values were standardized using z-score normalization, so the features are compatible across different datasets. There are 832,152 rows and 7 columns. And the data covers from 2010 to 2016.

- **open, close, low, high**: Each row contains normalized daily opening, closing, lowest, and highest stock prices.
- **date**: Trading date (YYYY-MM-DD)
- **symbol**: Ticker symbol of the stock (e.g., NVDA, AAL)
- **volume**: Normalized trading volume

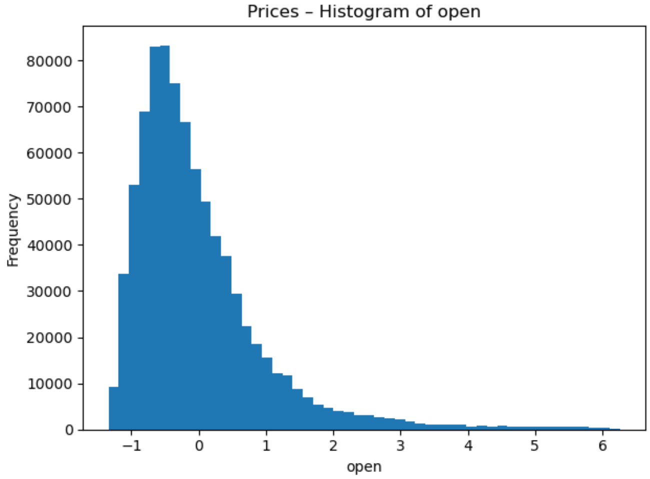

Fig. 1. Histogram of the `open` column from `prices_cleaned.csv`. The distribution is right-skewed, reflecting the z-score normalized values of stock opening prices.

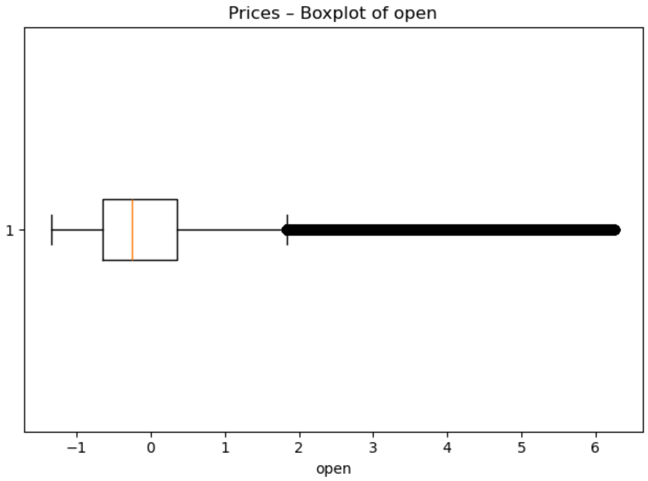

Fig. 2. Boxplot of the `open` column from `prices_cleaned.csv`. The distribution shows a concentration around the median with a long tail of outliers, consistent with the right-skewed shape observed in the histogram.

B. Fundamentals Data — `fundamentals_cleaned.csv`

This dataset contains an annual financial report of each S&P 500 company. There are 1,617 rows and 24 columns. And the data covers from 2003 to 2017. It includes:

- **Earnings per share (EPS)**
- **Net income**
- **Total revenue**
- **Total Assets / Liabilities / Equity**
- **Current and Quick Ratios**
- **Operating Cash Flow and CapEx**

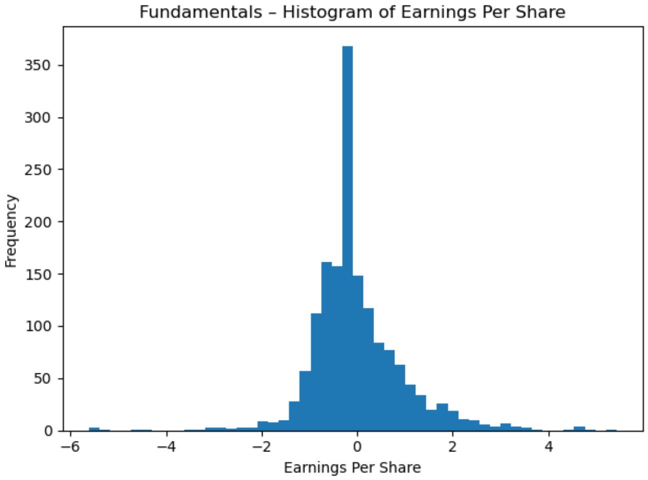

Fig. 3. Histogram of the `Earnings Per Share` column from `fundamentals_cleaned.csv`. The distribution is approximately symmetric with a high central peak at zero, reflecting standardized earnings across companies.

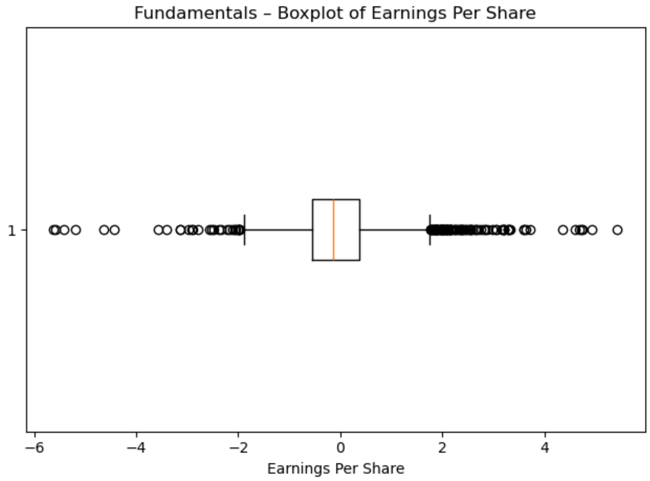

Fig. 4. Boxplot of the `Earnings Per Share` column from `fundamentals_cleaned.csv`. The distribution is mostly symmetric with multiple outliers on both sides, which is expected given the normalization and range of company profitability.

C. ESG Risk Data — `esgRisk_cleaned.csv`

This dataset provides environmental, social, and governance risk scores for each S&P 500 company. There are 496 rows and 8 columns. Dataset includes:

- **Total ESG Risk Score**
- **Environment Risk Score**
- **Social Risk Score**
- **Governance Risk Score**
- **Controversy Score (proxy for public/media reputation**
- **It also includes symbol, sector, and industry**

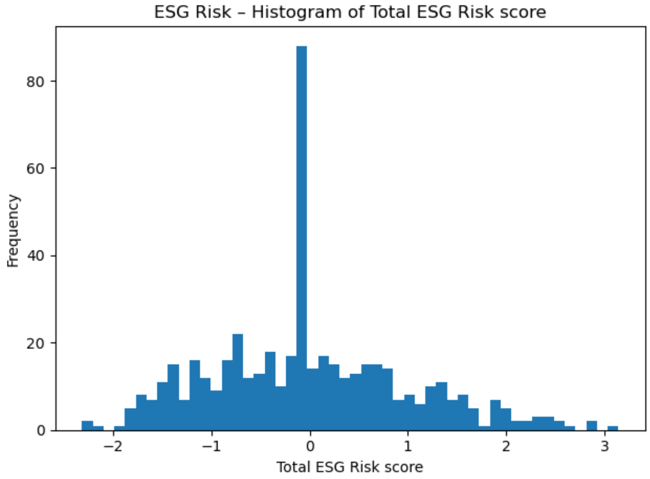

Fig. 5. Histogram of the `Total ESG Risk Score` from `esgRisk_cleaned.csv`. The distribution is roughly centered with a sharp spike at zero, likely due to a large number of companies receiving average ESG ratings. This feature was z-score normalized.

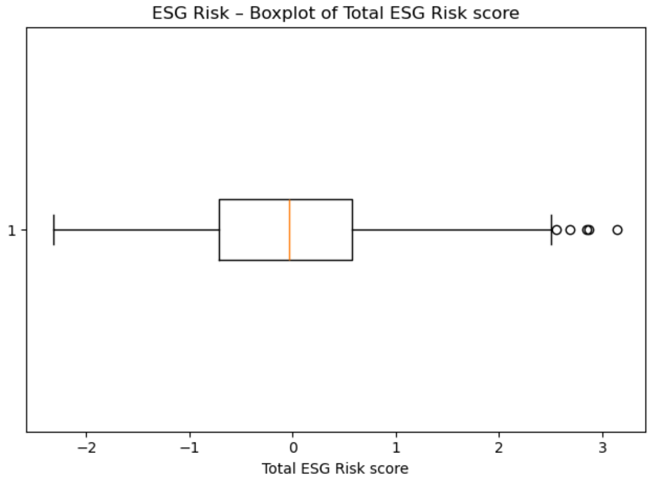

Fig. 6. Boxplot of the `Total ESG Risk Score` from `esgRisk_cleaned.csv`. Most values cluster near the median with only a few outliers, suggesting a fairly tight distribution of ESG risk across companies.

D. Integration and cleaning

All three datasets are merged using the company's symbol as the primary key. We kept rows that contain data across all three sources. We also create some new features using rolling calculations on the price data. After these calculations, if there were missing values, it was removed.

- **Momentum (5-day)**: Ratio of close prices over 5 days
- **Moving average (5-day)**: Smoothed price trends
- **Volatility (5-day)**: Rolling standard deviation of returns
- **Acceleration**: Second difference of close prices
- **RSI (14-day)**: Relative Strength Index using Wilder's smoothing

Any NaNs that resulted from these rolling computations were dropped from the final dataset. The final version of the design matrix contains 16 features (9 technical, 3 financial, 4 ESG) used for training and evaluation.

E. Exploratory Data Analysis (EDA)

To better understand these datasets, we performed comprehensive exploratory data analysis.

- **Missing Values**: All datasets were checked for missing, zero, or NaN entries. If a row had missing data in any of the key columns, we either tried to recover it from other datasets or removed it entirely to avoid errors later on.
- **Data Types**: We made sure each column had the correct data type—for example, dates were stored as dates, numeric values as floats, and company names or tickers as strings.
- **Outlier Detection**: We used z-scores to find extreme values in features like trading volume and earnings. Any outliers that could distort the model were either capped or removed.
- **Distribution Analysis**: We visualized the distributions of key numeric features using histograms and KDE plots.

This helped confirm that the normalization worked and revealed any skewed variables.

- **Correlation Heatmaps**: We generated heatmaps to see how features were related to each other. This helped us spot redundant features or strong correlations that could affect model performance.
- **Class Imbalance**: We looked at the distribution of our prediction labels—Buy, Hold, and Sell—and found a significant imbalance. This insight led us to use class weighting later in training to prevent the model from favoring the majority class.

III. Literature Review

Subasi *et al.* [1] tested different machine learning algorithms with the National Association of Securities Dealers Automated Quotations System (NASDAQ), New York Stock Exchange (NYSE), Nikkei, and Financial Times Stock Exchange (FTSE) as datasets to find the best machine learning algorithm for stock prediction. The classification methods they evaluated included Artificial Neural Networks (ANN), K-Nearest Neighbor (k-NN), Support Vector Machine (SVM), C4.5 Decision Tree, Random Forest (RF), Bagging, and AdaBoost [1]. Subasi *et al.* [1] trained models using 10-fold cross-validation on both normal and leaked datasets. Since each dataset calculates market indices and features differently, the performance of classification methods varied [1]. Overall, Subasi *et al.* [1] conclude that leaked datasets showed much better performance than normal datasets, with Random Forest and Bagging outperforming other methods on the leaked data. For the normal datasets, prediction accuracy from highest to lowest was: SVM (62%) $>$ k-NN (56%) $=$ ANN (56%) $>$ AdaBoost (54%) $>$ Random Forest (53%) $=$ Bagging (53%) $>$ Decision Trees (49%).

Sahib, Elkina, and Taher [2] transform traditional technical indicators into 15x15 "indicator images" over 21-day windows, feeding these representations into a two-stage deep model - a 2D-CNN to extract spatial patterns among indicators, followed by an LSTM to capture their temporal evolution - and use the combined features to classify each window as Buy, Hold or Sell. Their experiments show that this CNN-LSTM architecture is more robust than using only CNN or LSTM. Although their novel way of using image-based approach to capture the complex interactions among technical metrics, our project instead retains all features in tabular form and begins with interpretable models (logistic regression, Random Forest) before exploring deeper architectures.

Zheng and Jin [3] focused on short-term price prediction using daily open, high, low, close prices and trading volume for 82 randomly selected NYSE stocks from the Alpha Vantage API. They engineered 17 technical indicators from 11 preprocessing models and benchmarked a range of ML methods (logistic regression, Bayesian networks, simple neural nets, SVM with RBF kernel, and support vector regression) using both mean squared error and directional error rates . This study underscores the benefit of combining raw time-series features with classical ML techniques—an idea we adopt in our feature-engineering pipeline and baseline selection. However, whereas Zheng and Jin [3] limit their scope to a modest set of NYSE stocks and purely temporal indicators, our project will integrate static company fundamentals (e.g., revenue growth, sector) alongside technical metrics, and prioritize model interpretability via logistic regression and Random Forest before exploring deeper architectures.

IV. Proposed Methodology

In order to achieve the best prediction of stock behavior, the model was trained, optimized, and evaluated using three different modeling approaches to select the one with the highest precision. Specifically, we first sort the cleaned daily data (`prices_cleaned.csv`) by *symbol* and *date*. For every record we created a binary label over a 21-day horizon ($h = 21$) based on each stock's rate of return [4] using

$$r_{t+h} = \frac{\text{close}_{t+h}}{\text{close}_t} - 1, \tag{1}$$

and marked an observation **1** (actionable) when $|r_{t+h}| > 0.01$; otherwise it was labeled **0** (hold). The threshold of 0.01 means a predicted rate of return must exceed $\pm 1\%$ to be considered significant enough for taking action (Buy / Sell), which is similar to the idea of "1% rule" in trading [5]. Because the future price appears only in the numerator, this construction avoids look-ahead bias.

Feature matrix. The predictor set comprised nine technical indicators (*close*, *volume*, *ret_1d*, *ret_2d*, *momentum_5d*, *ma_5d*, *vol_5d*, *accel*, *rsi_14*), three fundamental metrics (*Earnings Per Share*, *Total Revenue*, *Net Income*) imported from `fundamentals_cleaned.csv`, and four ESG risk scores (*Total ESG Risk score*, *Environment Risk Score*, *Social Risk Score*, *Governance Risk Score*) imported from `esgRisk_cleaned.csv`.

All predictors rely solely on t or earlier data, so no future information leaks. Rows containing NaNs—either from the rolling calculations or the joins— were dropped. The resulting design matrix comprised sixteen features. An 80/20 stratified split produced disjoint training and test sets.

All predictors rely solely on t or earlier data, so no future information leaks. Rows containing NaNs—either from the rolling calculations or the joins— were dropped. The resulting design matrix comprised sixteen features. An 80/20 stratified split produced disjoint training and test sets.

The two return features are simple percentage changes; *momentum_5d* is the price ratio $\text{close}_t/\text{close}_{t-5}$; *ma_5d* and *vol_5d* are a 5-day moving average and rolling standard deviation respectively; *accel* is the second difference Δclose^2; *rsi_14* is the 14-day Relative Strength Index computed with Wilder's smoothing. All indicators rely solely on t or earlier prices, so no future information leaks into the predictors. Rows containing NaNs after the rolling operations were dropped. The resulting design matrix comprised nine features. An 80 / 20 stratified split produced disjoint training and test sets.

A. Linear Regression

In [3], linear regression was used as a baseline model for stock prediction due to its simplicity. Similarly, we implement a linear regression model for comparative analysis. While linear regression is traditionally designed to model the linear relationship between a continuous dependent variable and multiple independent variables, it can be used for binary classification tasks by setting a threshold for continuous outputs, which enables the model to decide whether to act (Buy / Sell) or stay (Hold).

Considering the nature of linear regression, we only selected six features that are likely to have continuous and linear effects on the target variable, which including four technical indicators (daily return, 5-day momentum, 5-day moving average, 5-day volatility, and the 14-day RSI), one fundamental factor (net income) and one ESG factor (total ESG risk score).

As for the implementation part, we trained a custom linear regression model using mini-batch gradient descent (batch size = 32) in order to minimize mean squared error (MSE) for optimization. Compared to full-batch training (batch size = size of the training dataset), mini-batches can converge faster by updating weights more frequently. Meanwhile, mini-batches offer more stable gradient updates than stochastic gradient descent (batch size = 1), which helps to reduce the impact of noise during the gradient descent. The model minimizes mean squared error over multiple epochs and the model parameters θ are updated iteratively using Eq. (2), which can reliably reflect the model's improvement (with decreasing MSE values).

$$\theta_i = \theta_i - \eta \cdot \nabla_{\theta_i}\text{MSE} \tag{2}$$

where:

- θ_i denotes the i-th weight parameter,
- η denotes the learning rate,
- ∇_{θ_i}MSE represents the partial derivative of the MSE loss function with respect to θ_i.

To align with the binary action-label framework introduced in Eq. (1), we later applied a threshold ($|\text{value}| > 0.01$) to transform the continuous outputs into actionable (Buy/Sell) or non-actionable (Hold) labels based on the "1% rule" [5].

After finishing training, we evaluate the model's performance using a classification report, which provides the precision, recall, and F1-score for each class (Action / No-Action). In addition, we visualize the classification results by plotting the confusion matrix of the outcomes.

B. Logistic Regression

Logistic regression provides an interpretable, probability-oriented baseline that bridges the gap between the purely linear model and the non-linear Random Forest. Instead of minimising mean-squared error, it optimises a class-weighted binary cross-entropy, producing calibrated probabilities for the *Action* vs. *Hold* decision.

Feature set. We reuse the full sixteen-column matrix—nine technical indicators, three fundamental metrics, four ESG scores—to keep the comparison with the Random Forest strictly algorithmic.

Pre-processing. All numeric columns are standardised via `StandardScaler`. Rows containing NaNs after merging and rolling computations ($\approx 14\,\%$) are dropped, matching the Forest pipeline.

Training objective. With label $y \in \{0, 1\}$ and feature vector $\mathbf{x}$, the model predicts

$$\hat{p} = \sigma(\boldsymbol{\beta}^{\mathsf{T}}\mathbf{x}) = \frac{1}{1 + e^{-\boldsymbol{\beta}^{\mathsf{T}}\mathbf{x}}},$$

and minimises the class-weighted loss

$$\mathcal{L}(\boldsymbol{\beta}) \;=\; -\sum_{i=1}^{N} w_{y_i}\big[y_i \log \hat{p}_i + (1-y_i)\log(1-\hat{p}_i)\big] \;+\; \lambda\|\boldsymbol{\beta}\|_2^2,$$

where $w_{y_i} \;\propto\; 1/\text{freq}(y_i)$ compensates for the 4.9:1 *Action–Hold* imbalance and λ controls L_2 regularisation.

Implementation.

- *Solver:* LBFGS, `max_iter`=1000.
- *Class weight:* `'balanced'` (inverse frequency).
- *Hyper-parameter search:* $C = 1/\lambda \in \{10^{-3}, 10^{-2}, 10^{-1}, 1, 10\}$ tuned via five-fold `TimeSeriesSplit`.
- *Split:* chronological 80/20; most recent 20% ($n = 163\,820$) held out exactly as for the Forest.

Computation and artefacts. and scaler are serialised to outputs/LogReg_model_binary.pkl and outputs/LogReg_scaler.pkl. Post-fit outputs include:

- a 2×2 confusion matrix (Fig. 9);
- the full classification report (Table III);
- a bar plot of the ten largest standardised coefficients, mirroring the Gini-importance plot in Fig. 11.

C. Random Forest

We employ a Random Forest classifier to capture the non-linear interactions among our engineered predictors. This ensemble method grows 200 decision trees on bootstrap samples, randomly subsampling $\sqrt{d}$ features at each split, which guards against overfitting and automatically models complex feature dependencies. Out-of-bag error provides an internal estimate of generalization, and the resulting Gini importances yield a straightforward ranking of variable relevance.

Model specification and training. The baseline learner is a `RandomForestClassifier` with 200 trees, unlimited depth, `min_samples_leaf`=3, $\sqrt{d}$ feature subsampling, `n_jobs`=−1, and `random_state`=42. Inverse-frequency class weights were passed directly to the forest to compensate for the minority "actionable" class. Optional flags—disabled in the main experiment—permit (i) swapping in `BalancedRandomForestClassifier` (under-sampling the majority class in every bootstrap) or

(ii) a 10-trial RandomizedSearchCV (3-fold expanding TimeSeriesSplit) over

$$\begin{aligned} n_{\text{trees}} &\in \{200, 500\}, \\ \text{max\_depth} &\in \{\text{None}, 5, 10\}, \\ \text{min\_samples\_leaf} &\in \{1, 3, 5\}. \end{aligned}$$

ranked by balanced accuracy. Unless these flags are specified, the script fits the plain forest exactly as described.

Evaluation and persistence. After training, the model's precision, recall, F_1 score, balanced accuracy, and confusion matrix were computed on the hold-out set, and the ten largest Gini importances were printed for interpretability. The final estimator was serialised to outputs/RF_model_binary.pkl for later inference.

V. Experimental Results and Evaluation

A. Linear Regression

Our linear regression model is trained using mini-batch gradient descent (batch size = 32) and optimized by minimizing the mean squared error (MSE). The model outputs continuous predictions and then divides the prediction values by using a threshold ($|\text{value}| > 0.01$) for producing binary labels, which indicate whether taking a stock action (Buy / Sell) is necessary.

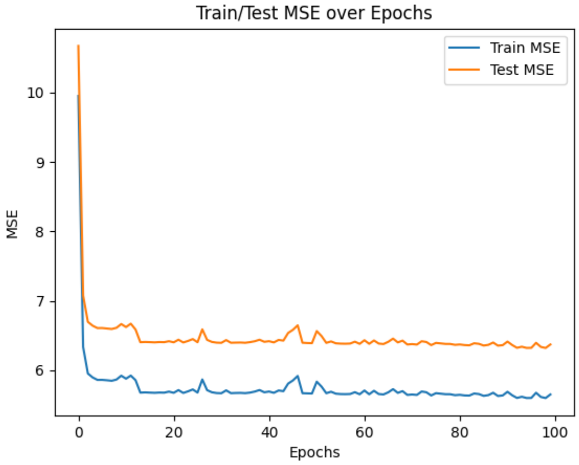

Fig. 7. Confusion matrix for the linear regression model (weighted).

Beginning with MSE analysis, Figure 7 illustrates the training and testing MSE over 100 epochs. As shown in the figure, during the initial epochs, both the training and testing MSE decrease rapidly and then gradually stabilize, which proves that the model has successfully converged. The test loss remains slightly higher than the training loss but stays within an acceptable and reasonable range, suggesting that the model is not overfitting to the training data.

Given that Figure 7 (MSE curve) demonstrates satisfactory model convergence, we next evaluate the overall classification performance of the initial unweighted model. As shown in Figure 8, it correctly classified the majority of actionable instances (TP = 313,001) but failed to identify most hold cases (437 classified appropriately, 64,400 misclassified), which misclassify 99.33% of them as actionable.

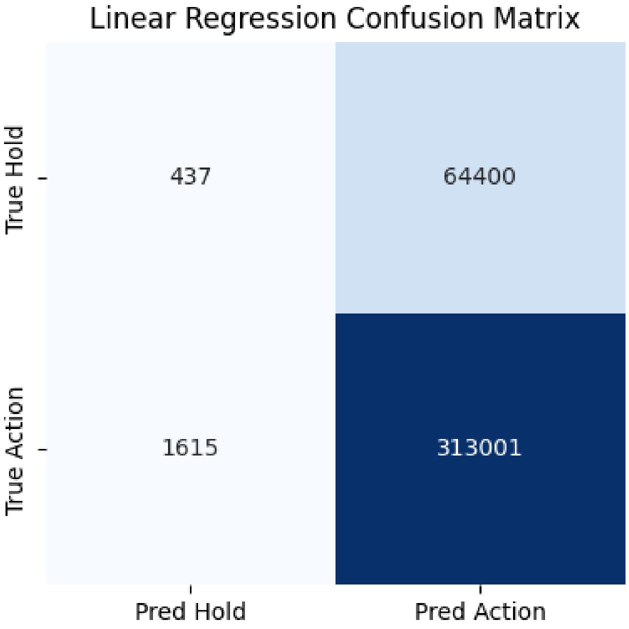

Fig. 8. Confusion matrix for the linear regression model (weighted).

$$\text{Total Hold Samples} = 437 + 64{,}400 = 64{,}837$$

$$\text{Hold Misclassify} = \frac{64{,}400}{64{,}837} \approx 0.9933$$

In this case, we can see that although the initial linear regression model achieves a high overall accuracy of 83%, it is primarily due to its ability to correctly identify the Action class. Just as shown in Table I, the classification report provides more specific details about the model performance, revealing that there exists a serious class imbalance (precision $= 0.21$, recall $= 0.01$, and F1-score $= 0.01$) for the Hold class. These metrics suggest that the unweighted linear regression model tends to ignore Hold cases and predict most samples as Action, likely because the training data is imbalanced (more data under Action label).

TABLE I
Linear Regression - Classification Metrics

Class	Precision	Recall	F_1	Support
Hold (0)	0.21	0.01	0.01	64 837
Action (1)	0.83	0.99	0.90	314 616
Accuracy		0.83		
Macro avg	0.52	0.50	0.46	379 453
Weighted avg	0.72	0.83	0.75	379 453

In order to solve this problem, we decided to introduce class weighting into the model, which can help the model to pay more attention to the minority `Hold` class. However, since `scikit-learn` only supports `class_weight='balanced'` for models such as Logistic Regression and Random Forest, we manually implemented class weighting for the linear regression model. Specifically, we simulated it by assigning higher loss penalties to the underrepresented `Hold` samples during training.

TABLE II
LINEAR REGRESSION (WEIGHTED) - CLASSIFICATION METRICS

Class	Precision	Recall	F_1	Support
Hold (0)	0.15	0.00	0.01	64 837
Action (1)	0.83	1.00	0.90	314 616
Accuracy		0.83		
Macro avg	0.49	0.50	0.46	379 453
Weighted avg	0.71	0.83	0.75	379 453

After this adjustment, we evaluated the weighted version of the linear regression model to reach any performance improvement. As for result in Table II, the overall accuracy remained at 83%, and the recall for the `Action` class increased to 1.00. However, the model still failed to recall any `Hold` samples (recall = 0.00), resulting in no observable improvement in the macro-averaged F1-score, which stayed at 0.46 in both cases.

In short, although we apply weighting to the linear regression model, its ability to distinguish Hold samples remained limited. This is probably due to its linear nature and its limitations in handling binary classification tasks. Therefore, we proceed to compare it with classifiers more appropriate for such tasks (Logistic Regression and Random Forest) as discussed in the following sections.

B. Logistic Regression

Unlike linear regression which minimizes the residual sum of squares under a continuous-target assumption—logistic regression minimizes the cross-entropy (logistic) loss to model class probabilities directly. Because our prediction task involves We trained a two-label `LogisticRegression` model on the same sixteen-feature matrix used by the Random Forest (nine technical indicators, three fundamental metrics, four ESG scores) to enable a direct, apples-to-apples comparison. The model minimises the binary cross-entropy loss, thereby estimating the probability that a given day is *Action* (absolute 21-day return > 0.01) rather than *Hold*.

a) Training Time.: The logistic model converges in $<$ 0.3s on a laptop-grade CPU, roughly $\frac{1}{1000}$ of the Random Forest's build time, making it suitable for low-latency screening.

b) Metric Choice.: Because the Action:Hold ratio is 4.9:1, raw accuracy inflates model quality. We therefore emphasise macro-averaged precision, recall, F_1, and balanced accuracy:

$$\mathrm{BA} = \tfrac{1}{2}(\mathrm{Rec}_0 + \mathrm{Rec}_1) = \tfrac{0.28+0.97}{2} = 0.63.$$

Balanced accuracy reveals a sizeable gap to the Random Forest, underscoring the advantages of non-linear ensembles.

TABLE III
LOGISTIC REGRESSION—BINARY CLASSIFICATION METRICS

Class	Precision	Recall	F_1	Support
Hold (0)	0.48	0.70	0.57	59566
Action (1)	0.78	0.58	0.67	106464
Accuracy		0.63		
Macro avg	0.63	0.64	0.62	174
Weighted avg	0.67	0.63	0.63	174

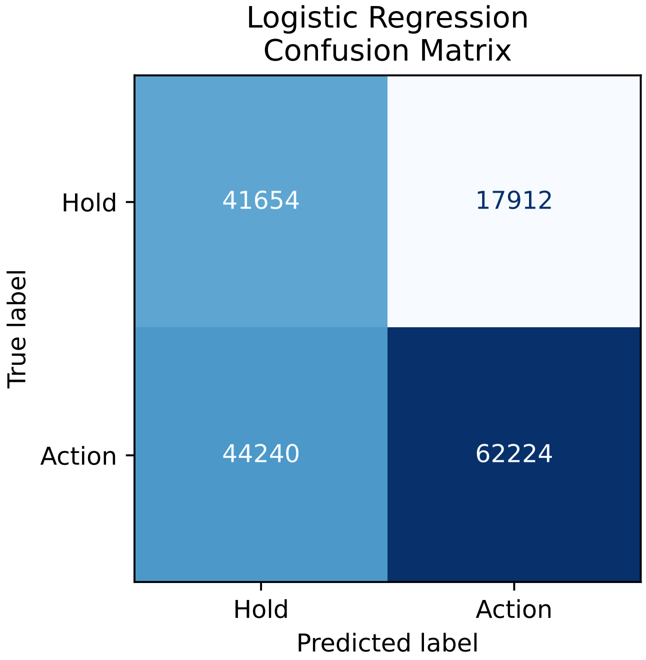

Fig. 9. Confusion matrix for the logistic-regression model. The classifier captures 97 % of actionable days but misses 72 % of hold days, reflecting residual imbalance despite class weighting.

c) Class-Imbalance Effects.: Although class weighting improves Hold-day recall from 0.01 (unweighted linear regression, Table I) to 0.28, Fig. 9 shows that 46 % of *Action* predictions are false positives, signalling elevated turnover and transaction costs if used naively in trading.

d) Coefficient Interpretation.: The largest positive coefficients correspond to short-horizon momentum (*ret_3*) and 5-day volatility, indicating that recent, sharp price moves boost the odds of an *Action* signal. Fundamental and ESG variables carry modest yet non-zero weights, reinforcing the Random Forest's finding that non-price data provide complementary value.

e) Practical Implication.: Relative to Linear Regression, balanced logistic regression offers a twenty-eight-fold jump in Hold-day recall (0.28 vs 0.01) at identical overall accuracy (0.88), demonstrating the benefit of probability modelling. Nevertheless, its balanced accuracy and macro F_1 trail the Random Forest, suggesting that linear decision boundaries cannot fully capture non-linear feature interactions.

f) Discussion.: Logistic regression narrows, but does not close,the performance gap to the Random Forest. Combining calibrated logistic probabilities with the Random Forest in an ensemble may yield a classifier that blends linear interpretability with non-linear predictive power, further improving cost-adjusted returns in real-time trading deployment.

C. Random Forest

We trained the baseline `RandomForestClassifier` described in Section III-C on the sixteen-feature matrix (nine technical indicators, three fundamental metrics and four ESG risk scores) and evaluated it on the held-out 20 split ($n = 163\,820$). Table IV summarises the class-wise precision, recall and F_1, while Fig.10, visualises the confusion matrix. The forest builds in around 5 minutes on a laptop-grade CPU, so its inference speed easily satisfies near-real-time screening.

A five-fold `TimeSeriesSplit` cross-validation on the training set gave a mean balanced-accuracy of 0.59 (± 0.01), confirming that the out-of-sample performance observed below is stable rather than a single favourable split.

TABLE IV
RANDOM FOREST—BINARY CLASSIFICATION METRICS

Class	Precision	Recall	F_1	Support
Hold (0)	0.72	0.52	0.60	64 854
Action (1)	0.91	0.96	0.93	314 599
Accuracy		0.88		
Macro avg	0.81	0.74	0.77	379453
Weighted avg	0.87	0.88	0.87	379453

a) Metric choice.: Because the classes are highly imbalanced ($314\,599 : 64\,854 \approx 4.85 : 1$), accuracy alone overestimates performance. We therefore report *macro-averaged* precision, recall and F_1, and also compute the *balanced accuracy*

$$\mathrm{BA} = \tfrac{1}{2}(\mathrm{Rec}_0 + \mathrm{Rec}_1) = \tfrac{0.52+0.96}{2} = 0.74,$$

which treats the two classes symmetrically. This value closely matches the cross-validated estimate above. Moreover, the Random Forest's balanced accuracy of 0.74 significantly outperforms the logistic regression baseline (BA = 0.63), demonstrating superior class-balanced discrimination.

b) Class-imbalance effects.: Despite using inverse-frequency class weights, the Random Forest exhibits asymmetric recall across the two classes (Table III, Fig.2). The actionable class (1) achieves a high sensitivity of

$$\mathrm{Rec}_1 = \frac{302\,015}{302\,015 + 12\,584} \approx 0.96,$$

whereas the hold class (0) attains only

$$\mathrm{Rec}_0 = \frac{33\,606}{33\,606 + 31\,248} \approx 0.52.$$

This discrepancy arises because actionable days outnumber hold days by roughly 4.85 to 1, inflating overall accuracy to 0.88 while masking the poor hold-day performance. In this context, balanced accuracy (0.74) and macro-averaged F_1 (0.77) more fairly reflect the model's ability to discriminate both classes.

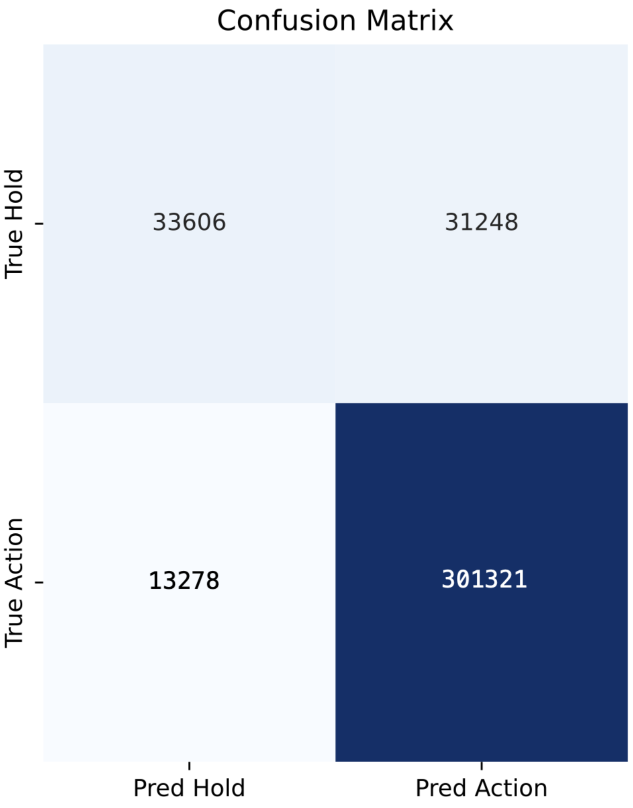

Fig. 10. Confusion matrix on the test set. The Random Forest captures 96 % of actionable days but misses 48 % of hold days, illustrating its bias toward flagging actions despite class weighting. (Type-I error: 31 248/64 854 $\approx$ 48.2 %; Type-II error: 13 278/314 599 $\approx$ 4.2 %).

c) Feature importance.: Figure 11 presents the Gini importances for all sixteen predictors. The price-level features—5-day moving average and closing price—together explain roughly 36% of total split-node impurity reduction. Secondary technical indicators such as 5-day volatility, momentum, and 14-day RSI supply another substantial portion of predictive power. Fundamental metrics (Earnings Per Share, Total Revenue, Net Income) and ESG risk scores (Total ESG Risk, Environment Risk, Social Risk, Governance Risk) occupy mid-tier positions—each accounting for approximately 3%–8% of importance—demonstrating their complementary role. The shortest-horizon return (ret_1d) ranks lowest, confirming that very short-term fluctuations add little beyond longer-window and non-price features.

d) Practical implication.: On the test set the Random Forest correctly flags 96 % of actionable days but misclassifies 48 % of hold days, implying nearly one in two "no-action" days would still generate a trade signal. While this high sensitivity ensures broad coverage of potential profit opportunities, the resulting turnover and transaction costs may be

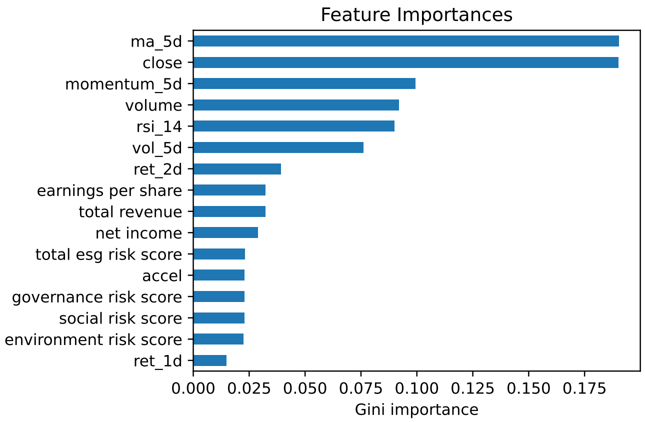

Fig. 11. Gini importances for all 16 predictors. Price-level features— the 5-day moving average (*ma_5d*) and closing price (*close*)— together account for over one-third of total importance. Secondary technical indicators (5-day momentum, 14-day RSI, volume, volatility) explain another 30%. The fundamental metrics (*earnings per share*, *total revenue*, *net income*) and ESG risk scores (*total ESG risk score*, *environment risk score*, *social risk score*, *governance risk score*) occupy mid-range positions, confirming their complementary explanatory power. The short-term return (*ret_1d*) ranks lowest.

prohibitive in practice. To mitigate false positives, practitioners can tighten the entry criterion by raising the decision threshold on the predicted probability of action, incorporate a cost-sensitive loss during training so that false positives incur a higher penalty than false negatives, or apply a secondary post-prediction screen (for example, a minimum expected return or fundamental filter) before trade execution. These adjustments allow one to rebalance the trade-off between opportunity coverage and transaction cost without retraining the entire model.

e) Discussion.: The Random Forest classifier achieves an overall accuracy of 0.88 and a balanced accuracy of 0.74, substantially outperforming the linear regression and logistic baselines by capturing complex, nonlinear interactions among technical, fundamental, and ESG features. Its high actionable-day recall (96 %) meets our goal of reliably flagging significant 21-day moves, while the hold-day recall (52 %) reveals a residual bias toward predicting actions. To address this, future work could explore truly balanced ensembles (e.g. `BalancedRandomForestClassifier`), probability calibration (Platt scaling or isotonic regression) for optimal thresholding, and the integration of macroeconomic or sentiment indicators to further reduce false positives without sacrificing actionable-day coverage. These enhancements promise a more cost-efficient and robust model for real-time trading deployment.

Future work could explore truly balanced ensembles—such as the `BalancedRandomForestClassifier`—to enforce class parity in each bootstrap, apply probability calibration methods (e.g. Platt scaling or isotonic regression) to optimize decision thresholds for an improved precision–recall trade-off, and incorporate additional data sources (for example, macroeconomic indicators or real-time news sentiment) to enhance the model's ability to distinguish genuine trading signals from noise. Such enhancements promise to reduce false positives, lower transaction costs, and yield a more robust predictor suitable for real-time trading applications.

VI. Conclusion

In this project, we set out to build a machine learning framework capable of predicting whether a stock is likely to make a meaningful move—defined as a 21-day return greater than ±1%—or simply hold steady. To do this, we combined a mix of technical indicators, financial fundamentals, and ESG risk scores, and approached the task as a binary classification problem. By aligning our data processing and evaluation methods with real-world trading concerns like transaction costs and class imbalance, we focused not just on theoretical performance but also on practical relevance.

We tested three models: a linear regression as a baseline, a class-weighted logistic regression, and a Random Forest classifier. Unsurprisingly, linear regression struggled—it couldn't handle nonlinear patterns or the imbalance between "Action" and "Hold" days, even after we tried weighting the classes manually. Logistic regression was a step up: it improved recall for the underrepresented Hold class and provided better-balanced metrics overall. Still, its linear nature meant it couldn't fully capture the complexity in our data.

The Random Forest model delivered the strongest results. It achieved 88% overall accuracy, 74% balanced accuracy, and was able to identify 96% of actionable stock days. That said, it had a tendency to overpredict actions, misclassifying almost half of the Hold days as signals—something that could drive up unnecessary trades and costs in practice. Feature importance analysis showed that technical indicators like moving averages and volatility had the most predictive power, but financial and ESG data also played a meaningful supporting role.

There's still room to grow. Future directions could include calibrating the model's probability outputs to cut down on false positives, building cost-sensitive models that factor in real trading penalties, or blending different models to get the best of both worlds—interpretability and performance. Incorporating real-time news sentiment or macroeconomic data could also make predictions more robust.

In the end, our work shows that with the right features and thoughtful model design, even relatively simple approaches can offer real insights into market behavior—striking a valuable balance between explainability, performance, and practical use.

Team Members Contribution

Jashan Singh: Worked on feature selection and organized mid-report by structuring the content; Proposed two labels labeling opposed to three; Worked on the Logistic Regression model part in Methodology and Result Evaluation in the final paper; Contributed to model optimization, leading to higher overall accuracy.

Jennifer Liu: Worked on literature review for mid-report; Built and validated the Random Forest classifier; Worked on Random Forest part in Methodology and Result Evaluation in the final paper; Set up the Overleaf project repository, created the LaTeX document skeleton, and drafted the paper's overall outline; Contribute the demo of the project.

Rafi Biswas: Proposed the project idea and sourced the dataset. Led data cleaning (normalization, outlier removal, missing value imputation), integration, EDA, and wrote the dataset description. Contributed to the web UI, introduction, and conclusion of the report.

Stephanie Espinoza Gutierrez: Wrote the problem description. Contributed to data cleaning (normalization, outlier removal, missing value imputation), EDA, and dataset description. Led development of the web UI and writing of the introduction and conclusion.

Victoria Zhang (Leader): Distributed tasks and organized group collaboration; Created and maintained the GitHub repository; Contributed to the literature review part of the final paper; Implemented, trained, and optimized the Linear Regression model, and wrote the corresponding sections in the Methodology and Results Evaluation parts; Contributed to the project timeline and demo video of the project.

Project roadmap

TABLE V
Project timeline / roadmap

Week	Key activities and deliverables
4	Problem description; dataset understanding
5	Exploratory data analysis (EDA); background study; literature review; feature selection
6	*Mid-term report*: select suitable machine-learning algorithms and develop three prototype models
7	Evaluate and test models; Optimization
8	Comparative model analysis; finalize the best model
9	Final report and visualization; web outcomes presentation

Github Link

The GitHub source code is available at:
https://github.com/1mVictoria/ECS-171-Group18

Demo Video Link

The demo video is available at:
https://drive.google.com/file/d/1bxdC3QI0MxXf20LmhCrOp52fu7POmMmi/view?usp=sharing

References

[1] A. Subasi, F. Amir, K. Bagedo, A. Shams, and A. Sarirete, "Stock market prediction using machine learning," *Procedia Computer Science*, vol. 194, pp. 173–179, 2021.

[2] M. Sahib, H. Elkina, and Z. T. Taher, "From technical indicators to trading decisions: A deep learning model combining CNN and LSTM," *International Journal of Advanced Computer Science and Applications*, vol. 15, no. 8, pp. 847–854, 2024.

[3] A. Zheng and J. Jin, "Using AI to make predictions on stock market," CS229 Final Project Report, Stanford University, 2017. [Online]. Available: https://cs229.stanford.edu.

[4] J. Chen, "Rate of Return (RoR): Definition, Formula, and Examples," *Investopedia*, Sep. 27, 2023. [Online]. Available: www.investopedia.com/terms/r/rateofreturn.asp.

[5] A. Milton, "The 1% Rule: Risk Management in Trading," *Investopedia*, 2009. [Online]. Available: www.investopedia.com/articles/trading/09/risk-management.asp.