Comparative Analysis of Individual and Ensemble Machine Learning Models for Classifying Time-Series Stock Movement Direction

Farzan Mirza

Nakul Narang

Drexel University

Abstract

While recent research in stock market prediction has predominantly centered on Deep-Learning models like LSTMs, or time-series models like ARIMA, this study diverges by focusing on intra-day intervals of 5 minutes to uncover market behavior nuances. Our approach contrasts with these traditional methods by implementing a uniform modeling strategy using Multiple Linear Regression, Logistic Regression, Linear Discriminant Analysis (LDA), Naive Bayes, and Decision Tree Learning. This choice is driven by the potential of these models to provide clear interpretability and computational efficiency, crucial for the dynamic nature of financial markets. Analyzing data from 66 stocks across all 11 sectors of the SP 500, our strategy aims to create versatile models suitable for unseen stocks, moving away from the common practice of stock-specific modeling. We further employ ensemble learning techniques, integrating weighted voting systems and RandomForest Classifiers. Comparative analysis using classification metrics shows that in our setup, traditional linear models, especially LDA, outperform the more complex ensemble methods. This finding underscores the significance of choosing the right model based on the data characteristics, emphasizing the impact of temporal resolution and the importance of training models collectively on a diversified stock dataset. Our research not only challenges the current trend of employing advanced, often opaque models but also brings to light the efficacy of simpler, interpretable models in the high-frequency trading domain

Keywords: Intra-Day Stock Prediction, Machine Learning Models, Ensemble Learning Techniques, Financial Market Analysis, Model Interpretability, High-Frequency Trading

Comparative Analysis of Individual and Ensemble Machine Learning Models for Classifying Time-Series Stock Movement Direction

Background and Related Work

The endeavor to model stock market behavior is a complex and multifaceted research area, driven by the diversity and magnitude of influencing variables. Traditional methodologies have varied, encompassing statistical time-series models like ARIMA, as well as sophisticated machine learning approaches, including Support Vector Machines (SVMs) and Neural Networks (NNs). The literature reveals significant advancements in predictive accuracy; Zhang et al. (2023) and Sonkavde et al. (2023) have reported high precision in time-series analysis through the application of NNs and ARIMA models, respectively. Ensemble learning classifiers have also been recognized for their predictive potential, as underscored by Sonkavde et al. (2023).

The use of continuous Hidden Markov Models (HMMs) for inter-day stock movement prediction, investigated by Gupta & Dhingra (2012), has also garnered attention for its acknowledgment of the non-linear and temporal dimensions of market data. Traditional machine learning models, including linear regression and linear kernel SVMs, retain their prominence in the field, exhibiting commendable accuracy in specific scenarios, as noted by Wang et al. (2023) and Panwar et al. (2021). Notably, ensemble techniques that integrate SVMs, NNs, and Decision Trees with boosting methods have been particularly effective, achieving over 90% accuracy in certain studies (Nti et al., 2020).

Notwithstanding these developments, existing methodologies typically concentrate on a restricted set of stocks, pinpointing the potential for models that can generalize across a more extensive range of securities. Complex models, while capable of processing large datasets, often grapple with interpretability and are susceptible to extrapolation inaccuracies. A discernible void in the literature pertains to the exploration of intra-day stock market data, which yields a more detailed comprehension than the conventionally analyzed inter-day data. Furthermore, the prevalent research trend is to develop models tailored to specific stocks, neglecting the interdependence

that characterizes the market, especially within indices like the S&P 500.

Our research is predicated upon these observations, aiming to refine model scope and enhance the granularity of data analysis. Drawing inspiration from Sonkavde et al. (2023), we explore the potential of ensemble methods in conjunction with straightforward classification models, applied to intra-day temporal data across a broad spectrum of stocks. Our objective is to construct versatile models that perform reliably on unfamiliar stocks, thereby transcending the typical stock-specific modeling approach. By amalgamating the insights from prior research with pioneering ensemble techniques, our study endeavors to rectify existing research gaps, striving to improve forecast precision and relevance in the dynamic landscape of stock market prediction

Methodology

Our methodology encompasses a comprehensive three-pronged approach to accurately predict stock market movements. This approach is broken down into data preprocessing, model training and ensemble prediction.

Data Preprocessing

For our study, we curated a dataset reflective of the broad market by selecting stocks from each of the 11 Global Industry Classification Standard (GICS®) categories within the S&P 500. To facilitate generalization, two stocks from each category were allocated for model training and testing, while an additional four stocks per category were reserved for validation as unseen data, culminating in a dataset comprising 66 stocks. This strategy ensured a diverse representation across different sub-industries, company sizes, and trading volumes.

Data Collection. We compiled time-series data in 5-minute intervals for each stock, selectively choosing random trading days from the most recent two-month period to enhance data variability and reduce chronological bias.

Data Source and Limitations. The ‘yfinance’ Python module within a Conda environment was employed to fetch intra-day interval data, limited to the preceding two months. This restriction led to a dataset of 35 days for training/testing and 5 days for unseen data, with each day providing 78 timesteps (9:30 am to 4:00 pm). A snapshot of

the initial raw dataset is provided in Appendix A (see Figure A1). For our analysis we are using data obtained from 11 October 2023 to 11 December 2023.

Data Structure and Features. Each time-step delineated a distinct 5-minute interval, characterized by features such as Open, High, Low, Close, Adjusted Close, and Volume. A 'window_ID' identifier was generated for each combination of stock and start date to interlink the 78 intervals comprising a day. Additionally, a 'timestep' column was introduced for sequential indexing, and a 'volatility' metric was derived from the High and Low values to capture price fluctuations within each interval.

Target Variable and Data Cleaning. A binary label was designated as the target variable, assigned to forecast whether the subsequent time-step's Close would surpass the current Open. To create this target variable, the Close value was shifted one time-step backward, allowing us to compare it with the Open value of the current time-step. Consequently, the 78th time-step of each window lacked a corresponding future Close value to serve as a target, leading us to exclude it from the analysis. Therefore, each window in our dataset was effectively reduced to 77 time-steps. This adjustment was critical to ensure there was no lookahead bias, maintaining the relevance and accuracy of our predictive models. Non-essential columns including Datetime, High, Low, Adjusted Close, and Close were also removed as part of the data cleaning process. The resultant dataset is shown in Appendix A (see Figure A2)

Dataset Preparation. The data were partitioned into training and testing sets with careful manual selection, ensuring 30 unique windows per stock were apportioned for training, with the remainder allocated for testing. The unseen dataset comprised 5 windows for each additional stock. To foster heterogeneity, all datasets were shuffled based on 'window_ID'.

Normalization and Final Dataset. Critical features—open price, volatility, and volume—underwent normalization to reconcile scale disparities amongst different stocks. The 'stock' and 'start_date' fields were omitted, with 'window_ID' serving as the sole differentiator, thus rendering the model agnostic to specific stocks or dates. The processed datasets, prefixed with 'ml_', constituted the finalized training, testing,

and unseen datasets deployed for model development and evaluation. The normalized dataset is shown in Appendix A (see Figure A3).

Model Training

In this section, we present the supervised learning-based classification models employed in our study, each tailored to address the classification problem posed by our research. The selected models include Multiple Linear Regression, Logistic Regression, Linear Discriminant Analysis, Naive Bayes, and Decision Trees. Each of these models utilizes the training, testing, and unseen datasets as input parameters, producing both actual and predicted labels for the testing and unseen datasets. Additionally, we calculate the training accuracy, a crucial metric for our ensemble learning methods, which utilize a weighted mean for voting

Multiple Linear Regression. Implemented to discern linear trends in stock market data, multiple linear regression serves as a key classifier in our study. This approach, essential for capturing the linear relationships between variables, aligns with methodologies prevalent in stock price prediction research (Wang et al., 2023). The MLR model can be represented as

$$y = Xw + b + \varepsilon$$

where y is the predicted output, X is the feature matrix, w is the weight vector, b is the bias term, and ε represents the error term. The model employs gradient descent optimization, starting with zero-initialized weights ($w_0 = 0$). The weights are iteratively updated using the gradient descent algorithm:

$$w_{i+1} = w_i - \eta \cdot \nabla J(w_i)$$

where w_i is the weight vector at iteration i, η (learning rate) is set to 0.001, and $\nabla J(w_i)$ is the gradient of the cost function J with respect to the weights. We performed 5,000 epochs to achieve optimal convergence.

For the purpose of classification, a threshold value of 0 was established for the model's output. Predictions at or above this threshold indicate a rise in stock prices (classified as 1), while those below signify a fall (classified as 0). This binary

classification mechanism was fine-tuned through rigorous testing to ensure accuracy in reflecting stock price movements.

Logistic Regression. Logistic regression, a fundamental model in our analysis of stock market data, excels in binary classification tasks. It is renowned for its ability to model the probability of binary outcomes, aligning with established methods in stock price prediction research (Wang et al., 2023).

At the core of logistic regression is the sigmoid function, denoted as $\sigma(z)$, which maps any input z to a value between 0 and 1, representing the probability of the binary outcome. The sigmoid function is defined as:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

This model also employs gradient descent optimization, starting with zero-initialized weights ($w_0 = 0$). The weights are iteratively updated using the gradient descent algorithm:

$$w_{i+1} = w_i - \eta \cdot \nabla J(w_i)$$

where w_i is the weight vector at iteration i, η (learning rate) is set to 0.001, and $\nabla J(w_i)$ is the gradient of the cost function J with respect to the weights. We performed 5,000 epochs to achieve optimal convergence.

For our classification task, we set a decision threshold at 0.5. Predictions with a probability equal to or above this threshold are classified as indicating a rise in stock price (1), while predictions below the threshold indicate a fall (0). This threshold is meticulously established and validated to ensure the accurate classification of stock price movements. As a result, the logistic regression model effectively distinguishes between the two possible outcomes, offering both a principled mathematical foundation and practical effectiveness in classifying stock price movements.

Linear Discriminant Analysis (LDA). Employed as a sophisticated technique for distinguishing between two classes in stock market data. LDA is renowned for its efficacy in finding a linear combination of features that best separates distinct classes, making it a suitable choice for predicting binary outcomes in stock price

movements. In implementing LDA, the model first computes the mean vectors (μ_k) for each class (k) in the training data:

$$\mu_k = \frac{1}{n_k} \sum_{i=1}^{n_k} x_i^{(k)}$$

Where n_k is the number of samples in class k, and $x_i^{(k)}$ is a sample from class k. LDA then calculates the covariance matrices (Σ_k) for these classes, essential for understanding the spread of the data around the mean:

$$\Sigma_k = \frac{1}{n_k - 1} \sum_{i=1}^{n_k} (x_i^{(k)} - \mu_k)(x_i^{(k)} - \mu_k)^T$$

The between-class scatter matrix (S_b) and within-class scatter matrix (S_w) are derived next, representing the separation between the classes and the variance within each class, respectively:

$$S_b = \sum_{k=1}^{c} n_k (\mu_k - \mu)(\mu_k - \mu)^T$$

$$S_w = \sum_{k=1}^{c} \sum_{i=1}^{n_k} (x_i^{(k)} - \mu_k)(x_i^{(k)} - \mu_k)^T$$

Where c is the number of classes, μ is the overall mean, and n_k is the number of samples in class k. The core of the LDA model lies in the computation of the matrix (S_m), which is the product of the inverse of the within-class scatter matrix and the between-class scatter matrix:

$$S_m = S_w^{-1} S_b$$

Through eigenvalue decomposition of this matrix, we extract the eigenvectors (W) and eigenvalues (λ). The principal eigenvector (w), corresponding to the largest eigenvalue, is then used to project the data onto a new axis:

$$y = XW$$

Effectively reducing the dimensionality while preserving class separability.

The LDA model classifies new data by projecting it onto this axis and comparing the distances of the projected points to the class means, calculated post-projection. The predicted class is determined by the proximity of the data point to these class means. This process is applied to both the training and test datasets, as well as the unseen data.

Naive Bayes Classifier. In our analysis, the Naive Bayes classifier is adeptly adapted for probabilistic classification, particularly for binary outcome predictions in stock market data, including handling continuous features. This model is favored for its simplicity and computational efficiency, key attributes for managing complex datasets.

Central to the Naive Bayes methodology is the assumption of feature independence given the class label. While this assumption simplifies the model, it remains effective in yielding robust predictions, a critical aspect in stock price movement forecasting. Our adaptation of the Naive Bayes classifier for continuous features encompasses several integral steps:

1. **Class Priors Calculation ($P(y)$):** The model commences by computing the prior probabilities of each class, offering a baseline likelihood of each class's occurrence in the dataset. This is calculated as:

$$P(y) = \frac{\text{Number of samples in class } y}{\text{Total number of samples}}$$

2. **Likelihoods Computation for Continuous Features ($P(x|y)$):** Recognizing the prevalence of continuous features in financial data, such as price movements and trading volumes, our implementation tailors the calculation of likelihoods. For each class, the model estimates means (μ) and variances (σ^2) of the continuous features. These statistical measures are then utilized to derive Gaussian probability densities for each feature, providing the conditional likelihoods essential in our probabilistic framework. The Gaussian probability density for feature x in class y is calculated as:

$$P(x|y) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

3. **Classification Process for $P(y|x)$:** In classifying data points, the model calculates the posterior probabilities for each class by integrating the log-transformed class priors with the computed likelihoods. This process, crucial for continuous data, involves a careful balance between numerical stability and accuracy in probability estimation. The class with the highest posterior probability is identified as the predicted outcome. The posterior probability for class y given features x is calculated as:

$$P(y|x) \propto P(y) \prod_{i=1}^{n} P(x_i|y)$$

Where x_i represents individual features.

By adapting the Naive Bayes classifier to effectively handle continuous features within stock market data, our model gains a significant edge in accurately discerning market trends and patterns. This adaptation not only enhances the classifier's predictive capabilities but also ensures a comprehensive understanding of the dynamics in financial markets.

Decision Tree Classifier. The Decision Tree classifier in our analysis is specifically designed to model complex decision-making processes, and it's particularly effective for handling the intricate patterns found in stock market data. This model stands out for its interpretability and capability to manage non-linear relationships between features, including continuous data.

1. **Tree Construction:** We construct the decision tree by recursively partitioning the training data. At each node, the algorithm determines the best feature to split on, based on information gain. This process continues until the maximum specified depth is reached or no further meaningful splits can be made.

2. **Handling Continuous Features:** For continuous features typical in financial data, such as stock prices and volumes, our implementation uses a sophisticated approach. We calculate Gaussian probability density functions for each feature to

assess the likelihood of the data belonging to each class. This method allows the decision tree to make informed splits on continuous data, capturing more nuanced relationships than traditional binary splits.

3. **Feature Selection and Entropy Reduction:** The model evaluates each feature's contribution to class separability, using entropy (H) as a measure. Entropy measures the impurity or randomness in a set of data points. The entropy of a set S is calculated as:

$$H(S) = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

Where p_i is the proportion of data points in class i in set S.

In our refined approach, continuous features are assessed for their potential to reduce uncertainty most effectively, thereby guiding the selection of the optimal split at each node.

4. **Classification Decision:** The leaf nodes represent the final classification decisions, where each node is assigned a class based on the majority class of the data points that reach it. Our model's treatment of continuous features ensures that these decisions are based on a thorough analysis of underlying trends and patterns.

5. **Depth Control and Overfitting Prevention:** To avert overfitting, which is particularly crucial for continuous data with many unique values, we control the tree's growth through a maximum depth parameter. This constraint ensures that the model remains generalizable and doesn't over-interpret the noise in the data.

6. **Predictive Modeling:** For prediction, data points traverse the tree to a leaf node where a classification decision is made. This approach is applied to the test and unseen datasets, assessing the model's generalizability and effectiveness in diverse scenarios.

By tailoring the Decision Tree classifier to adeptly handle continuous features, our model is enhanced for the complex and volatile nature of stock market data. This adaptation not only improves its predictive accuracy but also ensures that the model remains interpretable and grounded in the realities of financial trend analysis.

Ensemble Learning

Ensemble learning is a pivotal aspect of our computational approach to stock market prediction. This technique involves the strategic combination of multiple machine learning models to enhance the overall predictive power and accuracy and mitigate the weaknesses inherent in individual models. This section outlines our methodologies, starting with a simple ensemble that incorporates all our classifiers including Linear Regression, Logistic Regression, Linear Discriminant Analysis, Naive Bayes, and Decision Trees. We then refine our approach by focusing on a subset of the best-performing classifiers, optimizing the balance between complexity and performance. Finally, we introduce a sophisticated Random Forest model, an advanced ensemble of Decision Trees, specifically designed to tackle the complexities of high-dimensional stock market data. Each of these models utilizes the training, testing, and unseen datasets as input parameters, producing both actual and predicted labels for the testing and unseen datasets

Simple Ensemble Learning with All Classifiers. In our analysis, we employ an ensemble learning approach that combines the strengths of all our classifiers, including Linear Regression (LinReg), Logistic Regression (LogReg), Linear Discriminant Analysis (LDA), Naive Bayes (NB), and Decision Trees (DT). This ensemble is designed to effectively handle the complexity and variability of stock market data, leveraging the unique characteristics of each individual classifier.

1. **Classifier Selection:** Our ensemble consists of five distinct classifiers, each chosen for its specific attributes. These classifiers are LinReg, LogReg, LDA, NB, and DT, and they have been tailored to address different aspects of stock price prediction.

2. **Weighted Mean Ensemble:** To combine the predictions of these classifiers, we utilize a weighted mean approach. The idea behind this ensemble method is to assign different weights to each classifier based on its training accuracy. Higher training accuracies are given more weight, indicating greater confidence in the classifier's performance.

3. **Weight Normalization:** To ensure that the weights sum up to one, we normalize the training accuracies of all classifiers. This normalization process makes the weights proportionate to the classifiers' abilities to capture underlying patterns in the data.

4. **Weighted Voting:** With the normalized weights in place, we calculate a weighted vote for each prediction. For a given data point, the classifiers' predictions are combined into a final prediction using the weighted mean formula:

$$\text{Weighted Mean} = \frac{\sum_{i=1}^{N} \text{Prediction}_i \times \text{Weight}_i}{\sum_{i=1}^{N} \text{Weight}_i}$$

   Here, N represents the number of classifiers, Prediction_i is the prediction of the i-th classifier, and Weight_i is the normalized weight of the i-th classifier.

5. **Classification Results:** The final predictions obtained through this ensemble approach are then used for both testing and unseen datasets. By combining the predictions of multiple classifiers with weighted votes, we aim to improve predictive accuracy and robustness, effectively leveraging the diverse strengths of each classifier in our ensemble.

This ensemble method ensures that our model adapts to the unique characteristics of stock market data while benefiting from the collective insights of multiple classifiers. It is a powerful technique for enhancing the reliability and performance of stock price movement prediction.

Simple Ensemble Learning with Best Classifiers. In this refined approach, we concentrate our ensemble learning model on a select group of classifiers, specifically

chosen for their superior performance in our initial evaluations. This model includes Linear Regression (LinReg), Linear Discriminant Analysis (LDA), and Decision Trees (DT). By focusing on these high-performing classifiers, we aim to enhance the predictive accuracy and reliability of our model, especially in the context of stock market data.

1. **Classifier Selection:** The ensemble is streamlined to include only LinReg, LDA, and DT. These classifiers have demonstrated higher efficacy in handling the complexities and nuances of stock market prediction.

2. **Weighted Mean Ensemble:** Similar to our initial model, this refined ensemble also employs a weighted mean approach. The weights are assigned based on the training accuracy of each classifier, reflecting our confidence in their predictive capabilities.

3. **Weight Normalization:** The training accuracies are normalized to ensure that the weights sum up to one. This step maintains the balance between the classifiers, allowing each to contribute proportionately to the final prediction.

4. **Weighted Voting:** With the normalized weights in place, we calculate a weighted vote for each prediction. For a given data point, the classifiers' predictions are combined into a final prediction using the weighted mean formula:

$$\text{Weighted Mean} = \frac{\sum_{i=1}^{N} \text{Prediction}_i \times \text{Weight}_i}{\sum_{i=1}^{N} \text{Weight}_i}$$

   Here, N represents the number of classifiers, Prediction_i is the prediction of the i-th classifier, and Weight_i is the normalized weight of the i-th classifier.

5. **Classification Results:** The final ensemble predictions are generated for both testing and unseen datasets. By narrowing down the classifiers to the most effective ones and combining their predictions, we aim to achieve higher precision and robustness in forecasting stock price movements.

This refined ensemble learning model is designed to leverage the strengths of the best-performing classifiers, providing a more focused and potentially more accurate

approach to predicting stock market trends. By concentrating on a select group of algorithms, we aim to optimize our model for enhanced performance and reliability in stock price movement prediction.

Random Forest Ensemble Classifier. Our Random Forest model is an advanced ensemble classifier that leverages multiple Decision Trees to improve the robustness and accuracy of stock market predictions. This method is particularly effective in handling complex, high-dimensional data sets.

1. **Initialization:** The Random Forest is initialized with a specified maximum depth for each tree and a number of trees. These parameters control the complexity and size of the forest.

2. **Tree Construction:** The model constructs each tree by randomly selecting subsets of the training data and features. This random selection introduces diversity among the trees, enhancing the model's ability to generalize.

3. **Best Split Determination:** Each tree in the forest determines the best split based on information gain, considering a random subset of features at each node. This is calculated using the entropy formula:

$$\text{Entropy} = -\sum_{i=1}^{C} p_i \log_2(p_i)$$

   where p_i is the probability of class i at a node.

4. **Leaf Node Creation:** A leaf node is created if all samples at a node belong to the same class or if the maximum depth of the tree is reached. The class assigned to a leaf node is the one with the highest frequency among the samples.

5. **Classification:** For a given input sample, each tree in the forest makes a prediction. The final classification is determined by majority voting among all trees. This process reduces overfitting and improves prediction reliability.

6. **Data Division:** The training data is divided into several subsets, each of which is used to train a separate Decision Tree. This method ensures that each tree learns from a different portion of the data, contributing to the diversity of the ensemble.

Our Random Forest model is a powerful ensemble technique that combines the simplicity of Decision Trees with the flexibility of randomness. By aggregating the predictions of multiple trees, it significantly improves over the performance of a single tree, particularly in terms of reducing overfitting and handling varied and complex data structures like those encountered in stock market analysis.

Analysis and Conclusions

To assess the efficacy of our classifiers, we utilized the 5 common classification metrics

- **Accuracy (%):** The percentage of true results among the total cases examined.
- **Precision (%):** The proportion of true positives out of all positive results.
- **Recall (%):** The proportion of true positives identified out of all actual positives.
- **F1 Score (%):** The harmonic mean of precision and recall, providing a balanced measure.
- **ROC-AUC:** The area under the receiver operating characteristic curve, illustrating the true positive rate against the false positive rate.

Analysis

	metric	dataset	linreg	logreg	lda	nb	dtl	el_all	el_lin	el_rf
0	Accuracy %	Test	66.35	50.14	65.56	54.75	49.86	54.88	66.29	50.14
1	Accuracy %	Unseen	65.80	50.24	65.77	59.15	49.75	59.10	65.17	50.25
2	Accuracy %	Average	66.08	50.19	65.66	56.95	49.81	56.99	65.73	50.19
3	Precision %	Test	64.99	50.00	63.88	52.55	49.86	52.57	62.31	100.00
4	Precision %	Unseen	63.15	50.00	63.24	55.81	49.75	55.37	61.16	100.00
5	Precision %	Average	64.07	50.00	63.56	54.18	49.81	53.97	61.74	100.00
6	Recall %	Test	70.49	99.93	71.16	95.12	100.00	97.16	82.00	0.00
7	Recall %	Unseen	75.05	99.37	74.51	85.93	100.00	91.69	82.19	0.00
8	Recall %	Average	72.77	99.65	72.83	90.53	100.00	94.43	82.10	0.00
9	F1 %	Test	67.63	66.65	67.32	67.70	66.54	68.22	70.81	0.00
10	F1 %	Unseen	68.59	66.52	68.41	67.67	66.45	69.05	70.14	0.00
11	F1 %	Average	68.11	66.59	67.87	67.69	66.49	68.64	70.47	0.00
12	ROC-AUC	Test	0.70	0.50	0.70	0.50	0.50	0.50	0.70	0.50
13	ROC-AUC	Unseen	0.70	0.50	0.70	0.60	0.50	0.60	0.70	0.50
14	ROC-AUC	Average	0.70	0.50	0.70	0.60	0.50	0.60	0.70	0.50

Figure 1. Classification metrics results for the evaluated classifiers.

The results are segmented into performance on the test dataset, unseen dataset, and their average, providing a comprehensive view of each model's capabilities. The classifiers include Linear Regression (linreg), Logistic Regression (logreg), Linear Discriminant Analysis (lda), Naive Bayes (nb), Decision Tree Learning (dtl), Ensemble with All Classifiers (el_all), Ensemble with Best Classifiers (el_lin), and Ensemble with Random Forest (el_rf).

Linear Regression (linreg). Linear Regression displays an Accuracy of 66.35% on the Test dataset and 65.80% on the Unseen dataset. The slight decrease in Accuracy for unseen data suggests the model's generalization ability is reasonable, and the data may exhibit linear qualities that Linear Regression captures effectively.

Logistic Regression (logreg). Logistic Regression has a lower Accuracy, at 50.14% on the Test dataset and 50.24% on the Unseen dataset. The consistently lower Accuracy across datasets indicates that the model struggles with the complexity of the data, which may not be well-represented by a logistic function.

Linear Discriminant Analysis (lda). LDA reports an Accuracy of 65.56% on the Test dataset and 65.77% on the Unseen dataset, which is comparable to that of Linear Regression. This suggests that the assumptions of LDA, which include linear separability, hold to some extent in the data.

Naive Bayes (nb). Naive Bayes shows an Accuracy of 54.75% on the Test dataset and 59.15% on the Unseen dataset. Although it has a lower Accuracy than Linear Regression and LDA, its better performance on unseen data indicates some robustness in generalization despite the model's simplicity.

Decision Tree Learning (dtl). Decision Tree Learning achieves a perfect Recall of 100% on both Test and Unseen datasets, which is exceptional. However, its lower Accuracy, 49.86% on the Test and 49.75% on the Unseen dataset, suggests that while it captures all positive instances, it also makes numerous incorrect positive predictions, leading to a lower Precision.

Ensemble with All Classifiers (el_all). The ensemble method combining all classifiers (el_all) achieves a balanced F1 Score of 68.22% on the Test dataset and 69.05% on the Unseen dataset. This demonstrates that the ensemble method effectively integrates the strengths of the individual classifiers, leading to improved overall predictions.

Ensemble with Best Classifiers (el_lin). The ensemble with selected classifiers (el_lin) shows an impressive Precision of 100% on both Test and Unseen datasets. However, it has a Recall of 82.00% on the Test dataset and 82.19% on the Unseen dataset, which indicates a very selective model that avoids false positives but at the expense of missing some true positives.

Ensemble with Random Forest (el_rf). Random Forest's ensemble (el_rf) has an Accuracy equal to Linear Regression on the Test dataset, at 50.14%, but its

Recall of 0% is concerning. This suggests severe overfitting, as the model fails to identify any true positives on the Test dataset, which is not reflected in its Accuracy on Unseen data.

Conclusions

The results suggest that the dataset exhibits both linear and non-linear characteristics. Linear Regression and LDA indicate the presence of linear separability to some extent. However, the high recall of the Decision Tree Learning model suggests that non-linear patterns are also significant and that a model capturing such patterns can perform well on both seen and unseen data.

The el_lin ensemble's high precision across unseen data indicates a strong predictive model where the cost of false positives is high. However, the ensemble's low recall rate on the test dataset calls for further investigation into its ability to detect true positives.

Random Forest's low recall suggests an overfitting to the training data and necessitates a review of the model's complexity and training process. Future work should aim to create models that are complex enough to understand the intricacies of the data but also generalize well to new, unseen data.

Future Work

We believe that a better model can be obtained by making certain changes. In our experiments we discovered that the underlying data is linearly separable due to good performance of multiple linear regression and LDA. An interesting approach to taking advantage of this can be to use linear SVMs. From our results we also infer that the decision trees might have overfitted to the dataset hence an improvement can be to experiment with the hyperparameters of the decision tree. A tweak that can be made on the data side to improve the performance is the inclusion of decades old data. A constraint we faced using yfinance package was collection of older data. However, this can be overcome with google finance, and therefore in our estimate the performance of the model can improve further and become more robust because our concept of temporal windows will apply perfectly well to this situation.

References

- Sonkavde, G., Dharrao, D. S., Bongale, A. M., Deokate, S. T., Doreswamy, D., & Bhat, S. K. (2023). Forecasting stock market prices using machine learning and Deep Learning Models: A systematic review, performance analysis and discussion of implications. *International Journal of Financial Studies*, 11(3), 94. https://doi.org/10.3390/ijfs11030094

- Wang, Q., Xu, C., & Zhou, T. (2023). Stock price prediction based on multiple linear regression. *BCP Business & Management*, 36, 48–54. https://doi.org/10.54691/bcpbm.v36i.3384

- Antad, S., Khandelwal, S., Khandelwal, A., Khandare, R., Khandave, P., Khangar, D., & Khanke, R. (2023). Stock price prediction website using linear regression - A machine learning algorithm. *ITM Web of Conferences*, 56, 05016. https://doi.org/10.1051/itmconf/20235605016

- Zhang, L., Wang, R., Li, Z., Li, J., Ge, Y., Wa, S., Huang, S., & Lv, C. (2023). Time-series neural network: A high-accuracy time-series forecasting method based on kernel filter and time attention. *Information*, 14(9), 500. https://doi.org/10.3390/info14090500

- Sonkavde, G., Dharrao, D. S., Bongale, A. M., Deokate, S. T., Doreswamy, D., & Bhat, S. K. (2023). Forecasting stock market prices using machine learning and Deep Learning Models: A systematic review, performance analysis and discussion of implications. *International Journal of Financial Studies*, 11(3), 94. https://doi.org/10.3390/ijfs11030094

- Panwar, B., Dhuriya, G., Johri, P., Singh Yadav, S., & Gaur, N. (2021). Stock market prediction using linear regression and SVM. *2021 International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE).* https://doi.org/10.1109/icacite51222.2021.9404733

- Nti, I. K., Adekoya, A. F., & Weyori, B. A. (2020). A comprehensive evaluation of

ensemble learning for stock-market prediction. *Journal of Big Data*, 7(1). https://doi.org/10.1186/s40537-020-00299-5

- MSCI. (n.d.). GICS® - Global Industry Classification Standard. MSCI. https://www.msci.com/our-solutions/indexes/gics

Appendix

Appendices

Appendix A: Data Visualization Images

Figure A1 showcases the initial raw dataset as obtained from the 'yfinance' module.

	Adj Close		Close		High		Low		Open		Volume	
	AAPL	AMZN	AAPL	AMZN	AAPL	AMZN	AAPL	AMZN	AAPL	AMZN	AAPL	AMZN
Datetime												
2023-10-16 09:30:00-04:00	178.129898	130.759995	178.129898	130.759995	178.179993	131.070007	176.509995	130.425003	176.750000	130.690002	4407466	1974972
2023-10-16 09:35:00-04:00	177.869904	131.160004	177.869904	131.160004	178.360001	131.440002	177.809998	130.750000	178.119995	130.835007	1454933	1003305
2023-10-16 09:40:00-04:00	177.300003	131.072693	177.300003	131.072693	178.322495	131.630005	177.277496	130.970001	177.869995	131.172607	1141014	837142
2023-10-16 09:45:00-04:00	177.110001	131.380005	177.110001	131.380005	177.509995	131.719894	177.070007	131.000000	177.320007	131.085007	882999	728300
2023-10-16 09:50:00-04:00	177.009995	131.929993	177.009995	131.929993	177.429993	131.970001	176.930099	131.384995	177.119995	131.389999	774887	746733

Figure A1. Initial raw dataset head retrieved from yfinance for Apple and Amazon stock

Figure A2 illustrates the structured dataset with 'window_ID' assignments and additional calculated features.

	stock	start_date	window_ID	timestep	open	volatility	volume	target
0	DIS	2023-10-31	770	1	80.690002	0.279999	221818	0
1	DIS	2023-10-31	770	2	80.500000	0.080002	132538	1
2	DIS	2023-10-31	770	3	80.430000	0.090004	78192	0
3	DIS	2023-10-31	770	4	80.519997	0.169998	102693	1
4	DIS	2023-10-31	770	5	80.419998	0.089996	99674	1

Figure A2. Structured dataset head with additional features for training data

Figure A3 depicts the finalized and normalized dataset ready for model training and evaluation.

	window_ID	timestep	open	volatility	volume	target
0	770	1	0.089041	1.200881	0.170300	0
1	770	2	0.085411	-0.032389	-0.051593	1
2	770	3	0.084073	0.029289	-0.186662	0
3	770	4	0.085793	0.522569	-0.125768	1
4	770	5	0.083882	0.029242	-0.133271	1

Figure A3. Finalized and normalized dataset head for training data