NEW YORK UNIVERSITY TANDON SCHOOL OF ENGINEERING

Department of Finance and Risk Engineering

Machine Learning for Finance

Instructor: Prof. Amine Aboussalah

Portfolio Construction

Group Members

Syed Ahzam Tariq (sat10045@nyu.edu)

Ramya Venkatesh (rv2357@nyu.edu)

Tanishk Deoghare (td2479@nyu.edu)

Submitted: December 11, 2023

Abstract

We implemented a *two-stage portfolio optimization* strategy that predicts the stock movement direction and uses Markovitz mean-variance optimisation to construct a portfolio. The first stage of the strategy uses an ensemble of SVM, LightGBM, GBR, Random Forest, LSTM and XGBoost to forecast the price movement direction of stocks. The second stage constructs a portfolio using the Markovitz mean-variance optimisation and re-balances the portfolio on a daily basis. Forecasting returns was also implemented to build a portfolio, however, the results were not encouraging. Hence, we focused solely on a portfolio strategy that predicts the stock movement direction and uses it to construct a portfolio. The portfolio constructed is found to be giving lesser returns than S&P 500.

Contents

1 Introduction and Literature review

Artificial Intelligence is the most prominent advancement achieved by humans, often considered as revolutionary as the Internet and Mobile Phones [1]. In case of finance, Fintech firms, investment & retail banks and hedge funds are heavily investing in this emerging field [2]

This paper presents a two-stage portfolio construction technique based on stock return direction prediction using machine learning. Such a two-stage technique allows the portfolio manager to have a greater control and interpretability of the predictions from the model. Firstly we tried to predict the returns of the stocks under consideration using 6 machine-learning models. These returns are then used to construct a portfolio using Markowitz Portfolio optimization technique. This two-stage technique is suitable for financial applications since the return forecasts can help the portfolio manager and the investors understand the rationale behind the portfolio weights suggested by the algorithm. Portfolio managers have a duty towards their clients to explain the risks. Therefore, this approach naturally becomes attractive as it has a certain degree of interpretability in it and it can be a natural choice for both the investors and the portfolio managers[3]. Also, Markowitz mean-variance optimization is widely understood by investors and portfolio managers making this analysis comprehensible for a wider audience.

Zhang et al concluded that the complex models, without proper guidance, may instead lead to over-fitting or select false variable correlations. To choose the most appropriate data analysis method, analysts need to be familiar with different Machine Learning methods and their advantages and disadvantages; including the details of applying these models in financial forecasting, the need to have a solid understanding of the underlying data being modeled, and a strong market intuition. [22] Also, the work of Aziz et al suggests that firms

cannot simply 'apply' a machine learning risk management solution, but rather that it is a continuous process requiring a constant evaluation of whether their particular machine learning solution is currently considered best practice. When it comes to AI, where there is some or full automation of processes from data gathering to decision-making, the need for human oversight will become even more pressing. [21] This supports our approach of using a two-stage approach toward portfolio optimization. According to Ferrer et al in most cases, the datasets are composed exclusively of price data joined by other indicators of prices, such as maximum, minimum, volume, opening price or economic factors. [23] Hence we have followed a similar set of predictors in our model as well.

However, as per the preliminary results in the appendix, we are not able to predict the prices with a 1-day horizon very accurately. As per the work of Ismail et al [5], we understand that the price direction can be predicted with a much better accuracy, so we direct our focus on this side after the preliminary trials. As per the work of Ballings et al [6], we are using an ensemble approach for the same as it has yielded positive and encouraging results. Ansary et al [7] worked on various classification algorithms and concluded that they yield excellent results in predicting the direction of stocks.

Current works in the domain are also incorporating sentiment analysis through various mediums. Sidogi et al showed that incorporating sentiment features significantly improved the prediction results. [17] The work of Ma et al concludes that incorporating Geopolitical Risk in the analysis can yield significantly good results [18]. Nofer et al [19] reported greater accuracy after incorporating investor mood from Twitter data. These authors have reported a significant increase in accuracy after incorporating attention and mood values. However, in this work, we have limited ourselves to financial and market data but incorporating sentiment, attention, and mood could be a natural extension of the current work.

2 Dataset

We will assume that our stock universe is composed of the top 12 US stocks: MSFT - Microsoft Corporation, JPM - JPMorgan Chase & Co., PFE - Pfizer Inc., BRK-B - Berkshire Hathaway Inc. (Class B shares), ORCL - Oracle Corporation, AAPL - Apple Inc., IBM - International Business Machines Corporation, VZ - Verizon Communications Inc., BAC - Bank of America Corporation, KO - The Coca-Cola Company, NVDA - NVIDIA Corporation, PG - Procter & Gamble Company. To train our ML models, we will use the Daily traded volume of stocks, daily returns, stock price, S&P 500 index, 10-day Moving Average, 10-day exponential moving average, 10-day RSI, VIX index, 10-day historical rolling volatility, and 10-year US Treasury yields. The reason behind choosing these specific stocks was that the days of these was available for the last 20 years. This helped us in testing our strategy over a period of more than 15 years.

We have used 3 technical indicators in our analysis: 10-day moving average, 10-day exponential moving average, and 10-day Relative Strength Index. The use of these technical indicators has also been proposed by Dhokane et al who have obtained good results using moving average, exponential moving average, and RSI [25]. The use of moving averages has also been advocated by Gunasekarage [26].

2.1 Features for ML Model Training

In order to train our machine learning models, we will use the following features, which will be retrieved from Yahoo Finance for each stock in the top 20 US stocks:

- **Daily Traded Volume:** Represents the total number of shares traded each day.
- **Daily Returns:** Percentage change in the stock price from one day to the next.
- **Stock Price:** Current price of the stock.

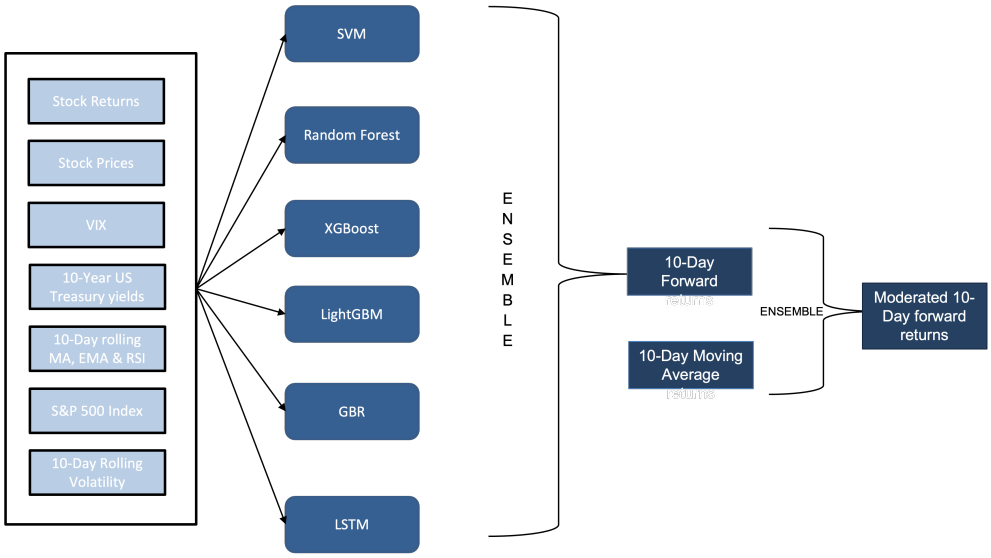


Figure 1: Our Machine Learning approach for forecasting returns

- **S&P 500 Index:** Represents the performance of the broader stock market.
- **10-Day Moving Average:** Helps identify trends and support/resistance levels.
- **10-Day Exponential Moving Average:** Places more weight on recent data than older data.
- **10-Day RSI (Relative Strength Index):** Measures the magnitude of recent price movements.
- **VIX Index:** Captures market volatility expectations.
- **10-Day Historical Rolling Volatility:** Measures the variability of returns over a specific period.
- **10-Year US Treasury Yields:** Represents the risk-free rate of return.

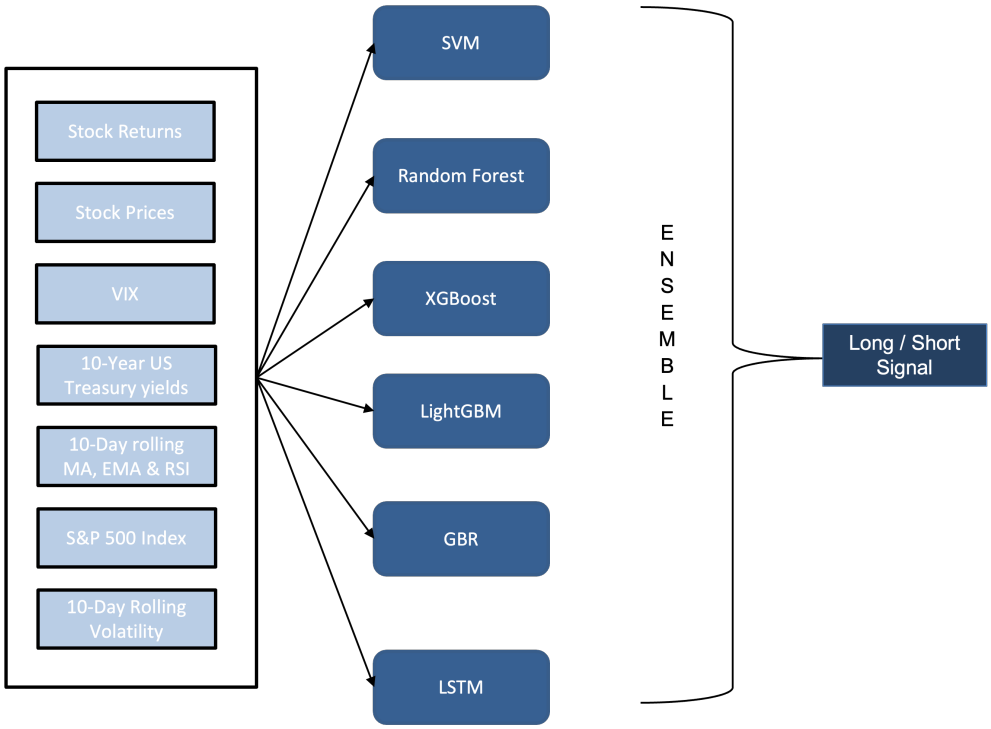


Figure 2: Our Machine Learning approach for long/short signal prediction*

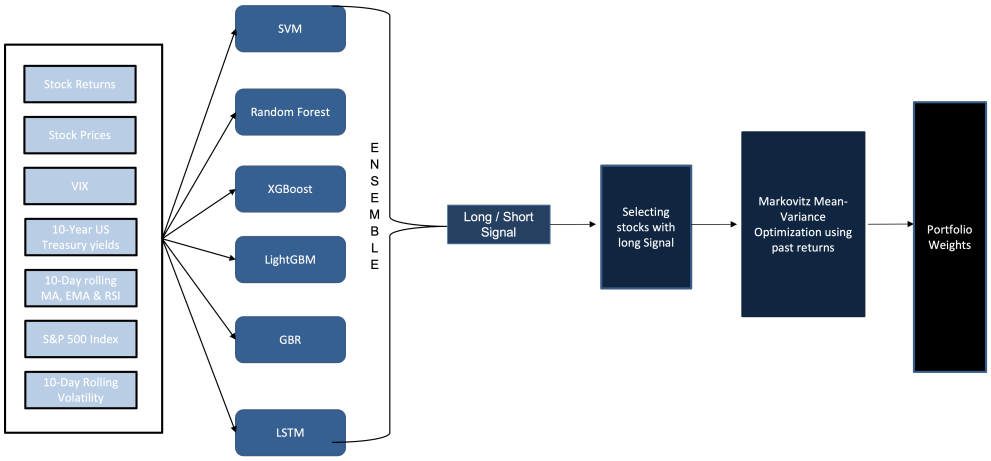


Figure 3: Our Machine Learning approach for using signal prediction strategy to construct Markovitz Portfolio

3 Methodology

We are first performing a regression analysis where we are predicting the stock prices using an ensemble model which will combine the predictions of 6 forecasting models: SVR, XGBoost, LightGBM, LSTM, Random Forest Regression, and GBR to produce a final forecast for the future returns of each asset. We have considered Jan-01-2020 to Dec-31-2022 under consideration. Of this, our training period is first 80% and we have tested our models on the remaining 20% of the period. The rationale behind choosing these models is that they have achieved good performance in predicting and other financial time series [2], [3], [4], [5], [6], [7], [8], [9]. Also, the use of ensemble learning has been shown superior to individual models for portfolio management [10]. The two-stage approach followed by us allows us to have better control as compared to a one-stage reinforcement learning model, as we get a peek inside after the first stage and can provide better explainability of the results to the management and clients as a portfolio manager. In normal circumstances, we would have proceeded to construct a portfolio with Markovitz mean-variance optimization.

However, as per the preliminary executions of the analysis, we have come to the understanding that it is being very difficult for our model to predict values since our results metrics are not very positive and our regression models are exhibiting large errors. This preliminary analysis was performed on top 20 stocks by market cap (listed in the US Markets)

Hence, we are implementing an improvised strategy using an ensemble model consisting of 5 Machine learning algorithms and combining LSTM to make another ensemble model. This ensemble model will help us predict whether the stock will move either up or down in the next trading day. As per the preliminary results (available in the appendix), we are getting good accuracy for the direction prediction. We have shown the flowchart of

our price direction prediction algorithm in Figure 2. Unlike our regression algorithm, we have not included a moderated forecast in our classification algorithm as we tested it and it yielded poor results. Also, as available in the appendix, the rolling moving average is not a good predictor for price direction prediction. *Also, we have implemented a slightly different version of our ensemble in our code. We have in fact first taken the ensemble of SVM, RandomForest, XGBoost, LightGBM and GBR. We have then used this ensemble model to construct our final ensemble model using the ensemble of the earlier 5 and the LSTM. This approach was found to give slightly better outcome in our case.

We will then use our signals generated from our classification algorithm to select the stocks with the long-signal from the 12-stocks under consideration. We will then construct a Markowitz portfolio assigning weights basis historical returns. This algorithm is depicted in Figure 3, by means of a flowchart, for ease of understanding.

3.1 Markowitz Mean Variance Optimisation

Markowitz portfolio theory has been a standard in portfolio optimisation and construction [33]. Due to its presence for a long time, it is clearly understood and accepted by a wider audience. We have made use of this methodology in our portfolio construction. For the stocks having a long prediction, we calculate their weights as per Markowitz Mean Variance Optimisation using past returns of 5 years. Here, we apply a constraint that all individual weights should be greater than or equal to zero in order to avoid short-selling. This is one of the major model assumptions and has been discussed critically in the Shortcomings of the current model section.

3.2 Performance Measurement through a trading strategy

In order to measure our portfolio performance, we propose a trading strategy that takes a long position for every stock whose price is expected to go up in the next day. We assume that we have an initial capital of $1,000,000 and we trade using 90% of the balance at any day. Other assumptions include, trading at closing price of the day. These assumption of this approach have been critically analysed in the Shortcomings of the current model section. We re balance our portfolio daily and select weights according to the Markowitz Mean Variance Optimisation. We then compare our portfolio performance with S&P500 over the period. We have conducted out-of-sample performance tests and compared them with S&P 500 for almost 2 decades. These results have been analysed in the next section.

4 Models

4.1 LSTM

Introduction and Literature Review

LSTM stands for Long-Short Term Memory. LSTM is a type of recurrent neural network but is better than traditional recurrent neural networks in terms of memory. Having a good hold over memorizing certain patterns, LSTMs perform fairly better. As with every other NN, LSTM can have multiple hidden layers, and as it passes through every layer, the relevant information is kept, and all the irrelevant information gets discarded in every single cell.

Fischer and Krauss used LSTM networks for the classification problem of predicting directional movements for the constituent stocks of S&P 500 from 1992 until. The authors concluded that the LSTM network could effectively extract meaningful information from the financial time series data. Based on prediction accuracy and daily returns after transaction costs, LSTM outperforms random forests, standard deep networks, and logistic regression [44].

M. Roondiwala et al. [43] proposed that the Long Short-Term Memory is the most popular RNN architecture. In the secret network layer, LSTM introduces a memory cell; a processing device that replaces conventional artificial neurons. Using these memory cells, networks can effectively link memory and remote input in time, making it suitable to dynamically capture data structure over time with a high predictive limit.

LSTM has 4 main components [42]:

1. **FORGET Gate**

- Decides relevant information for the cell state.
- Takes inputs h_{t-1} (previous hidden state) and x_t (current input).
- Sigmoid function filters information, discarding values tending towards 0.

2. **Input Gate**

- Updates cell state by identifying important information.
- Takes inputs h_{t-1} and x_t, processed by sigmoid and tanh functions.
- Tanh function regulates the network and reduces bias.

3. **Cell State**

- Utilizes information to compute the new cell state.
- Multiplies the cell state with the forget gate output, dropping less relevant values.
- Performs pointwise addition with the input gate output to update the cell state.

Step-by-Step LSTM Walkthrough

The first step is to decide what information we're going to forget from the cell state, shown as f_t. This decision is made by the forget gate. In the stock price prediction problem, the cell state might store the current stock price, and when the new stock price appears, this gate can forget the previous stock price.

The second step is to decide what new information we're going to store in the cell state. In this step, the LSTM unit will work in two parts. Firstly, the input gate will decide which values it we are going to update. Secondly, a tanh layer creates a vector of new candidate values C_t. Then these two values will be combined and are used to create an update to the state. For instance, we add the new stock price to our prediction model to replace the old onewe have forgotten.

The third step is to update the old cell state C_{t-1} to the new cell state C_t so that the new cell state can have both long-term features and short-term features.

The last step is to decide what is the output of this LSTM unit. First, a sigmoid layer is used to decide what parts of the cell state will be output. Then, the cell state will go

through a tanh layer and is multiplied by the output of the sigmoid gate, so that we only output the parts we decided to.

Backpropagation

Like most deep learning approaches, we also use Backpropagation to optimize the parameters of LSTMs.

There are three steps backward:

1. Calculate the output of every neuron. For LSTM, it is five variables, f_t, i_t, C_t, o_t, h_t.
2. Work backward to calculate the error of every neuron δ.
3. Calculate the gradient according to δ.

4.2 Support Vector Machine (SVM)

Introduction and Literature Review

The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space (N - the number of features) that distinctly classifies the data points. To separate the two classes of data points, there are many possible hyperplanes that could be chosen. Our objective is to find a plane that has the maximum margin, i.e., the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence.

Numerous researches have been conducted to investigate machine learning algorithms for stock market forecasting. In the stock market, fundamental and technical analysis are key phenomena. The combination of these indicators resulted in improved forecasting performance in nearly 95% of the cases. These results show that Technical and Fundamental analysis-based indicators can be used together for forecasting stock prices using machine learning-based models whenever possible for improved results. In most cases, Random Forest, Support Vector Machine (SVM), and Long Short-Term Memory Neural Network (LSTM) demonstrated better results compared to other machine learning algorithms .

Primal Optimization Problem (Soft Margin)

In cases where the categories are not linearly separable, it implies that for every hyperplane that exists, there exists $x_i \in S$ such that:

$$y_i[w \cdot x + b] < 1$$

This means that we cannot simultaneously hold all the constraints for the given data set. So, we try to relax a bit by introducing slack variables. The slack variables, ξ_i, as shown in the Figure and the below equation, are commonly used in optimization problems to relax the constraints. With slack variables, our equation takes the following form for each $i \in [m]$:

$$y_i(w \cdot x + b) \geq 1 - \xi_i$$

A slack variable $\xi_i \geq 0$ measures the distance by which the vector x_i violates the desired inequality $y_i(w \cdot x + b) \geq 1$. Slack variables are used in optimization to define relaxed versions of constraints. Each x_i must be positioned on the correct side of the appropriate marginal hyperplane to ascertain that it is not considered an outlier. As a consequence, a vector x_i with $0 < y_i(w \cdot x + b) < 1$ is correctly classified but still labeled as an outlier, i.e., $\xi_i > 0$.

Now we are faced with a very serious question. In the case of non-separable variables, how should we then choose the hyperplane? We are faced with two conflicting objectives: we want to minimize the total amount of slack due to outliers and at the same time, we seek a hyperplane with a large margin.

This leads us to a general optimization problem in the case of linearly-inseparable SVMs where the parameter $C \geq 0$ determines the trade-off between margin maximization and the minimization of the slack penalty:

$$\min_{w,b,\xi_i} \frac{1}{2}||w||^2 + C \sum_{i=1}^{m} \sqrt{\xi_i}$$

Subject to:

$$y_i(w \cdot x_i + b) \geq 1 - \xi_i \quad \text{and} \quad \xi_i \geq 0, \quad i \in [m]$$

The parameter C is typically determined by n-fold cross-validation. The above optimization problem obtained is a convex optimization problem since the constraints are affine, and the objective function is convex for any $p \geq 1$. There are many possible choices for p leading to more or less aggressive penalization of the slack terms. The choices $p = 1$ and $p = 2$ lead to the most straightforward solutions and analyses. The loss functions associated with $p = 1$ and $p = 2$ are called the hinge loss and the quadratic hinge loss, respectively. Figure 8 shows the plots of these loss functions as well as that of the standard zero-one loss function. Both hinge losses are convex upper bounds on the zero-one loss, thus making them well-suited for optimization. In what follows, the analysis is presented in the case of the hinge loss ($p = 1$), which is the most widely used loss function for SVMs.

Support Vectors (Soft Margin)

In this case as well, the support vectors are affine and thus qualified. The objective function as well as the affine constraints are convex and differentiable. Therefore, we can apply the KKT conditions at the optimum.

We introduce Lagrange variables $\alpha_i \geq 0$, $i \in [m]$, associated with the first m constraints, and $B_i \geq 0$, $i \in [m]$, associated with the non-negativity constraints of the slack variables. We define the Lagrangian for all $w \in R^n$, $b \in R$, and $\xi, \alpha, \beta \in R^m_+$ as:

$$L(w, b, \xi, \alpha, \beta) = \frac{1}{2}||w||^2 + C\sum_{i=1}^{m} \xi_i - \sum_{i=1}^{m} \alpha_i(y_i(w \cdot x_i + b) - 1 + \xi_i) - \sum_{i=1}^{m} \beta_i \xi_i$$

The KKT conditions are obtained by setting the gradient of the Lagrangian with respect to the primal variables w, b, and ξ to zero and by writing the complementarity conditions:

$$\nabla_w L = w - \sum_{i=1}^{m} \alpha_i y_i x_i = 0 \Rightarrow w = \sum_{i=1}^{m} \alpha_i y_i x_i$$

$$\nabla_b L = -\sum_{i=1}^{m} \alpha_i y_i = 0 \Rightarrow \sum_{i=1}^{m} \alpha_i y_i = 0$$

$$\nabla_{\xi_i} L = C - \alpha_i - \beta_i = 0 \Rightarrow \alpha_i + \beta_i = C$$

For all i, $\alpha_i(y_i(w \cdot x_i + b) - 1 + \xi_i) = 0 \Rightarrow \alpha_i = 0 \vee y_i(w \cdot x_i + b) = 1 - \xi_i$ and $\beta_i \xi_i = 0 \Rightarrow \beta_i = 0 \vee \xi_i = 0$.

For support vectors, $\alpha_i \neq 0$. As in the separable case, note that while the weight vector w solution is unique, the support vectors are not. If $\xi_i = 0$, then $y_i(w \cdot x_i + b) = 1$ and x_i lies on the marginal hyperplane.

Dual Optimization Problem (Soft Margin)

Plugging in the values of w_i and the constraint $\sum_{i=1}^{m} \alpha_i y_i = 0$, we get:

$$L = \frac{1}{2} \left\| \sum_{i=1}^{m} \alpha_i y_i x_i \right\|^2 - \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) + \sum_{i=1}^{m} \alpha_i$$

Since $\frac{1}{2} \left\| \sum_{i=1}^{m} \alpha_i y_i x_i \right\|^2 - \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) = -\frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j)$, we get:

$$L = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j)$$

Here, in addition to the condition $\alpha_i \geq 0$, we must also impose $\beta_i \geq 0$, which in turn implies that $\alpha_i \leq C$ since $\alpha_i + \beta_i = C$. This leads to the following dual optimization problem for SVMs in the non-separable case, which only differs from that of the separable case by the constraint $\alpha_i \leq C$:

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) \\ \text{subject to} \quad & 0 \leq \alpha_i \leq C \quad \text{and} \quad \sum_{i=1}^{m} \alpha_i y_i = 0, \quad i \in [m] \end{aligned}$$

The solution α of the dual problem can be used to determine the hypothesis returned by the SVM:

$$h(x) = \operatorname{sgn} \left(\sum_{i=1}^{m} \alpha_i y_i (x_i \cdot x) + b \right)$$

b can be obtained from any support vector x_i lying on the marginal hyperplane as:

$$b = y_i - \sum_{i=1}^{m} \alpha_i y_i (x_j \cdot x_i)$$

4.3 Random Forest Classification

Introduction and Literature Review

Random forest is a supervised learning algorithm, meaning that the data on which it operates contains labels or outcomes. It works by creating many decision trees, each built on randomly chosen subsets of the data. The model then aggregates the outputs of all of these decision trees to make an overall prediction for unseen data points. In this way, it can process larger datasets and capture more complex associations than individual decision trees. Here, each tree makes a prediction for a new data point, and the final prediction is the majority vote of all the trees.

Sadorsky has demonstrated [34] that Decision tree bagging and random forests predictions of stock price direction are more accurate than those obtained from logit models. For a 20-day forecast horizon, tree bagging and random forests methods produce accuracy rates of between 85% and 90% while logit models produce accuracy rates of between 55% and 60%. He also suggests the use of macroeconomic variables to offer additional insight into predicting the direction of stock prices which will scope future research.

Khaidem, Saha, and Dey [35] have worked on long-term direction where they were able to achieve accuracy in the range of 85-95% by using the random forest classifier. Tan, Yan, and Zhu [36] explain how fundamental analysis and long-term technical features are required for long-term direction prediction.

Model Summary [37]

1. Step-1: Make subsets of the original data through row and feature sampling with replacement, creating subsets of the training dataset.

2. Step-2: Create an individual decision tree for each subset.

3. Step-3: Each decision tree provides an output.

4. Step-4: Final output is considered based on Majority Voting for a classification problem.

The individual decision tree models are constructed using bootstrapping, a resampling method that uses random sampling with replacement.

Loss Function

The data is recursively split into partitions. At a particular node, the split is done by asking a question on an attribute. The choice for the splitting criterion is based on impurity measures such as Shannon Entropy or Gini impurity. In our model, we use Gini impurity as the criterion for the Random Forest Classifier.

Gini is the probability of a random sample being classified correctly when selected randomly. Gini denotes purity, whereas Gini impurity tells us about the impurity of nodes. Gini impurity gives us the probability of misclassifying an observation, i.e., the probability of a particular feature being classified incorrectly when selected randomly [38].

Therefore, Gini impurity is the summation of the product of p_i (probability of an item with label i being chosen) and $1 - p_i$ (probability of misclassification). The minimum value of the Gini Index is 0 (when the node is pure, i.e., all the elements in the node are of one particular class). And when all the cases in the node belong to a specific category, the Gini index reaches zero. Also, the maximum value is 0.5, and it is when the probability of the two classes is the same.

4.4 XGBoost

Introduction and Literature Review

In 2016, the XGBoost model was created by Chen and Guestrin and is one of the most advanced decision tree models. Similar to the random forest model, it allows choosing only a subset of features. Like the AdaBoost model, XGBoost employs boosting, creating trees

sequentially to improve the error after each tree. Additionally, XGBoost utilizes a gradient, leading to the name Extreme Gradient Boost . The standout feature of XGBoost lies in its use of various strategies such as tree pruning, regularization, among others, to significantly reduce the training time of the model and prevent overfitting while maintaining predictive power. XGBoost has proven to be a powerful machine learning technique and is frequently the algorithm of choice in many machine learning competitions. In our paper, XGBoost is one of the six models used in ensemble learning for stock direction prediction. The predicted directions by XGBoost are then employed in portfolio construction, with each of the six models weighted by their mean square error.

XGBoost Working Principle

XGBoost works by combining a number of weak learners to form a strong learner. A weak learner is a machine learning model that is only slightly better than random guessing. However, when weak learners are combined, they can form a strong learner that is much more accurate.

XGBoost operates by training a number of decision trees. Each tree is trained on a subset of the data, and the predictions from each tree are combined to form the final prediction. It represents an improvement on the Gradient Boosting Machine (GBM) algorithm. The main difference is that XGBoost uses a more regularized model, which helps prevent overfitting.

Pruning Parameter[1]

In our case, XGBoost is used to classify whether the direction of the stock is predicted to rise or fall. The example of a tree is below:

The prediction scores of each individual decision tree then sum up to get $\hat{y} = \sum_{k=1}^{K} f_k(x)$

If you look at the example, an important fact is that the two trees try to complement

[1]Reference [1]

each other. Mathematically, we can write our model in the form:

$\hat{y} = \sum_{k=1}^{K} f_k(x)$ where, K is the number of trees f is the functional space of F, F is the set of possible day

The objective function for the above model is given by:

$$\text{Objective} = \sum_{i=1}^{n} \ell(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

where the first term is the loss function and the second is the regularization parameter. Now, instead of learning the tree all at once which makes the optimization harder, we apply the additive strategy, minimize the loss what we have learned, and add a new tree which can be summarized below:

$$\text{Objective}^{(t)} = \sum_{i=1}^{n} \ell(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) + \text{constant}$$

The objective function of the above model can be defined as:

$$\text{Objective}^{(t)} = \sum_{i=1}^{n} \left[\ell(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i (f_t(x_i))^2 \right] + \Omega(f_t) + \text{constant}$$

Now, let's apply Taylor series expansion up to the second order:

$$\text{Objective}^{(t)} \approx \sum_{i=1}^{n} \left[\ell(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i (f_t(x_i))^2 \right] + \Omega(f_t) + \text{constant}$$

Here, w is the vector of scores on leaves of the tree, q is the function assigning each data point to the corresponding leaf, and T is the number of leaves. The regularization term is then defined by:

$$\Omega(f) = \gamma T + \frac{1}{2} \lambda \sum_{j=1}^{T} w_j^2$$

Now, our objective function becomes:

$$\text{Objective}^{(t)} \approx \sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i (f_t(x_i))^2 \right] + \gamma T + \frac{1}{2} \lambda \sum_{j=1}^{T} w_j^2 + \text{constant}$$

Now, we simplify the above expression:

$$\text{Objective}^{(t)} = \sum_{j=1}^{T} \left[\left(\sum_{i \in I_j} g_i \right) w_j + \frac{1}{2} \left(\sum_{i \in I_j} h_i + \lambda \right) w_j^2 \right] + \gamma T$$

where,

$$I_j = \{ i | q(x_i) = j \}$$

In this equation, w_j are independent of each other. The best w_j for a given structure $q(x)$ and the best objective reduction we can get is:

$$\text{Best}_{w_j} = -\frac{(\sum_{i \in I_j} g_i)^2}{\sum_{i \in I_j} h_i + \lambda}$$

where γ is the pruning parameter, i.e., the least information gain to perform a split.

Now, we try to measure how good the tree is. We can't directly optimize the tree; we will try to optimize one level of the tree at a time. Specifically, we try to split a leaf into two leaves, and the score it gains is:

$$\text{Gain} = \frac{1}{2} \left[\frac{(\sum_{i \in I_L} g_i)^2}{\sum_{i \in I_L} h_i + \lambda} + \frac{(\sum_{i \in I_R} g_i)^2}{\sum_{i \in I_R} h_i + \lambda} - \frac{(\sum_{i \in I} g_i)^2}{\sum_{i \in I} h_i + \lambda} \right] - \gamma$$

Gradient Boosting Classifier (GBR)

Introduction and Literature Review

Gradient Boosting Classifier is an ensemble machine learning algorithm that iteratively builds a predictive model by combining weak learners, often decision trees, to sequentially correct errors made by the ensemble. It minimizes a predefined loss function by training each new weak learner to predict the negative gradient of the loss with respect to the

current ensemble's predictions. The final model is a weighted sum of all weak learners, and for binary classification, it outputs probabilities through a sigmoid transformation of the sum of predictions, providing a robust and accurate predictive model for complex classification tasks.

A Mathematical Understanding

x_i - This is the input variables that we feed into our model.

y_i - This is the target variable that we are trying to predict.

We can predict the log likelihood of the data given the predicted probability.

y_i is the observed value (0 or 1).

p is the predicted probability.

The goal would be to maximize the log likelihood function. Hence, if we use the log(likelihood) as our loss function where smaller values represent better fitting models then:

$$\log(\text{likelihood}) = \sum_{i=1}^{n} [y_i \log(p) + (1 - y_i) \log(1 - p)]$$

Now the log(likelihood) is a function of predicted probability p but we need it to be a function of predictive log(odds). So, let us try and convert the formula:

We know that:

$$\log(\text{odds}) = \log\left(\frac{p}{1-p}\right)$$

Substituting,

$$\log(\text{odds}) = \log(p) - \log(1 - p)$$

Now,

$$\log(p) = \log(\text{odds}) + \log(1 - p)$$

Hence,

$$p = \frac{e^{\log(\text{odds})}}{1 + e^{\log(\text{odds})}}$$

Now that we have converted p to log(odds), this becomes our Loss Function. We have to show that this is differentiable.

This can also be written as:

$$p = \frac{1}{1 + e^{-\log(\text{odds})}}$$

Now we can proceed to the actual steps of the model building.

Step 1: Initialize model with a constant value

Here, y_i is the observed values, L is the loss function, and γ is the value for log(odds). We are summing the loss function, i.e., we add up the Loss Function for each observed value. argmin over γ means that we need to find a log(odds) value that minimizes this sum. **Step 2: for m = 1 to M:**

(A)

This step needs you to calculate the residual using the given formula. We have already found the Loss Function to be as:

$$\text{Loss Function} = -[y_i \log(p) + (1 - y_i) \log(1 - p)]$$

Hence,

$$\text{Residual} = y_i - p$$

(B)

Fit a regression tree to the residual values and create terminal regions. Because the leaves are limited for one branch hence, we might have more than one value in a particular

terminal region. In our first tree, $m = 1$ and j will be the unique number for each terminal node. So R_{11}, R_{21}, and so on.

(C)

For each leaf in the new tree, we calculate γ, which is the output value. The summation should be only for those records which go into making that leaf. In theory, we could find the derivative with respect to γ to obtain the value of γ but that could be extremely wearisome due to the hefty variables included in our loss function.

(D)

This formula is asking us to update our predictions now. In the first pass, $m = 1$, and we will substitute $F_0(x)$, the common prediction for all samples, i.e., the initial leaf value plus ν, which is the learning rate into the output value from the tree we built previously. The summation is for the cases where a single sample ends up in multiple leaves. Now we will use this new $F_1(x)$ value to get new predictions for each sample. The new predicted value should get us a little closer to the actual value. It is to be noted that in contrast to one tree in our consideration, gradient boosting builds a lot of trees, and M could be as large as 100 or more.

This completes our for loop in Step 2, and we are ready for the final step of Gradient Boosting.

4.5 LightGBM

Introduction and Literature Review

LightGBM (LGBM) is an open-source gradient boosting library that has gained tremendous popularity and fondness among machine learning practitioners. It has also become one of the go-to libraries in Kaggle competitions. It can be used to train models on tabular data with incredible speed and accuracy. This performance is a result of the way LightGBM samples the data (GOSS — Gradient-based One-Sided Sampling) and reduces the number

of features (EFB - Exclusive Feature Bundling) in sparse datasets during training [41].

The LightGBM algorithm developed by one of the Microsoft groups has also started to be applied to the financial field, and studies have proved that the stock price prediction model based on LightGBM has better prediction ability and higher returns [39].

LGBM is an advanced implementation of gradient boosting machines (GBTs) that introduces innovative techniques to enhance efficiency and performance compared to vanilla GBTs. The key ideas include Histogram-based Splitting Point Selection, Gradient-based One-Side Sampling (GOSS), and Exclusive Feature Bundling (EFB).

1. Histogram-based Splitting Point Selection:

- LGBM employs a histogram-based approach to find optimal splitting points in continuous features.
- Data is divided into a fixed number of bins, and the algorithm iterates through these bins to identify the best split points.
- This method is computationally efficient with a complexity proportional to $O(\text{data\_count} \times \text{features\_count})$, making it faster than traditional GBTs.

2. Gradient-based One-Side Sampling (GOSS):

- GOSS addresses efficiency by focusing on samples with higher gradients, which are more influential in the boosting process.
- a% of examples with the highest gradients and b% from the remaining samples are selected.
- Gradients of the b% examples are scaled by $(1-a)/b$, ensuring that smaller gradients are not neglected.
- This reduces the number of training examples at each boosting step, leading to memory and time savings.

3. Exclusive Feature Bundling (EFB):

- EFB is effective for sparse data where features often have zero values simultaneously.
- It reduces complexity by combining features into exclusive feature bundles.
- The process involves solving a graph coloring problem to identify mutually exclusive features.
- Bundles are created by considering combinations with a conflict count below a predefined threshold (K).
- While merging bundles, LGBM aims to preserve information by using all bins from constituent features with offset values.

These innovations collectively make LightGBM more efficient and faster, especially for large datasets with numerous features. The histogram-based method, gradient-based sampling, and exclusive feature bundling contribute to the algorithm's speed and effectiveness, making it a popular choice for boosting tasks in machine learning.

4.6 GBR

Gradient Boosting Classifier is an ensemble machine learning algorithm that iteratively builds a predictive model by combining weak learners, often decision trees, to sequentially correct errors made by the ensemble. It minimizes a predefined loss function by training each new weak learner to predict the negative gradient of the loss with respect to the current ensemble's predictions. The final model is a weighted sum of all weak learners, and for binary classification, it outputs probabilities through a sigmoid transformation of the sum of predictions, providing a robust and accurate predictive model for complex classification task

Rodriguez, PN. & Rodriguez predicted the short-term movement of stock market prices. A comparison of different ML algorithms has been performed, and seven different classifications algorithms have been applied to predict the daily movements of the stock prices of three large emerging markets stock indices, IPC (Mexico), Bovespa (Brazil) and KLSE Composite (Malaysia) with a sample period from Jan-1990 to Dec-2003. The technical indicators which have been used in this paper are: 1- and 2-day ROC, 4-day Momentum, 14-day RSI and Stochastic, OSCP, and 14- and 21-day Disparity. Gradient Boosting Classifier however was one of the best classifications models used in predicting the direction of the stock market among all other models, where AUC-ROC curve was used in order to evaluate the performance of the models [40].

5 Hands-On

We are considering two stocks, Apple and Microsoft, using three days of historical values for training to predict the trend on the fourth day. Our feature set comprises daily returns and the VIX index. Employing ensemble learning, we combine SVM and random forest predictions to forecast the stock trend for the fourth day. Subsequently, we leverage Markowitz portfolio theory to construct an optimal portfolio based on the predicted outcomes.

For the Apple stock, our training set consists of three rows (1.1, 2.1), $-1.1, 2.1$), and $(-1.1, -2.1)$. Our testing set consists of (1.1, 1.1). Let us classify points first and third as 1 and point second as -1. Our aim is to find the classification of fourth day using Support Vector Machines and Random Forest Classifier.

5.1 Support Vector Machines Hands-On

The objective function for Support Vector Machines is given by:

$$L(w, b, \alpha) = \frac{||w||^2}{2} - \sum_{i=1}^{n} \alpha_i (y_i (x_i^T . w + b) - 1)$$

subject to the constraints:

$$\frac{w^T w}{2} - \sum_{i=1}^{n} \alpha_i y_i x_i^T + b \sum_{i=1}^{n} \alpha_i y_i + \sum_{i=1}^{n} \alpha_i$$

$$\frac{1}{2} \begin{bmatrix} w1 & w2 \end{bmatrix} \begin{bmatrix} w1 \\ w2 \end{bmatrix} + \alpha_1 \begin{bmatrix} 1.1 & 2.1 \end{bmatrix} \begin{bmatrix} w1 \\ w2 \end{bmatrix} + \alpha_2 \begin{bmatrix} -1.1 & 2.1 \end{bmatrix} \begin{bmatrix} w1 \\ w2 \end{bmatrix} - \alpha_3 \begin{bmatrix} -1.1 & -2.1 \end{bmatrix} \begin{bmatrix} w1 \\ w2 \end{bmatrix} + b(-\alpha_1 - \alpha_2 + \alpha_3) +$$

Now, compute the partial derivatives of L with respect to its primal variables:

$$\frac{\partial L}{\partial w} = 0 \implies w* = \sum_{i=1}^{n} \alpha_i y_i x_i$$

This gives us:

$$w* = -\alpha_1 \begin{bmatrix} 1.1 \\ 2.1 \end{bmatrix} - \alpha_2 \begin{bmatrix} -1.1 \\ 2.1 \end{bmatrix} + \alpha_3 \begin{bmatrix} -1.1 \\ -2.1 \end{bmatrix}$$

Now, let us differentiate L wrt b:

$$\frac{\partial L}{\partial b} = 0 \implies \sum_{i=1}^{n} \alpha_i y_i = 0$$

This leaves us with:

$$-\alpha_1 - \alpha_2 + \alpha 3 = 0$$

Let us now define the Dual form of our equation:

$$\underset{\alpha_1, \alpha_2, \alpha_3}{max} \frac{-1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j + \sum_{i=1}^{n} \alpha_i$$

Now, let us pre-process the pairwise products $x_i^T x_j$ for ease of our computation:

$$x_1^T x_2 = -1.21 + 4.41 = 3.2$$

$$x_1^T x_1 = 1.21 + 4.41 = 5.62$$

$$x_1^T x_3 = -1.21 - 4.41 = -5.62$$

$$x_2^T x_2 = 1.21 + 4.41 = 5.62$$

$$x_2^T x_3 = 1.21 - 4.41 = -3.2$$

$$x_3^T x_3 = 1.21 + 4.41 = 5.62$$

On expanding the dual form with these pair-wise products we get:

$$M^{\max}_{(\alpha_1,\alpha_2,\alpha_3)} - \frac{1}{2}\left[5.62 + 6.4\alpha_1\alpha_2 + 11.24\alpha_1\alpha_3 + 5.62\alpha_2^2 + 6.4\alpha_2\alpha_3 + 5.62\alpha_3^2\right] + \alpha_1 + \alpha_2 + \alpha_3$$

Let's eliminate α_3 using the earlier derived relations $\alpha_1 - \alpha_2 + \alpha_3 = 0$:

$$\alpha_1 + \alpha_2 = \alpha_3$$

This transforms our dual optimization problem as:

$$M^{\max}_{(\alpha_1,\alpha_2,\alpha_3)} - \frac{1}{2}\left[22.48\alpha_1^2 + 35.28\alpha_1\alpha_2 + 17.64\alpha_2^2\right] + 2\alpha_1 + 2\alpha_3$$

Now, let us compute the partial derivatives of the dual with respect to the dual variables α_1 and α_2 and equate them to zero:

$$\frac{\partial L}{\partial \alpha_1} = 0 : -22.48\alpha_1 - 17.64\alpha_2 + 2 = 0$$

$$\frac{\partial L}{\partial \alpha_2} = 0 : -17.64\alpha_2 - 17.64\alpha_1 + 2 = 0$$

From the above two equations, we get:

$$(\alpha_1, \alpha_2, \alpha_3) = (0, 0.11, -0.11)$$

This implies that points corresponding to α_2 and α_3 are our support vectors, i.e., points B and C. We now have the data to figure out our maximum margin classifier with a soft margin.

We will compute w^* as:

$$w^* = \sum_{i=1}^{3} \alpha_i y_i x_i$$

This implies:

$$w* = -0.11 \begin{bmatrix} -1.1 \\ 2.1 \end{bmatrix} + 0.11 \begin{bmatrix} -1.1 \\ -2.1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0.46 \end{bmatrix}$$

This gives us our margin width to be:

$$\frac{2}{||w*||} = 2 * 2.16 = 4.32$$

The value of b* can be computed through any support vector:

$$y_2(x_2^T w* + b*) = 1 \implies b* = 1 + x_2^T w* = 0$$

This implies that our equation of maximum margin classifier is:

$$\begin{bmatrix} x1 & x2 \end{bmatrix} \begin{bmatrix} 0 \\ -0.46 \end{bmatrix} = 0$$

which is x_2=0. Therefore, our point D (1.1,1.1) which is above our maximum margin classifier will be classified as -1.

5.2 Random Forest Classifier Hands-On

A Random Forest is a collection of decision trees. To simplify the demonstration of a hands-on random classifier, we will explain the calculations involved in just one tree.

Bootstrapping

Bootstrapping is a statistical method involving random sampling with replacement. In our context, it's often applied during the training of algorithms to create multiple subsets from a single dataset, allowing for robust model training and evaluation. Here, our model is selecting the 1st and 3rd rows of the dataset for training.

Calculations for Decision Tree

Feature 1

$$\text{Feature 1} = 1.1 - \frac{1.1}{2} = 0$$
$$\text{Feature 2} = 2.1 - \frac{2.1}{2} = 0$$

$$\text{Probability}(\text{Feature 1} > 0.0) = \frac{1}{3}$$
$$\text{Probability}(\text{Feature 1} < 0.0) = \frac{2}{3}$$
$$\text{Probability}(\text{Feature 1} > 0.0 \text{ and trend} = 1) = 0$$
$$\text{Probability}(\text{Feature 1} > 0.0 \text{ and trend} = -1) = 1$$

$$\text{Gini Index} = 1 - \left(1^2 + 0^2\right) = 0$$

$$\text{Probability}(\text{Feature 1} < 0.0 \text{ and trend} = 1) = \frac{1}{2}$$
$$\text{Probability}(\text{Feature 1} < 0.0 \text{ and trend} = -1) = \frac{1}{2}$$

$$\text{Gini Index} = 1 - \left(\left(\frac{1}{2}\right)^2 + \left(\frac{1}{2}\right)^2\right) = 0.5$$

$$\text{Weighted Average of Gini Index} = \frac{1}{3} \cdot 0 + \frac{2}{3} \cdot 0.5 = \frac{1}{3}$$

Feature 2

$$\text{Probability(Feature 2} > 0.0) = \frac{2}{3}$$
$$\text{Probability(Feature 2} < 0.0) = \frac{1}{3}$$
$$\text{Probability(Feature 2} > 0.0 \text{ and trend} = 1) = 0$$
$$\text{Probability(Feature 2} > 0.0 \text{ and trend} = -1) = 1$$

$$\text{Gini Index} = 1 - \left(1^2 + 0^2\right) = 0$$

$$\text{Probability(Feature 2} < 0.0 \text{ and trend} = 1) = 1$$
$$\text{Probability(Feature 2} < 0.0 \text{ and trend} = -1) = 0$$

$$\text{Gini Index} = 1 - \left(1^2 + 0^2\right) = 0$$

$$\text{Weighted Average of Gini Index} = \frac{1}{3} \cdot 0 + \frac{2}{3} \cdot 0 = 0$$

Since Feature 2 has the lowest Gini index, it will be the root node.

Since our point D's second feature is greater than 0.0, it will go to the left subtree, and our prediction will be a downtrend.

Ensemble Prediction

In this ensemble prediction, we consider a 50% weight for both our SVM and Random Forest classifiers' predictions to forecast the stock trend for the fourth day. For simplification of hands-on calculation, the ensemble prediction for AAPL is calculated as follows:

$$\text{ensemble\_prediction\_AAPL} = \frac{\text{prediction from SVM} + \text{prediction from Random Forest Classifier}}{2}$$
$$= \frac{(-1) + (-1)}{2}$$
$$= -1$$

The above steps are repeated for forecasting trend for MICROSOFT by SVM and Random Forest Classifier. SVM predicts a downtrend (-1), while the Random Forest Classifier predicts an uptrend (-1). To reconcile these predictions and provide a more robust ensemble forecast, we take the average of the individual predictions, resulting in a neutral or stagnant market outlook

Portfolio Construction:

w_{AAPL} : Weight of AAPL

w_{MSFT} : Weight of MSFT

R_{AAPL} : Average return of AAPL

R_{MSFT} : Average return of MSFT

σ_{AAPL} : Standard deviation of AAPL

σ_{MSFT} : Standard deviation of MSFT

$\rho_{\text{AAPL, MSFT}}$: Correlation coefficient between AAPL and MSFT

R_f : Risk-free rate

$$S = \frac{w_{\text{AAPL}} \cdot (R_{\text{AAPL}} - R_f) + w_{\text{MSFT}} \cdot (R_{\text{MSFT}} - R_f)}{\sqrt{w_{\text{AAPL}}^2 \cdot \sigma_{\text{AAPL}}^2 + w_{\text{MSFT}}^2 \cdot \sigma_{\text{MSFT}}^2 + 2 \cdot w_{\text{AAPL}} \cdot w_{\text{MSFT}} \cdot \sigma_{\text{AAPL}} \cdot \sigma_{\text{MSFT}} \cdot \rho_{\text{AAPL, MSFT}}}}$$

Given returns:

$$\text{AAPL\_Returns} = [1.1, -1.1, -1.1, 1.1]$$

$$\text{MSFT\_Returns} = [5.71, 2.4, 1.3, 2.0]$$

Risk-free rate: $R_f = 0.01$

$$R_{\text{AAPL}} = \frac{1.1 - 1.1 - 1.1 + 1.1}{4} = 0$$
$$R_{\text{MSFT}} = \frac{5.71 + 2.4 + 1.3 + 2}{4} = 2.85$$

$$\begin{aligned}\text{Standard Deviation (RAAPL)} &= \sqrt{\frac{(1.1-0)^2 + (-1.1-0)^2 + (-1.1-0)^2 + (1.1-0)^2}{4}} \\ &= \sqrt{\frac{1.1^2 + (-1.1)^2 + (-1.1)^2 + 1.1^2}{4}}\end{aligned}$$

$$\begin{aligned}\text{Standard Deviation (MSFT)} &= \sqrt{\frac{(5.71-2.85)^2 + (2.4-2.85)^2 + (1.3-2.85)^2 + (2-2.85)^2}{4}} \\ &= \sqrt{\frac{(2.86)^2 + (-0.45)^2 + (-1.55)^2 + (-0.85)^2}{4}} \\ &= \sqrt{\frac{8.1796 + 0.2025 + 2.4025 + 0.7225}{4}} \\ &= \sqrt{\frac{11.5071}{4}} \\ &= \sqrt{2.876775} \\ &\approx 1.6957\end{aligned}$$

$$\text{Cov}(R_{\text{MSFT}}, R_{\text{AAPL}}) = \frac{\sum_{i=1}^{n}(R_{\text{MSFT},i} - \bar{R}_{\text{MSFT}})(R_{\text{AAPL},i} - \bar{R}_{\text{AAPL}})}{n-1}$$

By substituting the values, we get 1.47 as the covariance between MSFT and AAPL returns.

$$\text{MSFT, AAPL} = \frac{\text{Cov}(R_{\text{MSFT}}, R_{\text{AAPL}})}{\sigma_{\text{MSFT}} \cdot \sigma_{\text{AAPL}}}$$

By substituting the values, we get 0.59 as the correlation between MSFT and AAPL returns.

Finally, substituting all the values in the Sharpe ratio maximization, we get our weights for MSFT and AAPL: 0% for AAPL and 100% for MSFT.

6 Results and Analysis

As per the analysis performed till now, we have obtained results from regression analysis and obtained trade signals via a different ensemble classification algorithm. Preliminary results in the form of graphs and confusion matrices have been documented in the appendix.

Since our trade signals generated via the classification algorithm yield far greater results, we are currently working on a strategy to use the trade signals generated by our classification algorithm to construct a portfolio using Markowitz mean-variance optimization. Our approach of using trade signals generated by our classification model is based on our observed performance of our classification model and is therefore expected to generate good results. As depicted by the below graph our portfolio has outperformed the index by a significant margin in the previous year even after incorporating transaction costs. The accuracy of our ensemble model is depicted for each stock for each prediction year in the table presented below. Predictions for all the period can be found in our Jupyter Notebook.

However, for years during the financial crisis, our portfolio didn't trade much since most of the signals were sell signals. Hence, we see almost constant portfolio value as shown in the graphs below for the years 2007 and 2008.

For the year 2005, our portfolio delivered slightly negative returns compared to S&P which delivered positive returns of 3.5 percent. For the year 2006, our portfolio declined

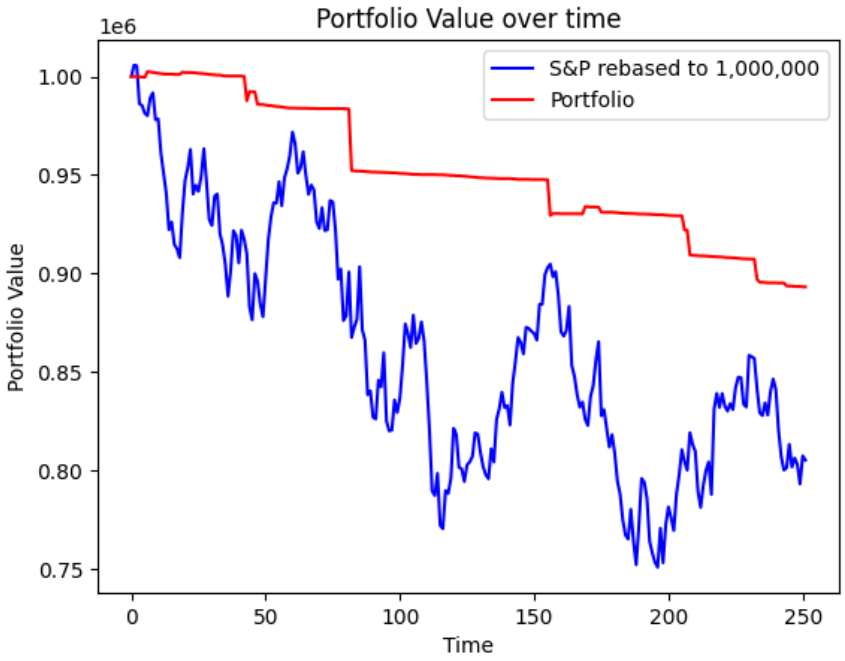


Figure 4: Portfolio performance vs S&P for year 2022.

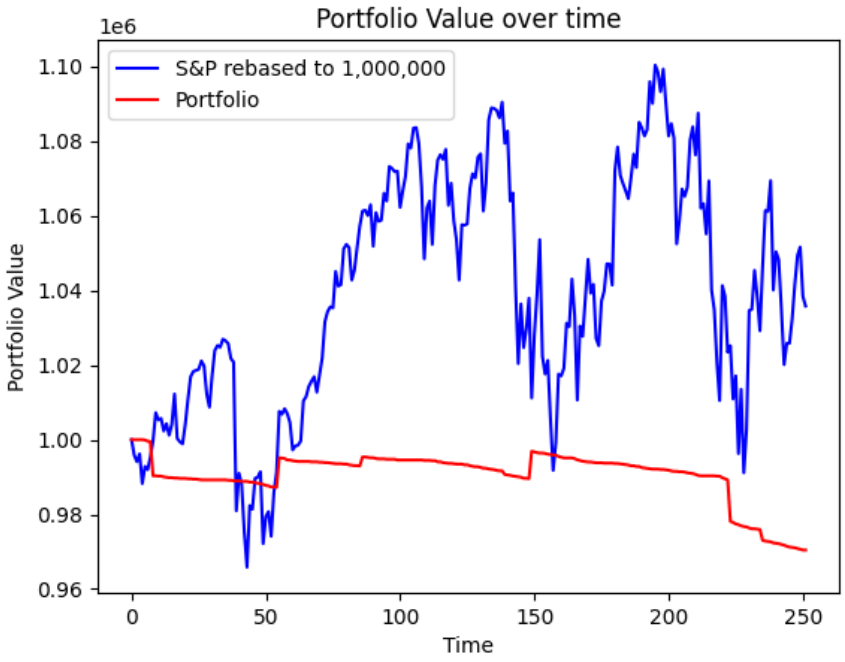


Figure 5: Portfolio performance vs S&P for year 2007.

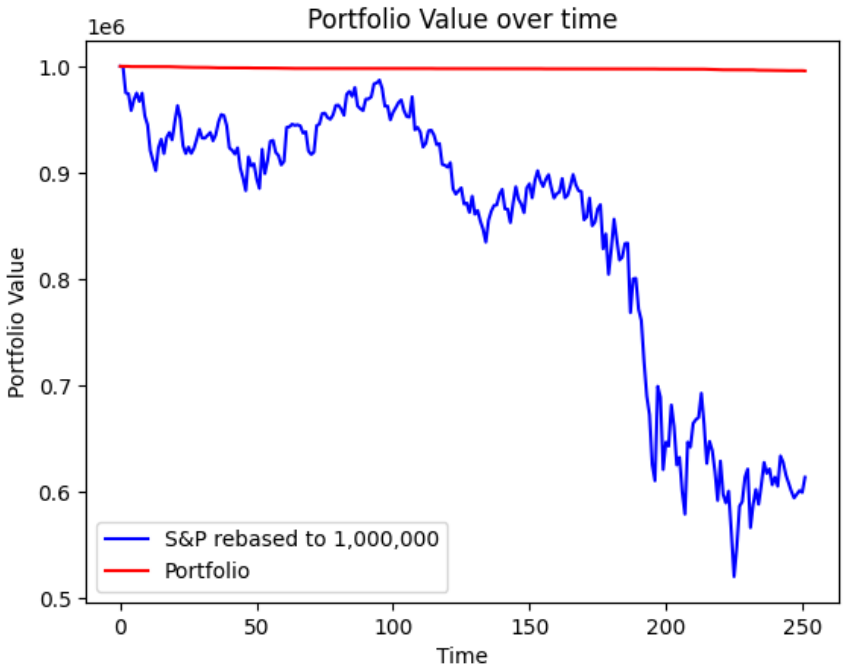


Figure 6: Portfolio performance vs S&P for year 2008.

marginally by 0.2 percent whereas the S&P grew 14 percent. For 2007 and 2008 we have already mentioned that our portfolio didn't trade due to mostly negative signals due to the financial crisis. For 2009 our portfolio delivered its first positive returns of 3.7 percent but was beaten by S&P by a huge margin which delivered 24.7 percent. For 2010, our portfolio delivered negative returns due to a lack of training data over the crisis period whereas S&P rose by 12.8 percent. Similarly, for 2011, our portfolio delivered poor and negative results compared to S&P due to poor training data during the financial crisis. The trend continued for the year 2012 as our training window consisted of three years, we observed negative returns whereas S&P grew with positive returns. For 2013, we maintained our portfolio value compared to S&P which grew at 29 percent. For the year 2014, our portfolio saw its value decline by 4.2 percent whereas S&P grew 12.4 percent. Year 2015 was the first time our portfolio beat the S&P by 3 percentage points as depicted in the graph below.

For 2016, we observe slightly positive returns but S&P beats our portfolio by 10 per-

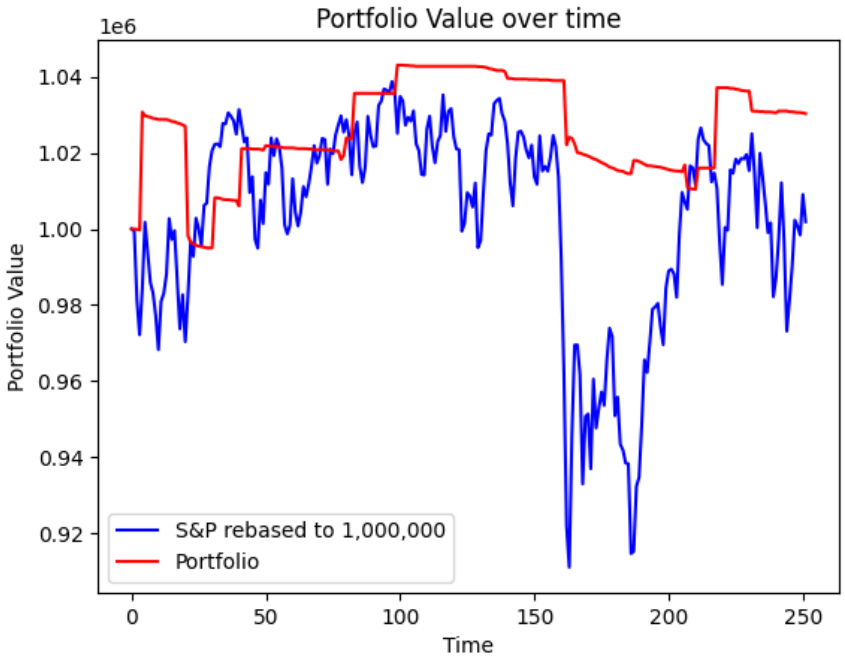


Figure 7: Portfolio performance vs S&P for year 2015.

centage points. For 2017, we maintained our portfolio whereas S&P grew at 19.4 percent. In 2018 we can beat the benchmark S&P by a large margin. Our portfolio delivered positive returns of 9.3 percent whereas S&P declined 7.5 percent as shown in the graph below.

For the years 2019 and 2020, our portfolio delivered negative 10 percent returns whereas S&P delivered positive 14.5 percent returns. For 2021 our portfolio declines slightly by 2 percent whereas S&P outperforms our portfolio by 29 percentage points.

The following table shows the accuracy of our ensemble prediction for each year. We observe a good performance of the ensemble over the years although it is difficult to beat the S&P. Further improvements to the model could be by adding more asset classes to choose from during periods of financial distress. These assets will also provide

Table 1: Stock Performance and Ensemble Accuracy in percentage (2005–2022)

	Stocks											
	MSFT	JPM	PFE	BRK-B	ORCL	AAPL	IBM	VZ	BAC	KO	NVDA	PG
2005	49.2	49.6	52.0	48.0	54.8	51.2	50.4	49.2	52.8	58.8	50.4	53.2
2006	48.4	50.0	51.6	52.0	49.2	44.8	46.4	49.6	50.8	54.4	56.0	52.4
2007	48.0	50.4	48.4	56.0	49.6	53.6	50.0	52.8	45.2	48.0	53.6	57.2
2008	54.0	55.2	54.4	53.6	53.2	49.2	49.6	54.4	54.8	49.6	49.6	56.0
2009	52.4	45.6	50.4	47.6	54.8	54.4	48.0	50.8	44.4	53.2	52.4	53.6
2010	48.8	51.6	57.2	52.4	44.8	50.4	48.4	46.8	56.4	50.0	51.6	55.2
2011	51.0	53.8	48.2	58.2	49.8	53.4	49.4	52.2	48.2	56.2	46.2	57.4
2012	46.0	53.2	52.0	52.0	51.6	54.0	44.8	52.4	52.4	45.2	52.8	58.0
2013	54.4	53.6	48.8	44.0	50.0	41.2	55.2	51.6	50.4	49.6	46.0	57.6
2014	48.8	52.0	45.6	52.0	52.0	48.0	46.8	51.6	46.4	52.4	49.2	55.6
2015	54.0	56.8	50.4	50.4	52.4	48.8	51.6	52.4	54.8	51.6	52.4	55.2
2016	51.2	47.2	50.8	53.6	48.4	50.8	49.2	50.8	51.6	42.0	49.2	55.6
2017	56.4	52.4	51.2	50.4	49.6	52.0	58.4	47.6	46.0	49.2	53.2	54.8
2018	48.4	48.4	50.4	51.2	52.0	53.2	43.6	49.2	46.0	52.8	50.4	54.0
2019	58.0	54.8	55.2	53.6	56.0	55.2	54.4	50.4	48.8	50.4	56.4	55.2
2020	55.2	44.4	45.2	55.2	47.6	58.4	48.0	48.4	50.0	56.4	51.2	57.6
2021	57.2	51.6	50.0	52.8	53.2	51.6	47.6	50.0	54.4	52.0	54.4	54.0
2022	49.6	49.2	47.2	50.0	47.2	48.8	50.0	52.4	50.8	45.6	46.4	58.0

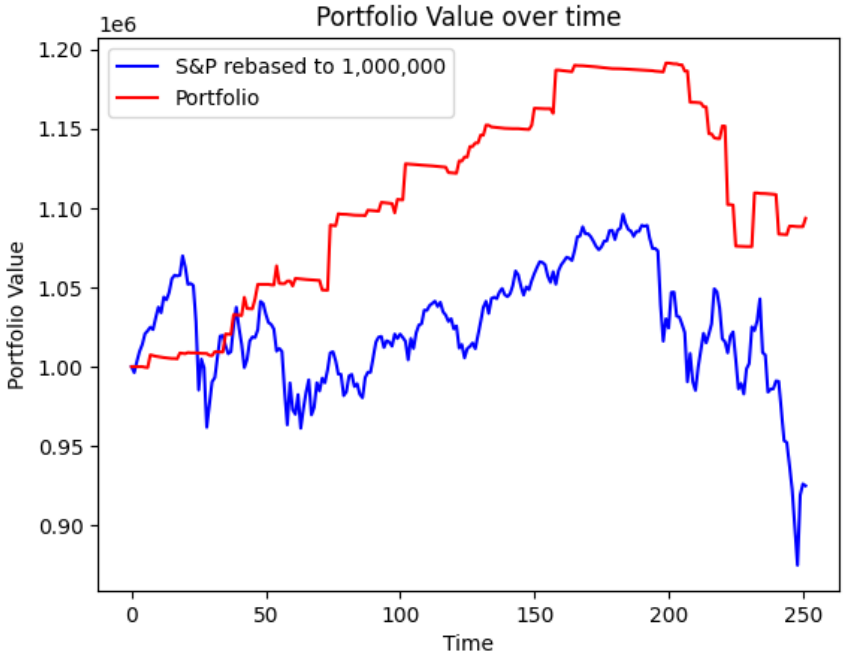


Figure 8: Portfolio performance vs S&P for year 2018.

6.1 Shortcomings of the current model

Our code and analyses has taken many assumptions that might affect the portfolio management either in real implementation.

Extremely Liquid Markets: We have assumed highly liquid markets that enable us to trade even fractions of stocks. This might not be the case in real life and would affect our actual portfolio weights. This assumption allows us to ascertain that our portfolio weights are in extreme conformity with the weights suggested by the Markowitz theorem. We have also assumed that each day, we trade with 90% of our capital. It might happen in the real scenario, that we are unable to exit our positions but the assumption of liquid markets helps us avoid the situation for the sake of this study.

Trading at closing prices: Our model calculates the outcome of trades using the closing values of the day. However, in the real scenario, we might not be able to execute and close

Table 2: Ensemble Accuracy by Year (2005–2022)

Year	Ensemble Accuracy (%)
2005	57.5
2006	54.8
2007	54.9
2008	56.9
2009	55.2
2010	56.2
2011	56.8
2012	55.9
2013	55.2
2014	55.7
2015	56.8
2016	56.1
2017	55.6
2018	56.1
2019	55.2
2020	55.9
2021	56.8
2022	56.0

our trades near the end of the day if our order is large.

Fixed Trading Fee: Our model assumes that the transaction costs are $100 per complete transaction for a stock (i.e., opening and closing of a trade). However, this might not be the case with all brokers. Some brokers take a flat fee and some take a percent of the transaction amount. So, it depends on the broker and the policies of the country/ exchange.

This might negatively affect the portfolio return should there be a high transaction volume. This leads to another aspect of portfolio management which is transaction cost analysis which has gained interest due to high-frequency trading since it results in high trading volumes [29].

Explainability: This analysis could have been more comprehensive and complete with the inclusion of XAI techniques to make the investors understand the rationale behind the trading decisions. However, given the limited time and other constraints, it was not possible in this area.

Potential other predictors: Besides the technical indicators, stock returns, prices, and volume; our model only takes into consideration S&P500 price, VIX index and 10-year US treasury bond yield. However, several other variables have been used to forecast prices. We could have used Gold, Brent Crude prices [30], Geopolitical Risk Index [31], Bitcoin price, etc. in our model. This might have resulted in certain improvements in portfolio performance.

Trading at fixed prices: Besides assuming that we can open and close our positions at closing prices, we have also assumed that we can buy and sell all of our stocks at the same price for a given order. However, that is not the case, as for a given order we might end up paying different prices for each unit of stock depending upon the order book [32].

No short selling: Perhaps, one of the biggest shortcomings of our model is that it ignores the short signals. For the sake of simplicity, we have assigned the minimum value for stock weights as 0.0 for our Markowitz optimizer. This leads to a lot of missed opportunities. Whenever our model is predicting that the price will go down, we are not utilising that information which is leading to lesser profits as compared to other funds that will be engaged in short selling. In the case of a Bearish regime, where most of the stocks, it is predicting a fall, we are executing no trades in the whole day. This will lead to our capital being eaten away by inflation. As we can see in a few plots our portfolio value remains constant for at times for longer periods. This can be explained by the bearish prediction for the periods.

However, this might not affect the portfolio performance for markets where there are bans on short-selling.

Market conditions-based performance: This is an outcome of the previous point. Since our model does not short-stock, we are not able to harness profits in bearish market conditions. This will lead to the degradation of our capital and poor performance of the portfolio.

No hyperparameter tuning: Due to a lack of state-of-the-art computational resources, we were not able to explore hyperparameter tuning for our models due to the significant computation time involved. However, hyperparameter tuning should be explored to build better models.

No-Feature Engineering: Due to a significant number of stocks under consideration, it is difficult to manually check and perform feature engineering and exploratory data analysis. With the availability of faster computing resources and manpower, this could have also been achieved.

No Risk Management strategy: As a good portfolio manager, we should have incorporated a risk management strategy employing VaR or Expected Shortfall. Having such a strategy could protect our portfolio against adverse market movements.

6.2 Associated Risks and steps taken to avoid them

- **Over fitting risk:** This is the risk that the AI model learns the training data too well, fits noise, and does not generalize well to new data. This can lead to poor performance of the portfolio on out-of-sample data. We have taken care of overfitting risk with ensemble learning as it uses several weak learners to make a better prediction. This is how we can tackle such risks associated with our model. Also, we have used dropouts in the case of our LSTM model, to avoid overfitting it.

- **Model deployment and monitoring:** Once an AI model is trained, it must be

deployed to production and monitored for performance. This can be a challenge, especially for models that are used to make critical decisions. Since we propose a 2-step approach, this enables manual intervention and assessment of results in an intermediate stage.

6.3 Challenges of this approach

- **Trust and transparency:** Investors need to trust that the AI systems used by financial institutions are making sound investment decisions. However, it can be difficult for investors to trust AI systems, as they are often complex and difficult to understand. This is why we have used a fairly common optimisation technique which is understood by a lot of investors. The use of Explainable AI techniques can further help us tackle this problem.

- **Regulatory compliance:** Financial institutions are subject to a variety of regulations. This can make it difficult to implement AI systems that comply with all applicable regulations. Again, our two-step approach helps us tackle this challenge.

7 Conclusions

Limitations, Future work and conclusion:

Our current research has 2 dimensions: Firstly, we tried to predict actual prices using our ensemble model. However, the results were erroneous and hence we pivoted our approach to predicting price movement direction and building a Markovitz portfolio using the signals generated from our prediction. Our Markovitz portfolio consists of only positive direction predicted stocks. Our models could be refined further and the use of alternate data (besides the current market and financial data) can increase the accuracy of our models.

However such changes will require significant computation time and resources. Researchers in academia are also focusing heavily on reinforcement learning for portfolio construction. Though, we had our reasons for using an ensemble model over an RL approach as explained earlier, explainable reinforcement learning could be a possible future work in this area [20]. Our work solely focuses on stocks, it could also be extended to a multi-asset optimization [24].

Later works could also incorporate the ability of Large Language models to make investment decisions. The work of Lira and Tang concludes that incorporation of LLM decisions can help improve the outcome of investment actions [27]. Steinert and Altmann investigated the potential improvement of the GPT-4 Language Learning Model (LLM) in comparison to BERT for modeling same-day daily stock price movements of Apple and Tesla in 2017, based on sentiment analysis of micro blogging messages and concluded that GPT-4 exhibited substantial accuracy, outperforming BERT in five out of six months and substantially exceeding a naive buy-and-hold strategy, reaching a peak accuracy of 71.47% in May 2019 [28]. So, it seems that the next leg of research in this domain would be incorporating LLMs.

8 Appendix

8.1 Questions

1. What is the primary objective of portfolio construction using Markowitz Sharpe Ratio?

   (a) Maximizing the number of assets in the portfolio

   (b) Minimizing the risk of individual assets in the portfolio

   (c) Achieving the highest possible return for a given level of risk

   (d) Maximizing the total market capitalization of the portfolio

Answer: (c) Achieving the highest possible return for a given level of risk

2. What are the advantages of using ensemble models for stock price prediction?

   (a) Increased computational complexity and longer training times

   (b) Reduced diversity in predictions leading to lower accuracy

   (c) Improved generalization and robustness by combining multiple models

   (d) Limited adaptability to changing market conditions

   Answer: (c) Improved generalization and robustness by combining multiple models

3. When comparing two-stage and one-stage ensemble learning approaches for stock price prediction, which statement is correct?

   (a) Two-stage ensembles involve a single model trained in one stage, while one-stage ensembles use multiple models sequentially.

   (b) One-stage ensembles combine diverse base models in a single stage, while two-stage ensembles involve stacking the predictions of multiple models in a second stage.

   (c) Two-stage ensembles are computationally more efficient than one-stage ensembles.

   (d) One-stage ensembles often suffer from overfitting, while two-stage ensembles provide better generalization.

   Answer: (b) One-stage ensembles combine diverse base models in a single stage, while two-stage ensembles involve stacking the predictions of multiple models in a second stage.

4. In the context of portfolio optimization using Markowitz Sharpe Ratio, how does the Sharpe Ratio help in decision-making?

(a) It focuses solely on maximizing returns without considering risk.

(b) It aims to minimize the total variance of the portfolio.

(c) It provides a measure of risk-adjusted return, guiding the selection of an optimal portfolio.

(d) It emphasizes equal distribution of assets for simplicity.

Answer: (c) It provides a measure of risk-adjusted return, guiding the selection of an optimal portfolio.

5. How can machine learning contribute to portfolio construction in the financial domain?

(a) By minimizing the use of quantitative data for decision-making.

(b) By introducing randomness to diversify portfolio holdings.

(c) By providing data-driven insights for asset selection and allocation.

(d) By prioritizing historical returns as the sole criterion for portfolio composition.

Answer: (c) By providing data-driven insights for asset selection and allocation.

6. In ensemble regression models, what are common methods for combining the results of individual regression models?

(a) Simple Averaging and Hard Voting

(b) Weighted Averaging and Stacking

(c) Bagging and Boosting

(d) Soft Voting and Stacking

Answer: (b) Weighted Averaging and Stacking

7. What is considered a key criterion when determining the best method for portfolio construction?

   (a) Using a method that maximizes the number of assets in the portfolio.

   (b) Selecting a method solely based on historical returns of individual assets.

   (c) Prioritizing a method that minimizes overall portfolio risk.

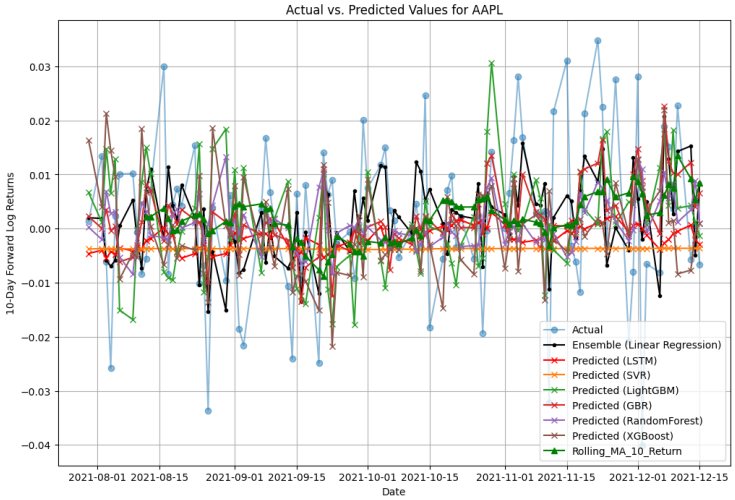
   (d) Opting for a method that equalizes the weights of assets for simplicity.

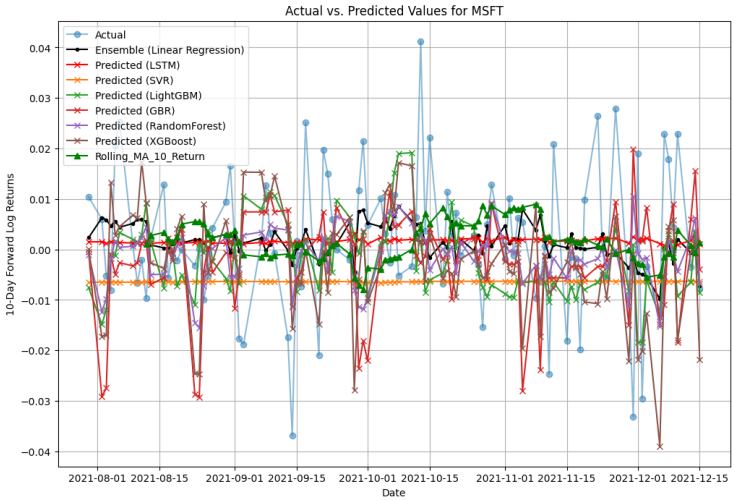
Answer: (c) Prioritizing a method that minimizes overall portfolio risk.

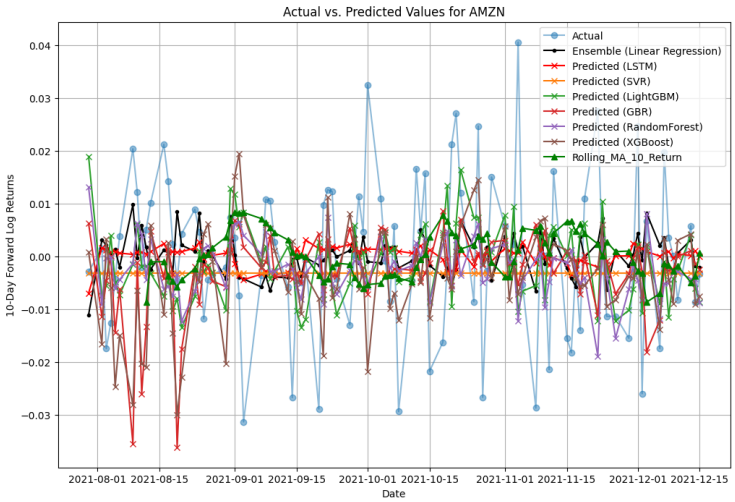
8.2 Regression results

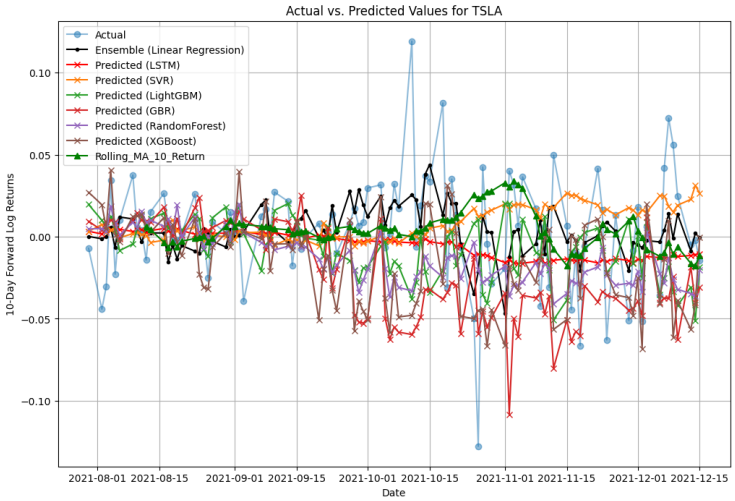
Table 3: Ensemble regression results for top 5 stocks

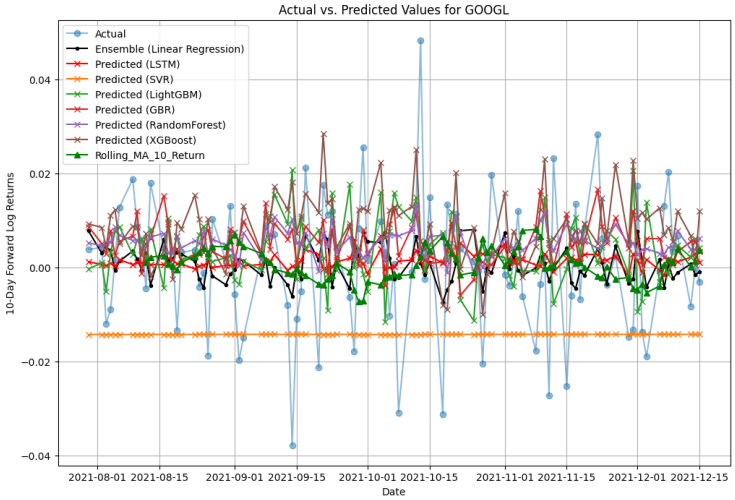
Stocks	MAE	RMSE	R2-Score
AAPL	0.0107	0.0139	0.0780
MSFT	0.0099	0.0132	0.0404
AMZN	0.0103	0.0135	0.1091
TSLA	0.0225	0.0311	0.0884
GOOGL	0.010	0.0135	0.0577

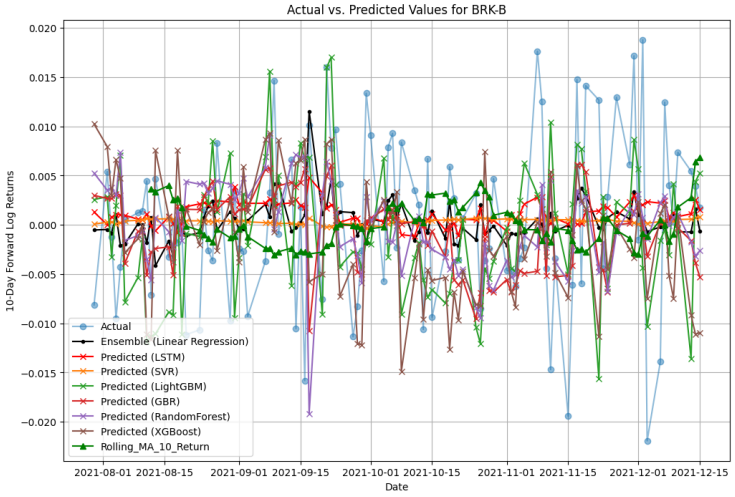

Actual vs. Predicted Values for AAPL
10-Day Forward Log Returns
Date
Actual
Ensemble (Linear Regression)
Predicted (LSTM)
Predicted (SVR)
Predicted (LightGBM)
Predicted (GBR)
Predicted (RandomForest)
Predicted (XGBoost)
Rolling_MA_10_Return

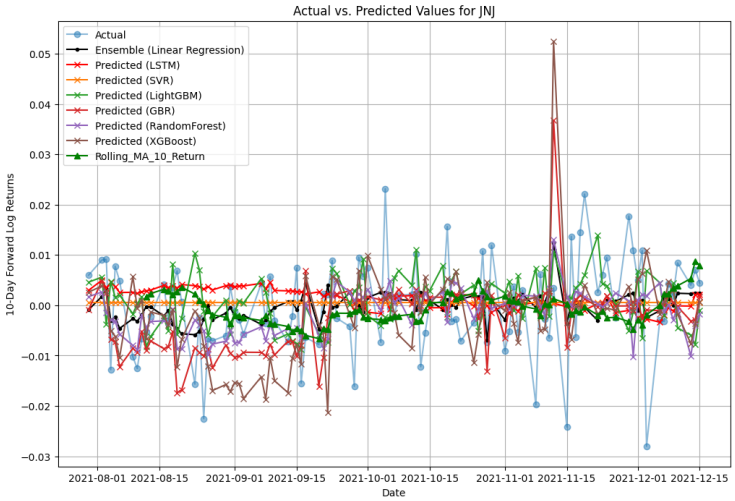

Actual vs. Predicted Values for MSFT
10-Day Forward Log Returns
Date
Actual
Ensemble (Linear Regression)
Predicted (LSTM)
Predicted (SVR)
Predicted (LightGBM)
Predicted (GBR)
Predicted (RandomForest)
Predicted (XGBoost)
Rolling_MA_10_Return

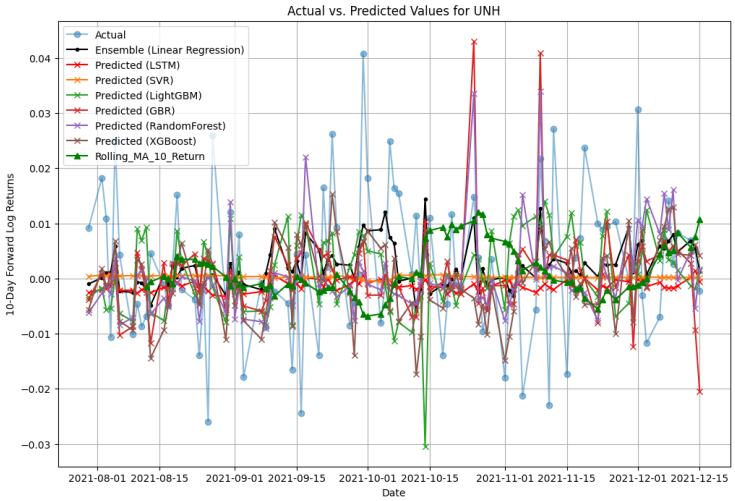

Actual vs. Predicted Values for AMZN
10-Day Forward Log Returns
Date
Actual
Ensemble (Linear Regression)
Predicted (LSTM)
Predicted (SVR)
Predicted (LightGBM)
Predicted (GBR)
Predicted (RandomForest)
Predicted (XGBoost)
Rolling_MA_10_Return

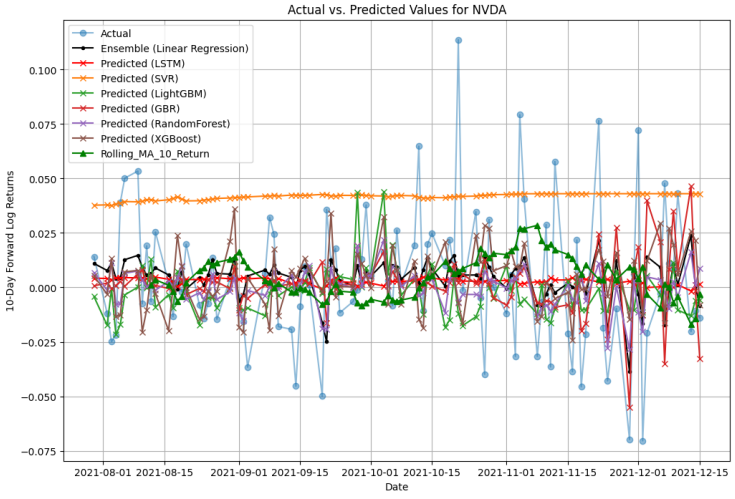

Actual vs. Predicted Values for TSLA
Actual
Ensemble (Linear Regression)
Predicted (LSTM)
Predicted (SVR)
Predicted (LightGBM)
Predicted (GBR)
Predicted (RandomForest)
Predicted (XGBoost)
Rolling_MA_10_Return
10-Day Forward Log Returns
0.10
0.05
0.00
−0.05
−0.10
2021-08-01 2021-08-15
2021-09-01 2021-09-15
2021-10-01 2021-10-15
2021-11-01 2021-11-15
2021-12-01 2021-12-15
Date

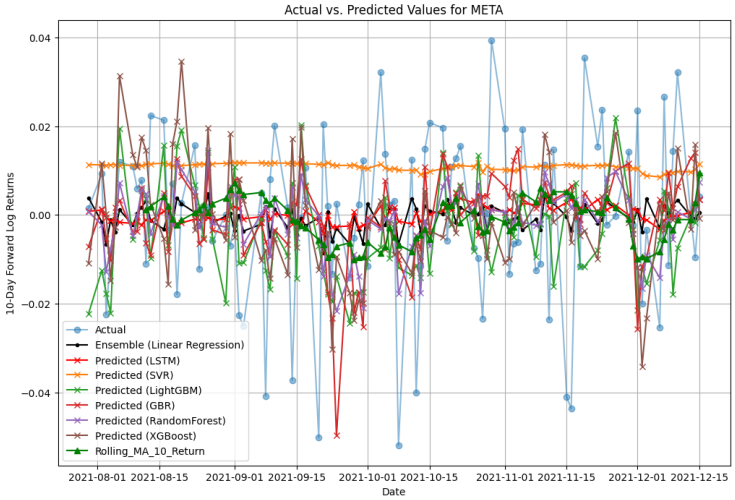

Actual vs. Predicted Values for GOOGL
Actual
Ensemble (Linear Regression)
Predicted (LSTM)
Predicted (SVR)
Predicted (LightGBM)
Predicted (GBR)
Predicted (RandomForest)
Predicted (XGBoost)
Rolling_MA_10_Return
10-Day Forward Log Returns
0.04
0.02
0.00
−0.02
−0.04
2021-08-01 2021-08-15
2021-09-01 2021-09-15
2021-10-01 2021-10-15
2021-11-01 2021-11-15
2021-12-01 2021-12-15
Date

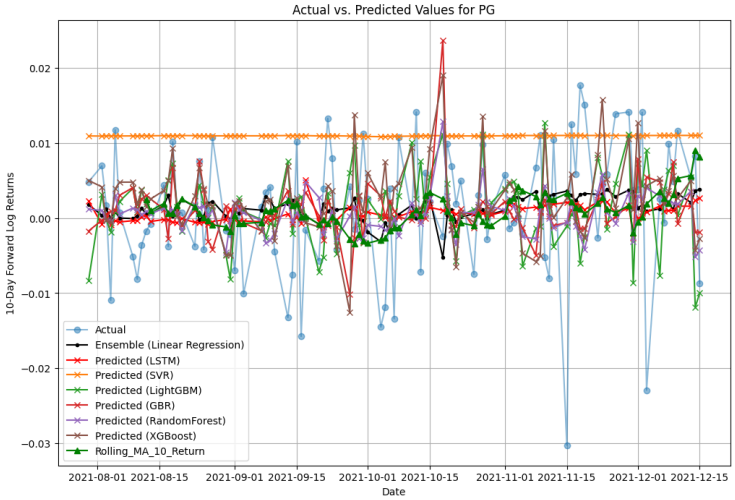

Actual vs. Predicted Values for BRK-B
Actual
Ensemble (Linear Regression)
Predicted (LSTM)
Predicted (SVR)
Predicted (LightGBM)
Predicted (GBR)
Predicted (RandomForest)
Predicted (XGBoost)
Rolling_MA_10_Return
10-Day Forward Log Returns
0.020
0.015
0.010
0.005
0.000
−0.005
−0.010
−0.015
−0.020
2021-08-01 2021-08-15
2021-09-01 2021-09-15
2021-10-01 2021-10-15
2021-11-01 2021-11-15
2021-12-01 2021-12-15
Date

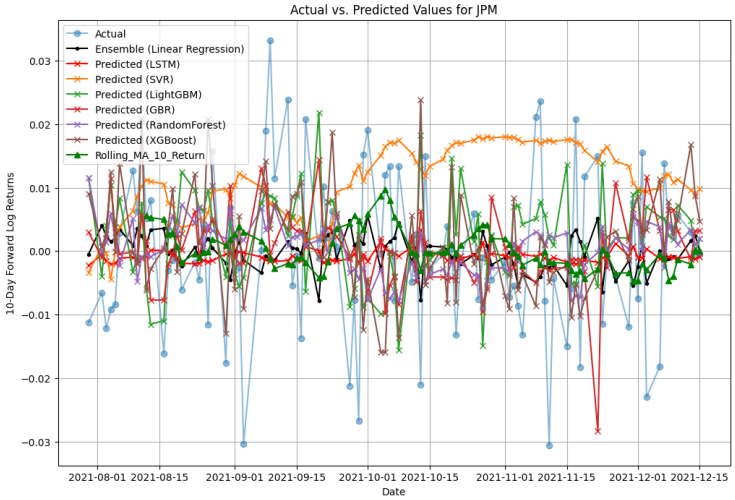

Actual vs. Predicted Values for JNJ
Actual
Ensemble (Linear Regression)
Predicted (LSTM)
Predicted (SVR)
Predicted (LightGBM)
Predicted (GBR)
Predicted (RandomForest)
Predicted (XGBoost)
Rolling_MA_10_Return
10-Day Forward Log Returns
Date

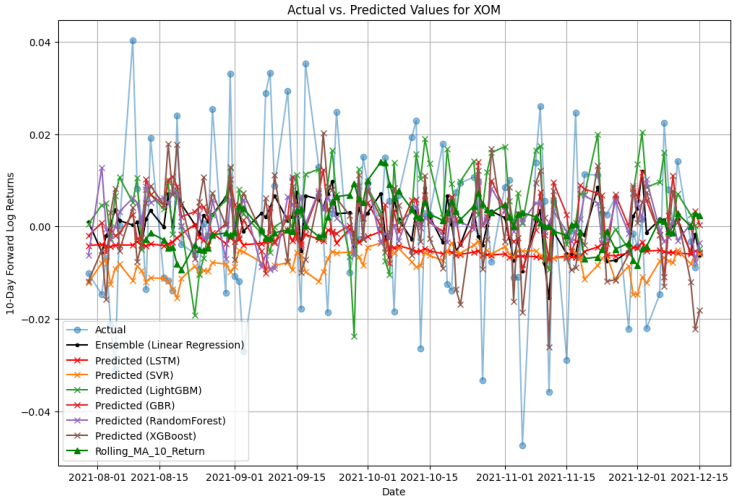

Actual vs. Predicted Values for UNH
Actual
Ensemble (Linear Regression)
Predicted (LSTM)
Predicted (SVR)
Predicted (LightGBM)
Predicted (GBR)
Predicted (RandomForest)
Predicted (XGBoost)
Rolling_MA_10_Return
10-Day Forward Log Returns
Date

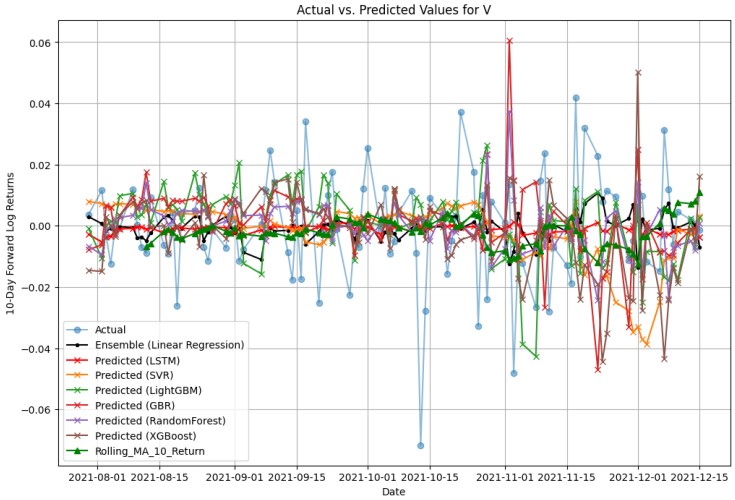

Actual vs. Predicted Values for NVDA
Actual
Ensemble (Linear Regression)
Predicted (LSTM)
Predicted (SVR)
Predicted (LightGBM)
Predicted (GBR)
Predicted (RandomForest)
Predicted (XGBoost)
Rolling_MA_10_Return
10-Day Forward Log Returns
Date

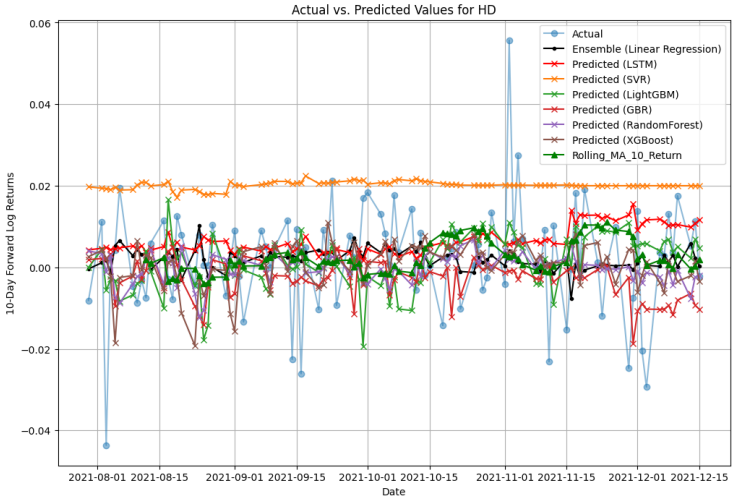

Actual vs. Predicted Values for META
10-Day Forward Log Returns
Date
Actual
Ensemble (Linear Regression)
Predicted (LSTM)
Predicted (SVR)
Predicted (LightGBM)
Predicted (GBR)
Predicted (RandomForest)
Predicted (XGBoost)
Rolling_MA_10_Return

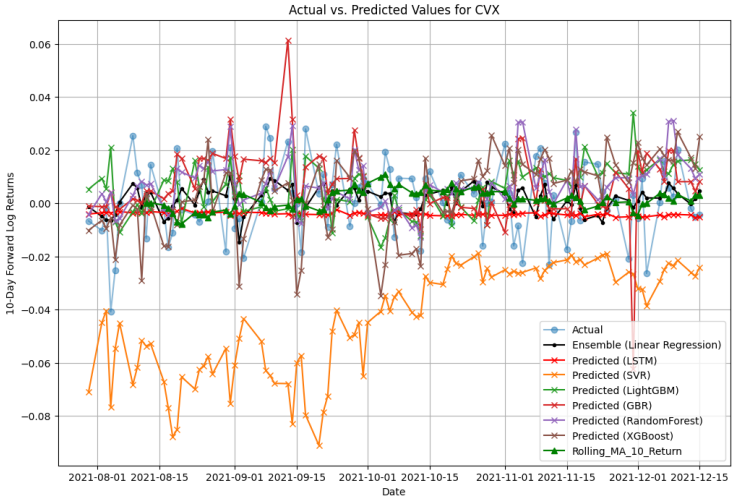

Actual vs. Predicted Values for PG
10-Day Forward Log Returns
Date
Actual
Ensemble (Linear Regression)
Predicted (LSTM)
Predicted (SVR)
Predicted (LightGBM)
Predicted (GBR)
Predicted (RandomForest)
Predicted (XGBoost)
Rolling_MA_10_Return

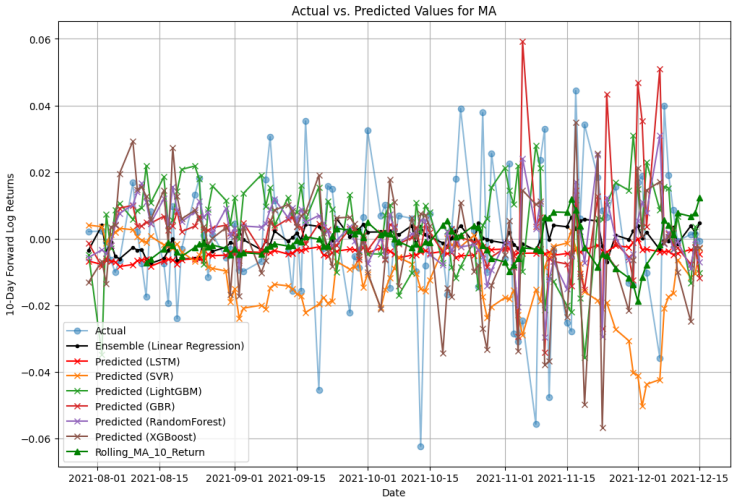

Actual vs. Predicted Values for JPM
10-Day Forward Log Returns
Date
Actual
Ensemble (Linear Regression)
Predicted (LSTM)
Predicted (SVR)
Predicted (LightGBM)
Predicted (GBR)
Predicted (RandomForest)
Predicted (XGBoost)
Rolling_MA_10_Return

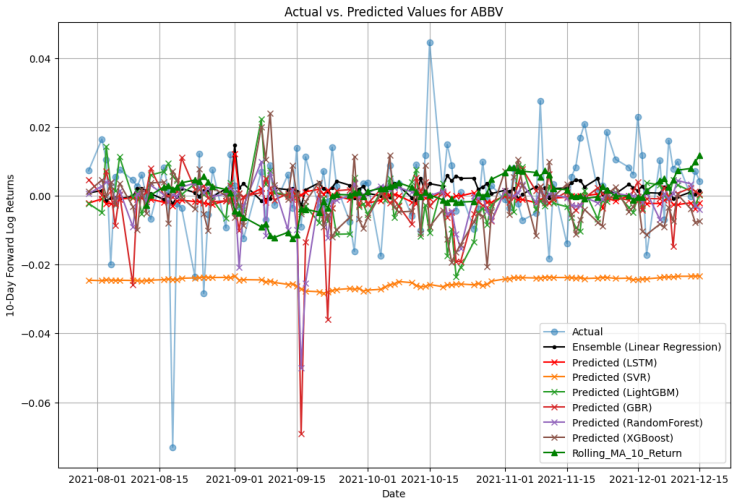

Actual vs. Predicted Values for XOM
10-Day Forward Log Returns
Date
Actual
Ensemble (Linear Regression)
Predicted (LSTM)
Predicted (SVR)
Predicted (LightGBM)
Predicted (GBR)
Predicted (RandomForest)
Predicted (XGBoost)
Rolling_MA_10_Return

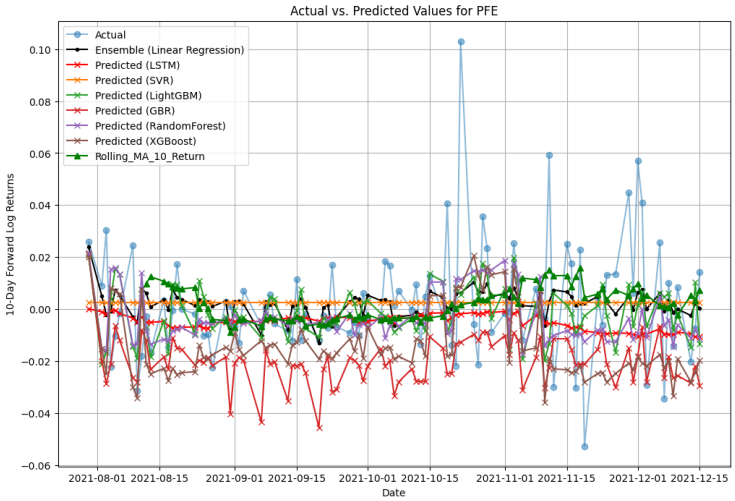

Actual vs. Predicted Values for V
10-Day Forward Log Returns
Date
Actual
Ensemble (Linear Regression)
Predicted (LSTM)
Predicted (SVR)
Predicted (LightGBM)
Predicted (GBR)
Predicted (RandomForest)
Predicted (XGBoost)
Rolling_MA_10_Return

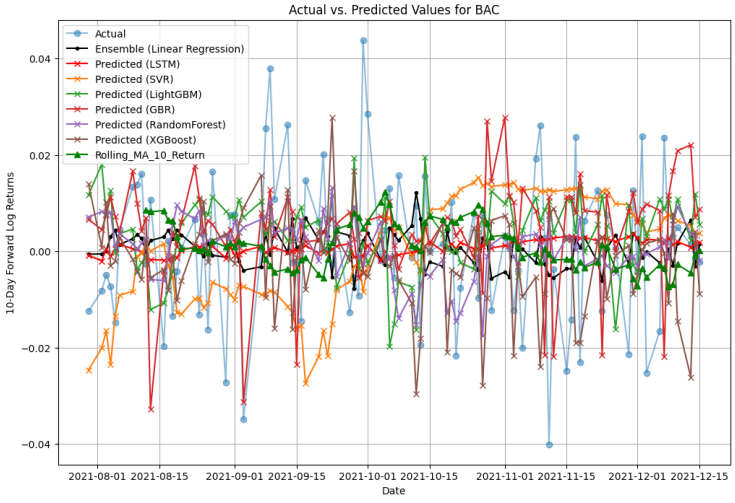

Actual vs. Predicted Values for HD
10-Day Forward Log Returns
Date
Actual
Ensemble (Linear Regression)
Predicted (LSTM)
Predicted (SVR)
Predicted (LightGBM)
Predicted (GBR)
Predicted (RandomForest)
Predicted (XGBoost)
Rolling_MA_10_Return


Actual vs. Predicted Values for CVX
10-Day Forward Log Returns
Date
Actual
Ensemble (Linear Regression)
Predicted (LSTM)
Predicted (SVR)
Predicted (LightGBM)
Predicted (GBR)
Predicted (RandomForest)
Predicted (XGBoost)
Rolling_MA_10_Return


Actual vs. Predicted Values for MA
10-Day Forward Log Returns
Date
Actual
Ensemble (Linear Regression)
Predicted (LSTM)
Predicted (SVR)
Predicted (LightGBM)
Predicted (GBR)
Predicted (RandomForest)
Predicted (XGBoost)
Rolling_MA_10_Return


Actual vs. Predicted Values for ABBV
10-Day Forward Log Returns
Date
Actual
Ensemble (Linear Regression)
Predicted (LSTM)
Predicted (SVR)
Predicted (LightGBM)
Predicted (GBR)
Predicted (RandomForest)
Predicted (XGBoost)
Rolling_MA_10_Return


Actual vs. Predicted Values for PFE
Actual
Ensemble (Linear Regression)
Predicted (LSTM)
Predicted (SVR)
Predicted (LightGBM)
Predicted (GBR)
Predicted (RandomForest)
Predicted (XGBoost)
Rolling_MA_10_Return
10-Day Forward Log Returns
0.10
0.08
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
2021-08-01 2021-08-15
2021-09-01 2021-09-15
2021-10-01 2021-10-15
2021-11-01 2021-11-15
2021-12-01 2021-12-15
Date


Actual vs. Predicted Values for BAC
Actual
Ensemble (Linear Regression)
Predicted (LSTM)
Predicted (SVR)
Predicted (LightGBM)
Predicted (GBR)
Predicted (RandomForest)
Predicted (XGBoost)
Rolling_MA_10_Return
10-Day Forward Log Returns
0.04
0.02
0.00
−0.02
−0.04
2021-08-01 2021-08-15
2021-09-01 2021-09-15
2021-10-01 2021-10-15
2021-11-01 2021-11-15
2021-12-01 2021-12-15
Date

8.3 Stock price direction prediction results

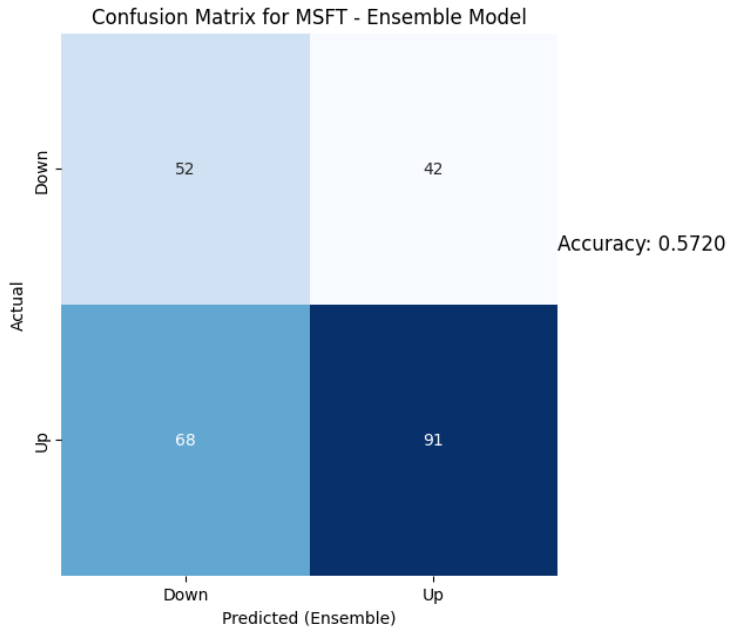


Figure 9: Confusion Matrix for MSFT in 2005

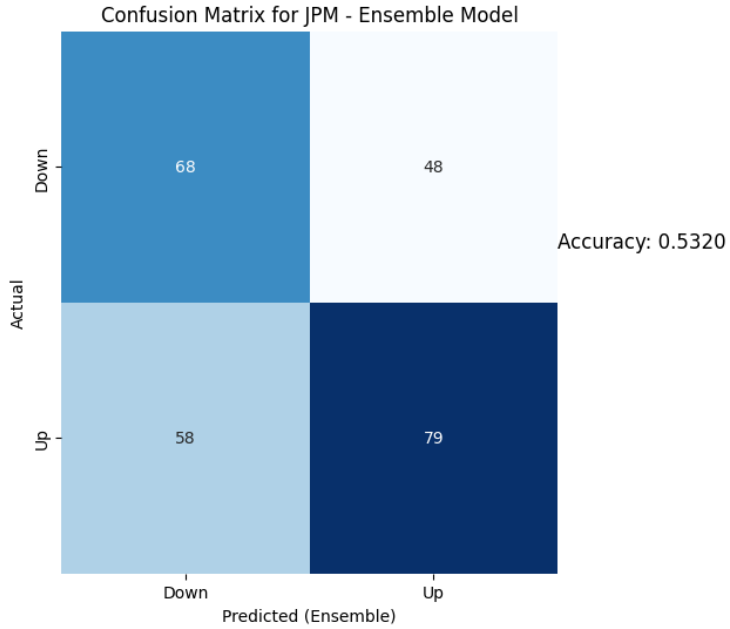


Figure 10: Confusion Matrix for JPM in 2005

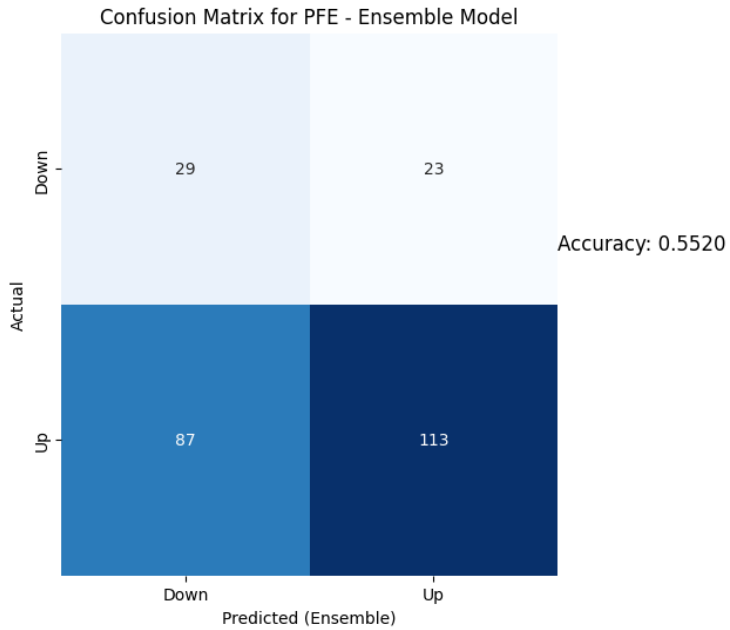


Figure 11: Confusion Matrix for PFE in 2019

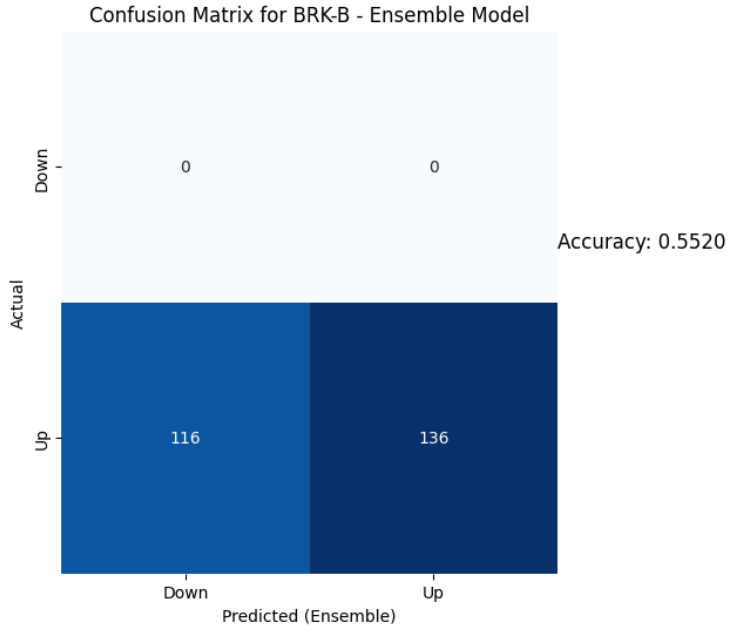


Figure 12: Confusion Matrix for BRK-B in 2005

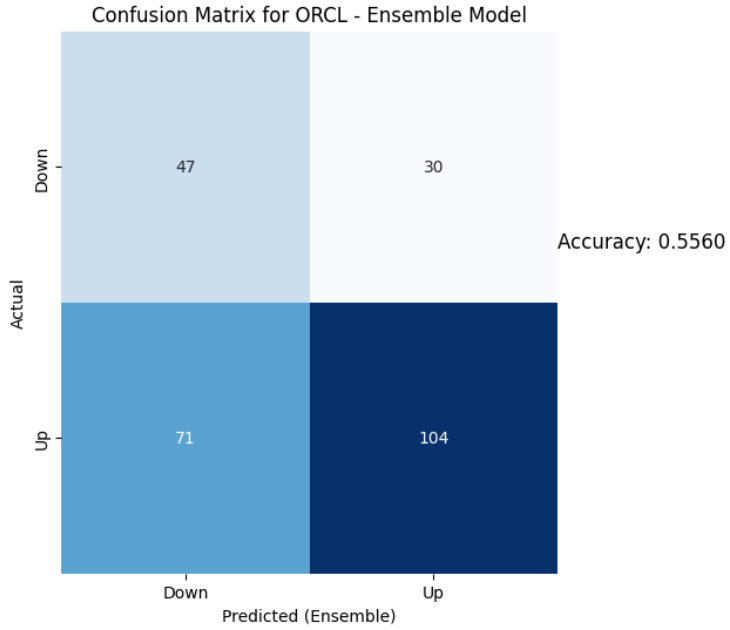


Figure 13: Confusion Matrix for ORCL in 2005

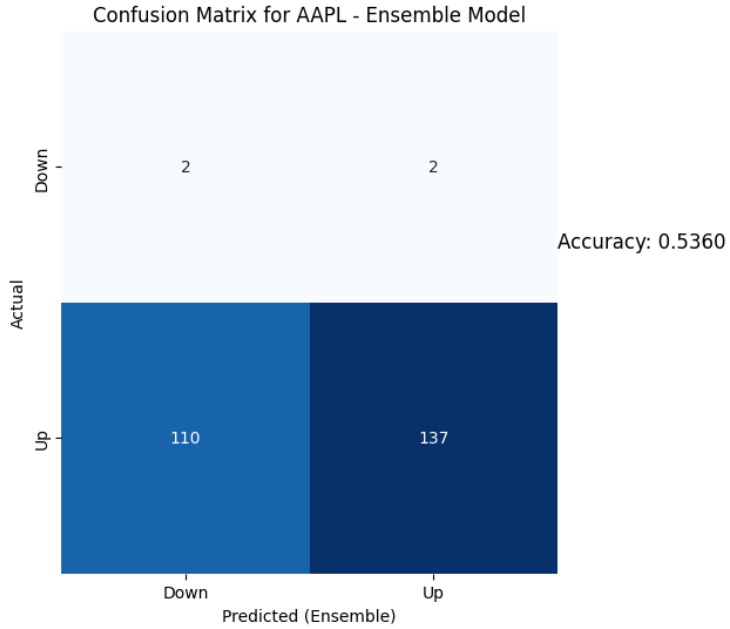


Figure 14: Confusion Matrix for AAPL in 2007

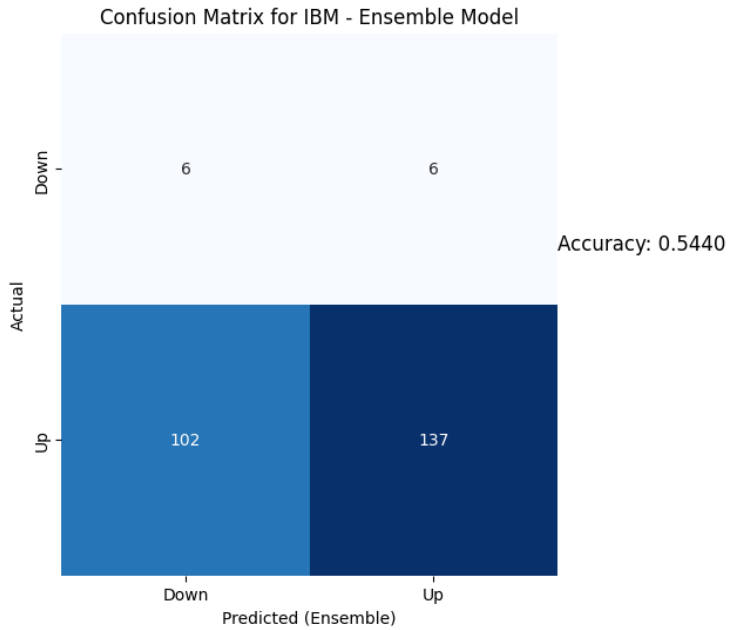


Figure 15: Confusion Matrix for IBM in 2019

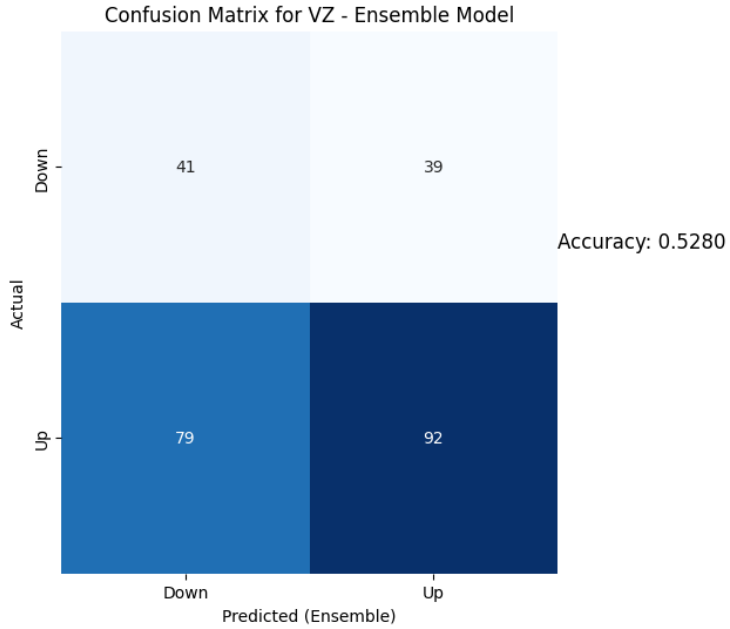


Figure 16: Confusion Matrix for VZ in 2007

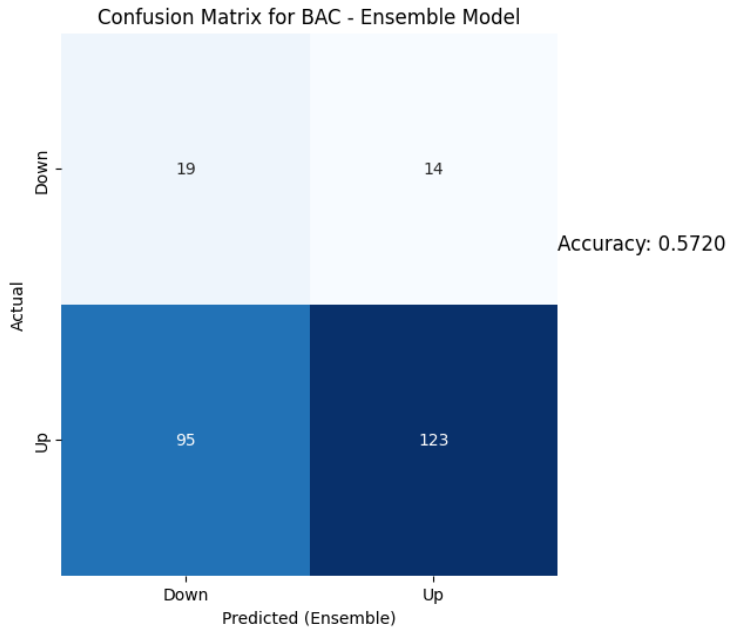


Figure 17: Confusion Matrix for BAC in 2005

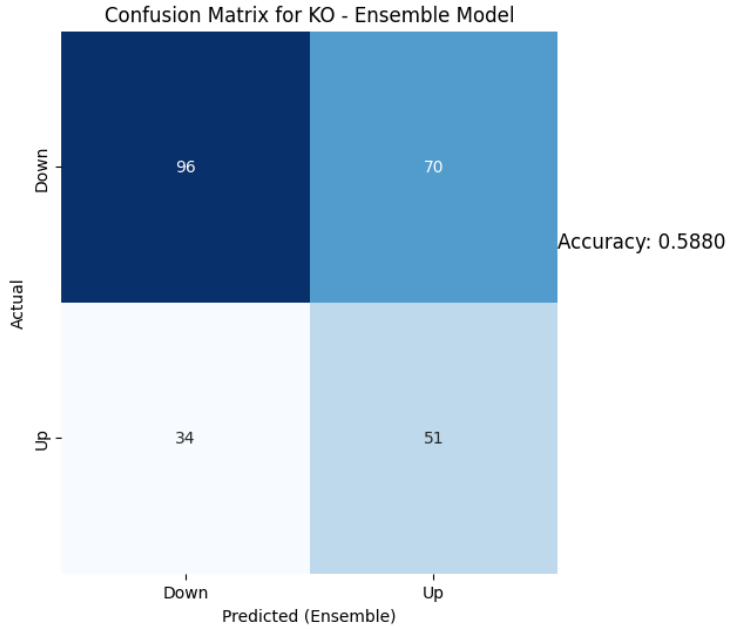


Figure 18: Confusion Matrix for KO in 2005

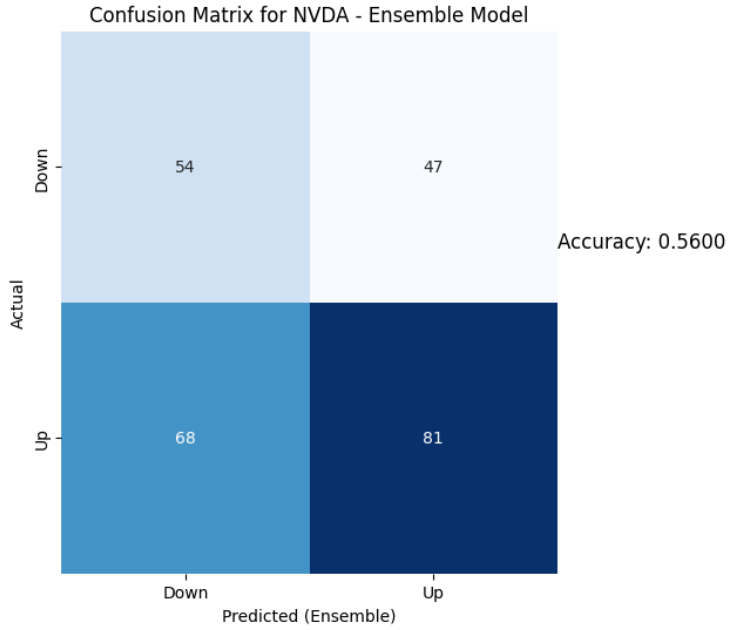


Figure 19: Confusion Matrix for NVDA in 2007

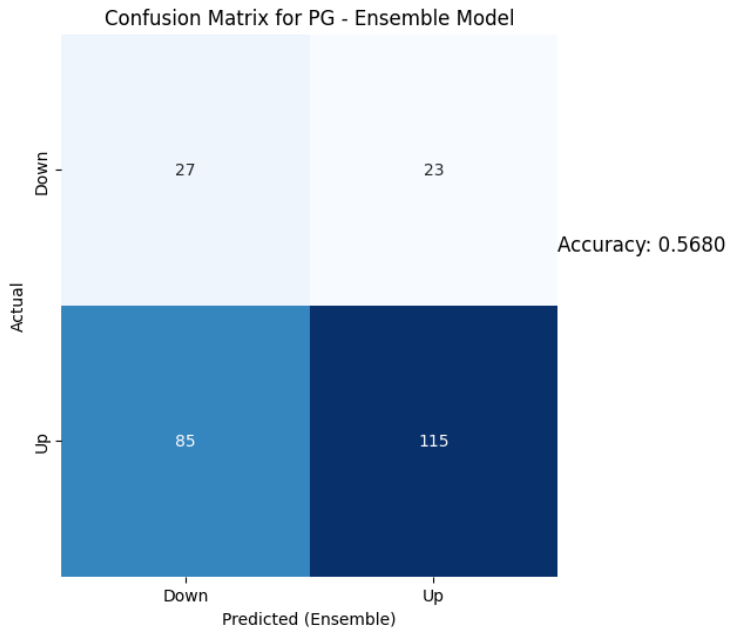


Figure 20: Confusion Matrix for PG in 2005

9 References

References

[1] Gates, B. (2023, March 21). The age of ai has begun. gatesnotes.com. https://www.gatesnotes.com/The-Age-of-AI-Has-Begun

[2] Goodell, J. W., Kumar, S., Lim, W. M., Pattnaik, D. (2021). Artificial intelligence and machine learning in finance: Identifying foundations, themes, and research clusters from bibliometric analysis. Journal of Behavioral and Experimental Finance, 32, 100577.

[3] "Machine Learning: Why Finance Is Different." AQR Capital Management, 7 June 2019, www.aqr.com/Learning-Center/Machine-Learning/Machine-Learning-Why-Finance-Is-Different.

[4] Zhou, Z., Song, Z., Ren, T., & Yu, L. (2023). Two-Stage Portfolio Optimization Integrating Optimal Sharp Ratio Measure and Ensemble Learning. In IEEE Access (Vol. 11, pp. 1654–1670). Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.1109/access.2022.3232281

[5] Ismail, M. S., Noorani, M. S. M., Ismail, M., Razak, F. A., Alias, M. A. (2020). Predicting next day direction of stock price movement using machine learning methods with persistent homology: Evidence from Kuala Lumpur Stock Exchange. Applied Soft Computing, 93, 106422.

[6] Ballings, M., Van den Poel, D., Hespeels, N., Gryp, R. (2015). Evaluating multiple classifiers for stock price direction prediction. Expert systems with Applications, 42(20), 7046-7056.

[7] Ansary, M. S., Brinto, A. (2022, December). Prediction of Stock Price Direction with Trading Indicators using Machine Learning Techniques. In 2022 IEEE International

Women in Engineering (WIE) Conference on Electrical and Computer Engineering (WIECON-ECE) (pp. 1-4). IEEE.

[8] L. Mason, J. Baxter, P. Bartlett, and M. Frean, "Boosting algorithms as gradient descent in function space," in Proc. Adv. Neural Inf. Process. Syst., 1999, pp. 1–30.

[9] L. Breiman, "Random forests," Mach. Learn., vol. 45, no. 1, pp. 5–32, Oct. 2001, doi: 10.1023/A:1010933404324.

[10] C.-J. Lu, T.-S. Lee, and C.-C. Chiu, "Financial time series forecasting using independent component analysis and support vector regression," Decis. Support Syst., vol. 47, no. 2, pp. 115–125, May 2009, doi: 10.1016/j.dss.2009.02.001.

[11] J. Patel, S. Shah, P. Thakkar, and K. Kotecha, "Predicting stock market index using fusion of machine learning techniques," Exp. Syst. Appl., vol. 42, no. 4, pp. 2162–2172, Mar. 2015, doi: 10.1016/j.eswa.2014.10.031.

[12] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Aug. 2016, pp. 785–794, doi: 10.1145/2939672.2939785.

[13] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T. Y. Liu, "LightGBM: A highly efficient gradient boosting decision tree," in Proc. Adv. Neural Inf. Process. Syst. (NIPS), vol. 30, 2017, pp. 1–9. Available: `https://www.webofscience.com/wos/alldb/fullrecord/WOS:000452649403021`

[14] A. L. D. Loureiro, V. L. Miguéis, and L. F. M. Da Silva, "Exploring the use of deep neural networks for sales forecasting in fashion retail," Decis. Support Syst., vol. 114, pp. 81–93, Oct. 2018, doi: 10.1016/j.dss.2018.08.010.

[15] M. Liu, K. Luo, J. Zhang, and S. Chen, “A stock selection algorithm hybridizing grey wolf optimizer and support vector regression,” Exp. Syst. Appl., vol. 179, Oct. 2021, Art. no. 115078, doi: 10.1016/j.eswa.2021.115078.

[16] Z. Zhou, Z. Song, T. Ren and L. Yu, ”Two-Stage Portfolio Optimization Integrating Optimal Sharp Ratio Measure and Ensemble Learning,” in IEEE Access, vol. 11, pp. 1654-1670, 2023, doi: 10.1109/ACCESS.2022.3232281.

[17] T. Sidogi, R. Mbuvha and T. Marwala, ”Stock Price Prediction Using Sentiment Analysis,” 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Melbourne, Australia, 2021, pp. 46-51, doi: 10.1109/SMC52423.2021.9659283.

[18] Ma, F., Lu, F., Tao, Y. (2022). Geopolitical risk and excess stock returns predictability: New evidence from a century of data. Finance Research Letters, 50, 103211.

[19] Nofer, M., Hinz, O. (2015). Using twitter to predict the stock market: Where is the mood effect?. Business Information Systems Engineering, 57, 229-242.

[20] Vouros, G. A. (2022). Explainable deep reinforcement learning: state of the art and challenges. ACM Computing Surveys, 55(5), 1-39.

[21] Aziz, S., Dowling, M. (2019). Machine learning and AI for risk management. Disrupting finance: FinTech and strategy in the 21st century, 33-50.

[22] Zhang, Q. T., Li, B., Xie, D. (2022). Machine Learning and AI in Financial Portfolio Management. In Alternative Data and Artificial Intelligence Techniques: Applications in Investment and Risk Management (pp. 33-74). Cham: Springer International Publishing.

[23] Mirete-Ferrer, P. M., Garcia-Garcia, A., Baixauli-Soler, J. S., Prats, M. A. (2022). A Review on Machine Learning for Asset Management. Risks, 10(4), 84.

[24] Atkinson, C., Mokkhavesa, S. (2004). Multi-asset portfolio optimization with transaction cost. Applied Mathematical Finance, 11(2), 95-123.

[25] Dhokane, R. M., Agarwal, S. (2023). Stock market prediction using the LSTM algorithm in association with the Relative Strength Index (RSI) and Exponential Moving Average (EMA) indicators.

[26] Gunasekarage, A., Power, D. M. (2001). The profitability of moving average trading rules in South Asian stock markets. Emerging markets review, 2(1), 17-33.

[27] Lopez-Lira, A., Tang, Y. (2023). Can chatgpt forecast stock price movements? return predictability and large language models. arXiv preprint arXiv:2304.07619.

[28] Steinert, R., Altmann, S. (2023). Linking microblogging sentiments to stock price movement: An application of GPT-4. arXiv preprint arXiv:2308.16771.

[29] Antonopoulos, D. D. (2016). Algorithmic Trading and Transaction Costs (Doctoral dissertation, Athens University of Economics and Business).

[30] Chen, Y. C., Huang, W. C. (2021). Constructing a stock-price forecast CNN model with gold and crude oil indicators. Applied Soft Computing, 112, 107760.

[31] Salisu, A. A., Lasisi, L., Tchankam, J. P. (2022). Historical geopolitical risk and the behaviour of stock returns in advanced economies. The European Journal of Finance, 28(9), 889-906.

[32] Understanding the order book: How it impacts trading. (2023). Retrieved from https://www.simtrade.fr/blog$_s imtrade/understanding-the-order-book-how-it-impacts-trading/$.

[33] Markowitz, H. M. (1991). Foundations of portfolio theory. The journal of finance, 46(2), 469-477.

[34] Sadorsky, Perry. 2021. A Random Forests Approach to Predicting Clean Energy Stock Prices. Journal of Risk and Financial Management 14: 48. https://doi.org/10.3390/jrfm14020048

[35] Khaidem, L. (2016, April 29). Predicting the direction of stock market prices using random forest. arXiv.org. https://arxiv.org/abs/1605.00003

[36] Tan, Z., Yan, Z., Zhu, G. (2019, August 1). Stock selection with random forest: An exploitation of excess return in the Chinese stock market. Heliyon. https://doi.org/10.1016/j.heliyon.2019.e02310

[37] Saini, A. (2022, August 26). An Introduction to Random Forest Algorithm for beginners. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/10/an-introduction-to-random-forest-algorithm-for-beginners/

[38] Gupta, A. (2022, January 27). Comprehensive Guide to Decision Tree Learning for Classification. Medium. https://therised.medium.com/comprehensive-guide-to-decision-tree-learning-for-classification-61bf2c510c6b

[39] Tailor V M. Exploiting LightGBM Ensemble Method for Stock Prediction[J]. International Journal of Scientific and Engineering Research, 2020, 11(10):648–650

[40] Minna Castoe. (n.d.). Predicting Stock Market Price Direction with Uncertainty Using Quantile Regression Forest. U.U.D.M. Project Report

[41] LightGBM: A Highly Efficient Gradient Boosting Decision Tree. Microsoft Research.

[42] Shekhar, S. (2023, June 14). What is LSTM for Text Classification? Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/06/lstm-for-text- classification/: :text=Inccuracy

[43] M. Roondiwala, H. Patel and S. Varma, "Predicting stock prices using LSTM,"

[44] Fischer, T., Krauss, C. (2018). Deep learning with long short-term memory networks