Erasmus University Rotterdam
Erasmus School of Economics
Bachelor Thesis Econometrics & Operations Research

Binary & Non-Binary Random Forest & Support Vector Machine for Commodity Price Prediction

Ka Wah Man (560190)

Abstract

In this study, we explore the effectiveness of using Random Forest and Support Vector Machine (SVM) models in contrast to a traditional OLS regression model for predicting commodity price movements. We highlight the implications of binary and non-binary input encoding in these models, which are built using 110 binary and 37 non-binary indicators and tested on monthly commodity data from 1982 to 2023. Despite a higher overall predictive accuracy (57.52%), the OLS regression model tends to overestimate upward price movements and is sensitive to market fluctuations. Conversely, the Random Forest and SVM models exhibit better accuracy in predicting upward trends and overall stability, albeit with slightly lower overall accuracies (56.33% and 54.91% respectively). These results underscore a trade-off between performance and stability. Our study further indicates that short-term indicators, especially the moving average rule, are important in predicting commodity movements regardless of encoding. Curiously, binary indicators may provide an edge in modeling highly volatile commodities. We also note that the non-binary OLS regression model shows significantly superior predictability compared to other models. Although binary inputs slightly improve machine learning models' accuracy, there's no statistical evidence to back this claim. This research showcases the relevance of machine learning models and technical indicator encoding in commodity price prediction and may aid in developing more nuanced trading strategies.

Supervisor:	Dick van Dijk
Second assessor:	Terri van der Zwan
Date final version:	2nd July 2023

1 Introduction

The global economy is intricately intertwined with the movements of commodity prices, making their accurate prediction a challenge for investors, traders and policymakers (Caballero, Farhi & Gourinchas, 2008). Over the years, traditional forecasting models based on technical indicators have played a crucial role in predicting the movement of commodity prices. However, the unprecedented events of the 2020 COVID-19 pandemic introduced an unparalleled level of volatility and uncertainty into the market, significantly challenging conventional forecasting approaches. The pandemic led to global disruptions in the supply chain, reduced demand and sharp fluctuations in commodity prices (Bakas & Triantafyllou, 2020). These extraordinary circumstances highlight the need for innovative and adaptive techniques that are able to capture the complex dynamics in commodity markets.

In this paper, we explore the effectiveness of using Random Forest and Support Vector Machine (SVM) models in contrast with a traditional OLS regression model for predicting the up-or-down movement of commodity prices using a multitude of binary and non-binary encoded technical indicators. We use monthly observations of eight commodity prices, starting from January 1982 to January 2023 obtained from World Bank (2023). As a benchmark model, we use the OLS regression model and binary technical indicators as proposed by Wang, Liu and Wu (2020). We introduce a non-binary variant on these indicators and use both original binary and non-binary input for their OLS regression model and extend upon their method by also using these inputs for a Random Forest and an SVM model.

While the widely known Efficient Market Hypothesis asserts that it is impossible to predict asset prices, with Fama (1970) stating that asset prices reflect all available information thus concluding that technical analysis should render fruitless. However, this hypothesis relies on traditional linear econometric models like ARIMA, VAR and GARCH. With advancements in technology and the ever-growing abundance of data, there has been an increased interest in using machine learning to aid in better predictions. In contrast to traditional models, machine learning models allow us to capture non-linear relations and complex interactions between indicators (Aminimehr, Raoofi, Aminimehr & Aminimehr, 2022).

The Random Forest model (Breiman, 2001) is an ensemble learning method that combines multiple decision trees to make predictions. Each individual decision tree in a forest trains on a random subset of the data and the predictions of each tree are aggregated to obtain the final prediction. The algorithm works by splitting the data based on various features and creating a decision rule at each node of the trees. This approach reduces overfitting and improves generalization. Random Forests are known for their ability to handle high-dimensional data and provide insights into feature importance. The SVM (Cortes & Vapnik, 1995) is a supervised learning method that aims to find an optimal hyperplane that separates the up and down observations in the data. This algorithm aims to maximize the distance between this hyperplane and the nearest data points from each class, this is known as the margin. By maximizing the margin, the SVM tries to find the best decision boundary that will generalize well to unseen data. It transforms the input space using a kernel function to project the data into a higher-dimensional feature space. SVMs are known to be effective in high-dimensional input space.

We find that the traditional OLS regression model with an average of 57.22% accuracy

significantly outperforms the Random Forest (56.33%) and SVM (54.91%). While the OLS obtains the highest average accuracy, it is more sensitive to market movement and overestimates upward price movement. The Random Forest and SVM exhibit higher accuracy in predicting upward movement, appear more stable and are less sensitive to economic changes, with a slight decrease in overall accuracy. This points to a trade-off between performance and stability. There is no evidence of a superior encoding for predictive accuracy, although we find that the non-binary OLS significantly outperforms most models. Although binary input seems to marginally increase the accuracy of the Random forest and SVM, there is no statistical evidence for this. Lastly, non-binary encoding leads to a slight increase in Sharpe ratio compared to binary encoding for the OLS (0.465 versus 0.450) and Random Forest (0.264 versus 0.258), but we find the opposite for the SVM (-0.041 versus 0.143). This does not take into account transaction costs, however, which range from 0.0004% to 0.033% in the futures market (Locke & Venkatesh, 1997), which may impact profit.

The outline of the paper is as follows. Section 2 discusses relevant literature. Section 3 describes the commodity prices and the binary and non-binary technical indicators. We cover the proposed Random Forest, SVM models and evaluation methods in Section 4 and discuss and analyze their results in Section 5. Finally, we conclude the paper in 6.

2 Literature Review

There are numerous studies on forecasting commodity prices, primarily using traditional econometric models such as ARIMA, VAR and GARCH (Deaton & Laroque, 1992). These models rely on assumptions that are often at odds with the behavior of the real market (Pindyck & Rotemberg, 1990). They depend on the belief that the time series is linear, that the mean and variance of the price remain constant over time and that variables are independent and normally distributed (Engle, 1982). These premises are overly simplistic and fail to capture the complex dynamics of the real world (Kwiatkowski, Phillips, Schmidt & Shin, 1992; Kilian & Murphy, 2014). Many researchers have already rejected this idea by using algorithms that can model more intricate dynamics of the financial system more effectively (Lo, Mamaysky & Wang, 2000; Bhattacharjee & Bhattacharja, 2019).

In recent years, there has been an increased interest in using machine learning for asset price prediction (Gu, Kelly & Xiu, 2020; Sezer, Gudelek & Ozbayoglu, 2020). Existing studies compare the performance of different machine learning models with Neural Networks, Random Forests and SVMs being popular choices (Sezer et al., 2020; Nabipour et al., 2020). However, these studies yield inconsistent results concerning which model achieves the highest accuracy and also do not consider financial applications (Kumar & M., 2006; Ballings, Van Den Poel, Hespeels & Gryp, 2015). Predictive accuracy may not necessarily translate into financial gain or optimal risk-adjusted returns, so we also investigate the Sharpe ratio of our models' predictions. Contrary to most studies that use a conservatively predetermined set of dozen technical indicators in fear of the dimensionality curse (Dash & Dash, 2016; Choudhry & Garg, 2008), our work looks at a wider range employing 110 binary indicators and 37 non-binary indicators, making use of feature selection to navigate the high-dimensional data efficiently.

When examining the encoding of various indicators, it becomes apparent that most research-

ers utilize continuous technical indicators (Dash & Dash, 2016; Bhattacharjee & Bhattacharja, 2019). However, there is a lack of extensive research conducted in this specific field, which limits the understanding of alternative methods. One such attempt was made by Patel, Shah, Thakkar and Kotecha (2015), who explored the use of binary input in various machine learning models. In comparison to non-binary inputs, Patel et al. (2015) suggests that binary input can provide superior performance, reporting an accuracy rate of 88%, as opposed to 80% for non-binary cases. Nonetheless, given the high accuracy reported, the findings warrant careful scrutiny and further validation, especially given the limited body of research on this particular topic.

To our knowledge, this is the first paper that compares different machine learning models, using a greater set of technical indicators with binary and non-binary encoding. Addressing this gap, this study aims to shed light on the role of encoding and its performance for accurate machine learning prediction compared to a traditional OLS regression model.

3 Data

We obtain monthly data on eight commodity indices from World Bank (2023). The price movements of the commodities are shown in Figure 1. The commodities are classified as energy, non-energy and precious metals (PM). The energy index is the weighted average of the prices of coal, crude oil and natural gas. The non-energy group is the weighted average of other indices, it consists of agriculture, beverage, food, raw materials (RM) and metals and minerals (M&M). The sample spans January 1982 to January 2023, giving us 492 monthly observations for each of the eight commodities. The initial training set spans January 1982 to December 1990 making the first out-of-sample prediction on January 1991, further explanation on the estimation window can be found in Subsection 4.1.

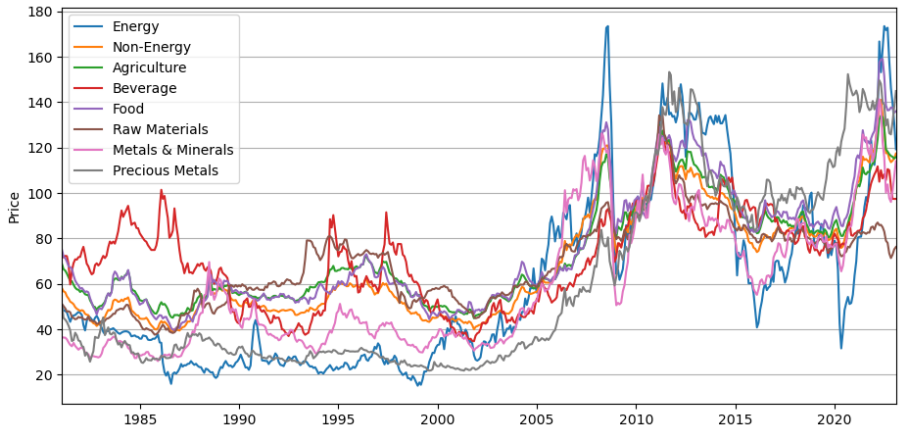

Figure 1: The eight commodity price indices over time.

Table 1 displays the descriptive statistics of our sample. We find the commodities precious metals and energy to be the most volatile and beverage and raw materials the least volatile. We find the distribution of up movements for the initial training set in Table 2 and for the initial out-of-sample set in Table 3. There are greater disparities between up and down movements in the initial training set, which get balanced out when we look at the initial out-of-sample distribution.

Table 1: Descriptive statistics of the sample

	Energy	Non-Energy	Agriculture	Beverage	Food	RM	M&M	PM
Mean	59.90	68.77	72.88	71.49	74.62	69.35	60.35	60.13
Std. Dev.	39.72	25.10	22.88	20.82	27.12	18.84	29.29	41.17
Min	15.20	38.97	44.87	34.77	39.65	37.69	26.05	21.84
Max	173.48	141.07	134.07	124.81	159.04	134.56	141.28	153.29

Commodities are subject to supply and demand dynamics, geopolitical events, weather conditions and global economic trends (Pindyck, 2004). Therefore, they can experience significant price fluctuations over relatively short periods of time. Commodities such as precious metals are generally perceived as less volatile compared to energy commodities, which are often influenced by geopolitical events and supply disruptions. However, precious metals are often used as 'safe-haven' assets during periods of economic uncertainty, causing investors to flock to this commodity during financial crises leading to significant price swings, explaining its high standard deviation in Table 1. In 2008, there was a significant surge in commodity prices followed by a sharp decline caused by the global financial crisis (Caballero et al., 2008). More recently, 2020 had another period of volatility primarily due to the COVID-19 pandemic which for example resulted in a historic collapse of oil prices caused by global lockdowns (Bakas & Triantafyllou, 2020). Both events are visible in Figure 1.

Table 2: Up distribution of the initial training set

	Energy	Non-Energy	Agriculture	Beverage	Food	RM	M&M	PM
Sum	58	53	56	62	55	66	57	57
%	44.6	40.8	43.1	47,7	42.3	50.8	43.8	43.8

Table 3: Up distribution of the initial out-of-sample set

	Energy	Non-Energy	Agriculture	Beverage	Food	RM	M&M	PM
Sum	212	203	187	197	188	192	203	183
%	57.1	54.7	50.4	53.1	50.7	51.8	54.7	49.3

3.1 Data Transformation

For the OLS method, we use a log transformation to make the data stationary and perform the predictive regression as follows:

$$r_{t+1} = \log(P_{t+1}) - \log(P_t), \tag{1}$$

where P_t denotes a commodity price in month t. Because we are interested in the directional movement, we convert this to an up-or-down prediction by considering the sign,

$$y_{t+1} = \text{sign}(r_{t+1}). \tag{2}$$

Similarly for the Random Forest and SVM, we consider 0 for a negative return and 1 for a positive return. Furthermore, we normalize the non-binary input by dividing it by its standard

deviation. This ensures that larger values do not overwhelm smaller input value inputs, allowing the models to focus on the relative relationships rather than being influenced by their absolute values and helping to reduce prediction error (Pan, Zhuang & Fong, 2016).

3.2 Technical Indicators

We use the technical indicators as proposed by Wang et al. (2020). While they only use binary indicators, we also consider the continuous non-binary variant. We denote the non-binary indicators by $S_{t,rule}$ and the binary indicators by $Z_{t,rule}$. By transforming the indicators from binary to non-binary to reflect the absolute signal strength, we lose less information. The non-binary indicators may be richer and more nuanced, capturing subtleties that the binary indicator might miss. However, the binary indicator may enhance the stability of the model, by reducing the impact of possible outliers and noise. Furthermore, the binary representation may generalize better by preventing overfitting. The continuous input can capture non-linear patterns and interactions between continuous features, while binary model interactions are based on the presence or absence of indicators. More model-specific interactions with binary and non-binary indicators are discussed in Subsection 4.2.1 and 4.3.1, after the models are explained.

3.2.1 Momentum Rule (MOM)

The momentum rule (MOM) measures the rate of change in price. For the binary variant, it helps identify the direction of the price over a certain period. For the non-binary variant, it also indicates the strength of the signal. If the momentum starts to slow down, it may signal that the trend is about to reverse. It can also help identify overbought or oversold conditions. When the momentum reaches an extreme high, it may indicate a pullback by considering the asset overbought. Contrariwise, when the momentum reaches an extreme low, the asset can be considered oversold which may indicate a potential rally. The momentum rule is as follows:

$$S_{t,MOM} = \log(P_t) - \log(P_{t-k}), \tag{3}$$

$$Z_{t,MOM} = \begin{cases} 1, & \text{if } P_t \geq P_{t-k}, \\ 0, & \text{if } P_t < P_{t-k}, \end{cases} \tag{4}$$

where k is the look-back period. We choose $k = 1$, 3, 6, 9 and 12 resulting in five binary and five non-binary momentum indicators.

3.2.2 Filtering Rule (FLT)

For the binary indicator, the filtering rule (FLT) produces a buying or selling signal whenever the price has surpassed its most recent low or high by more than a given percentage. For the non-binary indicator, we define it as the percentual difference between its current price and its previous minimum and the percentual difference between its previous maximum and its current price. By incorporating a threshold or looking at the specific percentual difference, this rule can help find when the signal is considered valid. It can also manage risk by limiting a trader's potential losses and locking in their profits, by setting up a filter. It is defined as follows:

$$S^{buy}_{t,FLT} = \log\left(\frac{P_t}{\min\{P_1, \ldots, P_{t-k}\}}\right), \tag{5}$$

$$S^{sell}_{t,FLT} = \log\left(\frac{\max\{P_1, \ldots, P_{t-k}\}}{P_t}\right), \tag{6}$$

$$Z^{buy}_{t,FLT} = \begin{cases} 1, & \text{if } P_t \geq (1 + \frac{\eta}{100}) * \min\{P_1, \ldots, P_{t-k}\}, \\ 0, & \text{otherwise,} \end{cases} \tag{7}$$

$$Z^{sell}_{t,FLT} = \begin{cases} 1, & \text{if } P_t \leq (1 - \frac{\eta}{100}) * \max\{P_1, \ldots, P_{t-k}\}, \\ 0, & \text{otherwise,} \end{cases} \tag{8}$$

where the look-back period $k = 1$, 3, 6, 9 and 12, and $\eta = 5$ and 10, resulting in 20 binary and 10 non-binary indicators.

3.2.3 Moving Average Rule (MVA)

The moving average rule (MVA) compares the short and long-term moving average. For the binary variant, when the short-term moving average crosses above the long-term moving average, this is called a golden cross and produces a buy signal. Conversely, when the short-term average crosses below the long-term moving average, this is called a death cross and produces a sell signal. For the non-binary variant, we consider the percentual difference between the short-term and long-term moving averages. This rule helps smooth out price fluctuations and noise, allowing us to better discern underlying up and downtrends. Traders often use moving averages to determine optimal entry and exit points for trades. It is defined as follows:

$$S_{t,MVA} = \log(MVA_{s,t}) - \log(MVA_{l,t}), \tag{9}$$

$$Z_{t,MVA} = \begin{cases} 1, & \text{if } MVA_{s,t} \geq MVA_{l,t}, \\ 0, & \text{otherwise,} \end{cases} \tag{10}$$

where $MVA_j, t = (\frac{1}{j}) \sum_{i=0}^{j-1} P_{t-i}$, $j = s, l$. Here, s and l are the short and long look-back periods. We use $s, l = 1$, 3, 6, 9 and 12 ($s < l$), resulting in 10 binary and 10 non-binary indicators.

3.2.4 Oscillator Rule (OSC)

The oscillator trading rule (OSC) reflects whether there have been too rapid price swings in a recent period. The binary variant produces a buy or sell signal on the relative strength indicator (RSI). For the non-binary variant, we consider the RSI directly. This rule is primarily used to identify overbought and oversold conditions. The oscillator reaching an extreme high implies that the asset is overbought and suggests that it is a good time to sell. On the other hand, when the oscillator reaches an extreme low, the asset is considered oversold, suggesting it may be a good time to buy. Furthermore, it can also confirm the strength of a current trend. If both the price and oscillator are rising, it can confirm the strength of the upward trend. The converse is true for a downward trend. It is defined as follows:

$$S_{t,OSC} = RSI(k), \tag{11}$$

$$Z^{buy}_{t,OSC} = \begin{cases} 1, & \text{if } RSI \leq 50 + \eta, \\ 0, & \text{otherwise,} \end{cases} \tag{12}$$

$$Z^{sell}_{t,OSC} = \begin{cases} 1, & \text{if } RSI \geq 50 + \eta, \\ 0, & \text{otherwise,} \end{cases} \tag{13}$$

where

$$RSI(k) = 100\frac{U_t(k)}{U_t(k) + D_t(k)}, \tag{14}$$

$$U_t(k) = \sum_{j=0}^{k-1} I(P_{t-j} - P_{t-j-1} > 0)(P_{t-j} - P_{t-j-1}), \tag{15}$$

$$D_t(k) = \sum_{j=0}^{k-1} I(P_{t-j} - P_{t-j-1} < 0)|P_{t-j} - P_{t-j-1}|. \tag{16}$$

Here $k = 1$, 3, 6, 9 and 12, $\eta = 5$ and 10. This results in 20 binary indicators and five non-binary indicators.

3.2.5 Support-Resistance Rule (SUP)

Lastly, the support-resistance rule (SUP) compares the current price with the support and resistance level, that is respectively representing the buying and selling pressure. The binary variant looks at whether the price exceeds a certain threshold, while the non-binary variant looks at the percentual difference between the current price and the previous maximum and between the previous minimum and the current price. Like the moving average rule, the support-resistance rule can help determine entry and exit points. It can also help predict price breakouts, that is when the price breaks through a support or resistance level and causes significant price movement. Like the filtering rule, it can also help with setting stop-loss orders. Lastly, it helps in understanding the psychology of traders, representing the price levels at which traders are willing to buy or sell an asset. It is defined as follows:

$$S_{t,SUP}^{buy} = \log\left(\frac{P_t}{\max\{P_1, \ldots, P_{t-k}\}}\right), \tag{17}$$

$$S_{t,SUP}^{sell} = \log\left(\frac{\min\{P_1, \ldots, P_{t-k}\}}{P_t}\right), \tag{18}$$

$$Z_{t,SUP}^{buy} = \begin{cases} 1, & \text{if } P_t \geq (1 + \frac{\eta}{100}) * \max\{P_1, \ldots, P_{t-k}\}, \\ 0, & \text{otherwise,} \end{cases} \tag{19}$$

$$Z_{t,SUP}^{sell} = \begin{cases} 1, & \text{if } P_t \geq (1 - \frac{\eta}{100}) * \min\{P_1, \ldots, P_{t-k}\}, \\ 0, & \text{otherwise,} \end{cases} \tag{20}$$

where for the binary variant we consider $k = 1$, 3, 6, 9 and 12 and $\eta = 5$ and 10, resulting in 50 indicators and for the non-binary variant we consider $k = 3$, 6, 9 and 12, giving us eight indicators. We forego $k = 1$ for the non-binary variant, as this is equal to $S_{t,FR}^{buy}$ for $k = 1$.

4 Methodology

To predict the directional movement of the monthly commodity prices, we incorporate the indicators in Random Forest and SVM models. For each model, we consider a binary and non-binary variant. We then evaluate the out-of-sample performance of these models, comparing their results to a simple weighted statistical regression as performed by Wang et al. (2020). Lastly, we assess the financial gain by means of the Sharpe ratio and also assess the significance of the models by two statistical tests, McNemar (1947) and Pesaran and Timmermann (2009). The computations are performed in Python 3.10 using the Scikit-learn 1.2.2 library.

4.1 Estimation Window

We use an expanding window for training the models and making out-of-sample predictions for OLS, Random Forest and SVM. We divide the sample of T observations, into an in-sample part with the first M observations and an out-of-sample part with the remaining $T - M$ observations. To obtain the prediction y_k at time k, we use $Z_{t,i}$ for the binary model and $S_{t,i}$ for the non-binary model, the observations $t = 1, ..., k-1$ for $k \geq M$ with indicator rules $i \in \{MOM, FR, MVA, OSC, SUP\}$. Repeating this procedure for $k = M+1, ..., T$ yields a time series of $T - M$ one-step-ahead predictions.

4.2 Random Forest (RF)

Introduced by Breiman (2001), the Random Forest is an ensemble learning method that combines multiple decorrelated decision trees, each built on different data subsets to make predictions. The diverse trees mitigate overfitting and enhance model robustness, which is particularly beneficial given the multitude of technical indicators as input. Because of their ensemble nature, Random Forests are robust to outliers which is a desirable trait given the volatile nature of commodity prices. The model handles non-linear relationships and interactions between technical indicators well, capturing complex patterns for accurate predictions. For each commodity, we construct a forest with binary and non-binary input.

The training dataset consists of N input vectors $X_i \in R^d$ and corresponding labels $Y_i \in \{1, 0\}$. The algorithm begins by creating multiple subsets of the training data, also called bootstrap samples, to reduce the in-sample bias of each tree by randomly sampling with replacement, $\{D_1, ..., D_B\}$, each of size N. This is bootstrap aggregating, also known as bagging. By doing so, we introduce diversity into the training data for each decision tree. Next, the Random Forest constructs a decision tree T_b for each bootstrap sample D_b. The decision tree splits the data recursively based on a randomly selected set of features and optimization criteria. The randomization of the feature set ensures that each tree focuses on different features and reduces the correlation among the trees.

Let the data at node m be Q_m with n_m samples. For each candidate split $\theta = (j, t_m)$ consisting of a feature j and threshold t_m partition the data into a left and right subtree:

$$Q_m^{left}(\theta) = ((X, Y)|X_j \leq t_m), \tag{21}$$

$$Q_m^{right}(\theta) = ((X, Y)|X_j > t_m). \tag{22}$$

The quality of a proposed partition of a node m is then computed through, in our case of classification, an impurity function $H(\cdot)$,

$$G(Q_m, \theta) = \frac{n_m^{left}}{n_m} H(Q_m^{left}(\theta)) + \frac{n_m^{right}}{n_m} H(Q_m^{right}(\theta)), \tag{23}$$

where we use the Gini impurity measure and define this as

$$H(Q_m) = \sum_k p_{mk}(1 - p_{mk}), \tag{24}$$

where

$$p_{mk} = \frac{1}{n_m} \sum_{y \in Q_m} I(y = k) \tag{25}$$

is the proportion of class k observations in node m. Finally, we select the parameters that minimize the impurity,

$$\theta^* = \text{argmin}_\theta G(Q_m, \theta). \tag{26}$$

This recursive process continues until a stopping criterion is met, such as reaching a minimum number of samples in each leaf node. For classification, the final Random Forest prediction is determined through majority voting,

$$f(X) = \text{argmax}_c(\sum_b I(T_b(X) = c)), \tag{27}$$

where c represents the class label. We present the pseudocode in Appendix B.

Furthermore, Random Forests provide a measure of feature importance, indicating which technical indicators have the greatest influence on price movement. The importance of a feature is calculated according to its ability to increase the pureness of the leaves, this is called Gini importance. Features that are selected for splitting near the root of the trees in the forest have a larger impact on node purity. We take the average Gini importance of all trees and normalize these values so they add up to 1.

4.2.1 Binary versus Non-Binary Input

We consider the Random Forest with binary and non-binary input. The binary input creates a straightforward decision boundary, essentially splitting the feature space into 0 and 1 at each decision node creating a clear separation. However, the non-binary input is able to create a more complex partition of the feature space, as at each node it can split at any value within the non-binary range. This model may be finer-grained and create a more nuanced modeling of the relationships, compared to its binary counterpart. Because of the Random Forest's inherent ability to select important features, we use all 110 binary and 37 non-binary features as input.

4.2.2 Hyperparameters

The Random Forest is a non-parametric model but has several hyperparameters: number of trees in the ensemble, maximum depth of each tree, maximum features considered at each split and splitting criterion. As described above, we use the Gini impurity as a splitting criterion. Unless explicitly stated otherwise, we adhere to the default hyperparameter values as described by the Scikit-learn implementation of the Random Forest algorithm. We vary the two hyperparameters in our random forest model: the number of trees in the ensemble and the number of features to try at each split. Following Ballings et al. (2015), we consider a large number of trees and the square root of the total number of features as the maximum features hyperparameter. However, to ensure robustness and optimize performance, we also explore additional configurations seen in Table 4.

Table 4: Hyperparameters of the Random Forest

Parameters	Levels
Maximum trees	100, 250, 500
Maximum features	6, 12, 18

4.3 Support Vector Machine (SVM)

SVM is a supervised learning method that has been used for classification, recognition, regression and also time series. It originates from research in statistical learning theory regulating generalization to find the optimal trade-off between structural complexity and empirical risk (Cortes & Vapnik, 1995). It finds an optimal hyperplane in a high-dimensional feature space to separate different classes, in this case, up or down. SVMs are known to be effective in high-dimensional input space, which is beneficial given the multitude of technical indicators. Like for the Random Forest, we also create a binary and a non-binary SVM for each commodity, with its respective indicator input.

The SVM constructs a hyperplane as the decision surface to maximize the margin between the up and down examples. It learns to classify a training set of examples, with input vectors $X_i \in R^d$ and corresponding labels $Y_i \in \{1, 0\}$. The goal is to construct a decision function based on the available examples, to minimize the probability of misclassifying a new sample. SVM maps the input vectors X_i to a high-dimensional feature space $\phi(X_i) \in H$ to construct an Optimal Separating Hyperplane, which maximizes the distance between the hyperplane and the nearest observations of both classes in the space H. The cost parameter C also called the regularization or penalty parameter, controls the trade-off between the size of the margin and the hyperplane violations. That is to say, the trade-off between a low training error and minimizing the complexity of the decision boundary. This represents a non-linear decision boundary in the input space, as illustrated in Figure 2. The training examples that are closest to the maximum margin are called the support vectors, all other examples are irrelevant to defining the boundary. The pseudo-code can be found in Appendix B.

This maximum margin hyperplane can be represented as the following:

$$f(X) = \text{sign}(\sum_{i=1}^{N} Y_i \alpha_i \cdot K(X, X_i) + b), \tag{28}$$

$$\max \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j \cdot Y_i Y_j \cdot K(X_i, X_j), \tag{29}$$

$$\text{subject to } 0 \leq \alpha_i \leq c, \tag{30}$$

$$\sum_{i=1}^{N} \alpha_i Y_i = 0, i = 1, 2, ..., N. \tag{31}$$

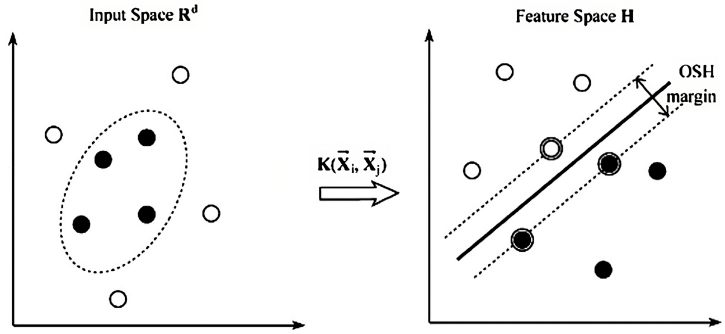

Figure 2: Visualization of a non-linear boundary in the input space corresponding to a separating hyperplane in the feature space (Hua & Sun, 2001).

Here, K is the kernel function. We use the Gaussian radial basis function as the kernel function, as in Equation (32), because it is able to capture complex nonlinear relationships well (Kim, 2003). Kim also shows that the Gaussian kernel provides better results and takes less time to train compared to other non-linear kernels. This in turn maps $\phi(\cdot)$, with γ being the constant. The γ defines the influence of each training example on the decision boundary.

$$K(X_i, X_j) = exp(-\gamma ||X_i - X_j||^2) \tag{32}$$

4.3.1 Binary versus Non-Binary Input

We consider the SVM with binary and non-binary input. The binary input can work well with the SVM, especially in cases where there is a clear separation between classes, here a hard margin can be drawn. Non-binary input, on the other hand, allows for a soft margin where the SVM tries to balance maximizing the margin and reducing classification errors. This is useful when the classes are not perfectly separable, as often is the case in financial data. Non-binary input may also be beneficial in the context of kernel functions, as it provides the kernel with more information which may lead to more complex and effective mappings. Using many features increases the dimensions of the higher dimensional projection performed by the kernel even more and considering that training the SVM is a quadratic optimization problem, this can become computationally complex and expensive. Moreso, compared to the Random Forest, the SVM is more prone to overfitting. For these reasons, we perform feature selection for each commodity and encoding by considering the features from the Random Forest model that have an average

feature importance greater than 0.02 for the binary features and 0.03 for the non-binary features. The threshold difference is explained by the difference in the number of features for the binary and non-binary indicators, with the non-binary encoding having fewer indicators (37) than its binary counterpart (110). This is done to obtain an appropriate amount of features.

4.3.2 Hyperparameters

The SVM here has only two hyperparameters: the cost parameter C and the kernel coefficient γ. Tay and Cao (2001) show the importance of the hyperparameter choice in the performance of SVMS. Improper selection can lead to under- or overfitting. To determine the optimal parameter values, we follow Kara, Acar Boyacioglu and Baykan (2011), but because of limited computing power, we consider greater intervals in the γ level. Table 5 shows the hyperparameter combinations.

Table 5: Hyperparameters of the SVM radial kernel

Parameters	Levels
C	1, 10, 100
γ	2, 2.5, 3, 3.5, 4

4.4 Benchmark

We follow Wang et al. (2020) and use the following OLS predictive regression to detect predictability for each commodity:

$$r_{t+1} = \alpha + \beta x_t + \epsilon_{t+1} \sim N(0, \sigma_\varepsilon^2), \tag{33}$$

where r_t denotes the commodity price change, x_t is a predictor variable and ϵ_t is the disturbance term, with assumed i.i.d. We perform this regression for each commodity, encoding and technical indicator. Each trading rule generates many technical indicators which result in different forecasts. For each trading rule, we simply take the weighted average forecast,

$$\hat{r}_{j,t} = \frac{1}{P_j} \sum_{i=1}^{P_j} \hat{r}_{i,j,t}, j \in R \tag{34}$$

where $R = \{MOM, FR, MVA, OSC, SR\}$. Finally, we consider an equally weighted forecast combination,

$$\hat{r}_{t,comb} = \sum_{j=1}^{N} \frac{1}{|R|} \hat{r}_{j,t}, \tag{35}$$

and use the signs of these combined predictions as the benchmark for the machine learning models. In this paper, we will refer to this OLS regression model as the OLS model.

4.5 Evaluation

4.5.1 Predictability

We evaluate the different models' performance for their robustness in predictability by means of accuracy, which is defined as follows,

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{36}$$

where $TP = \#$ true positive values, $TN = \#$ true negative values, $FP = \#$ false positive values, $FN = \#$ false negative values.

4.5.2 Financial Gain

To assert the financial gain of the different models, we create a portfolio by considering a simple long-only strategy and evaluate the Sharpe ratio for the models. This measures the performance of investments compared to a risk-free asset but scales it to its risk. It is defined as follows,

$$\text{Sharpe} = \frac{R_p - R_f}{\sigma}, \tag{37}$$

where R_p is the portfolio return, R_f is the risk-free return where we take the U.S. 3-month treasury bill. The top part is known as the excess return and σ is the standard deviation of the excess return. We multiply the Sharpe ratio by $\sqrt{12}$ to obtain the annualized Sharpe ratio.

4.5.3 Significance

To assess the significance of the findings, we perform the Pesaran and Timmermann (2009) test and McNemar (1947) test. In order to compare the models' predictive ability, we use the McNemar test. This non-parametric method assesses the difference in the error rates between two models that predict the same data. By constructing a 2×2 contingency table that counts the correct and incorrect predictions made by both models, this statistic evaluates the symmetry of change between the two models to see if one outperforms the other. The Pesaran-Timmermann test is designed to evaluate the directional predictive accuracy of economic and financial forecasts and tests the null hypothesis that the model's accuracy is equal to 0.5, which would be akin to using a coin toss to make predictions. This is achieved by comparing these two accuracies.

5 Results

We explore the feature importance obtained from the binary and non-binary Random Forest models in Subsection 5.1. Next, we look at the performance achieved by various hyperparameter settings in Subsection 5.2 and compare their confusion matrices and overall accuracy per commodity and their significance in Subsection 5.3. Following, we look at the accuracy over time in Subsection 5.4 and lastly consider the achieved financial gain of each model in Subsection 5.5.

5.1 Prominent Features

Figure 3 shows the average feature importances of the binary Random Forest, offering insight into indicator characteristics. Most support-resistance indicators have minimal importance, except for the buy and sell indicators with look-back period $k = 3$ displaying relatively high importance for the commodities agriculture and raw materials, respectively. The $k = 1$ buy indicator of the filtering and momentum rule exhibits relatively high importance across all commodities, likely capturing the most recent price movement. The moving average rule displays a pervasive relevance across all commodities, although the indicators with short and long look-back periods $(s, l) = (3, 6)$ and (3,12) appear to be weaker, suggesting that longer time period comparisons yield less informative results. Conversely, the moving average indicators with $(s, l) =$ (6,9) and (9,12) demonstrate notable strength, indicating that short-term yet old data may hold considerable predictive power. Finally, we see that the oscillator indicators with $k = 6$ and 9 with threshold $\eta = 5$ stand out as the most influential indicators across all commodities.

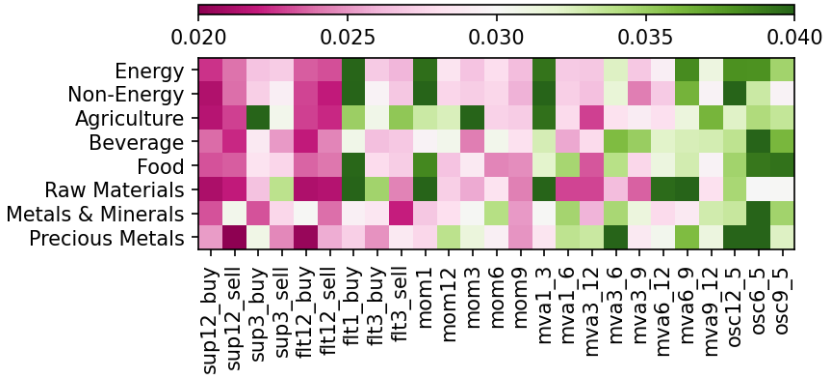

Figure 3: Binary Random Forest feature importance heatmap, where the color white is the threshold for SVM feature selection.

Figure 4 shows the heatmap for the non-binary Random Forest. Here the support-resistance rule gains some importance, while the oscillator rule, which was crucial in the binary model, loses relevance. None of the oscillator indicators surpass the feature selection threshold, indicating its diminished importance across all commodities. The filtering rule is more present here than in the binary model, with notably more sell indicators than buy ones. As we saw in the binary model, the moving average indicator with $(s, l) =$ (1,3) retains a strong presence across all commodities except for metals & minerals. A similar pattern can be seen for $(s, l) =$ (9,12), which performs even better than in the binary model.

Upon comparison, a few key observations emerge. The oscillator rule while highly relevant in the binary model, is barely important in the non-binary model. Conversely, the opposite is true for the support-resistance rule. The filtering rule appears to be an important rule for both models for certain time indicators. However, the most interesting observation is the consistent and considerable influence of the moving average rule in both models, particularly with $(s, l) =$ (1,3). Longer time horizon indicators appear to be less important in both models. This could be attributed to the higher level of noise present in such indicators, thus being less reflective of the actual stock movement, particularly in the case of non-binary ones.

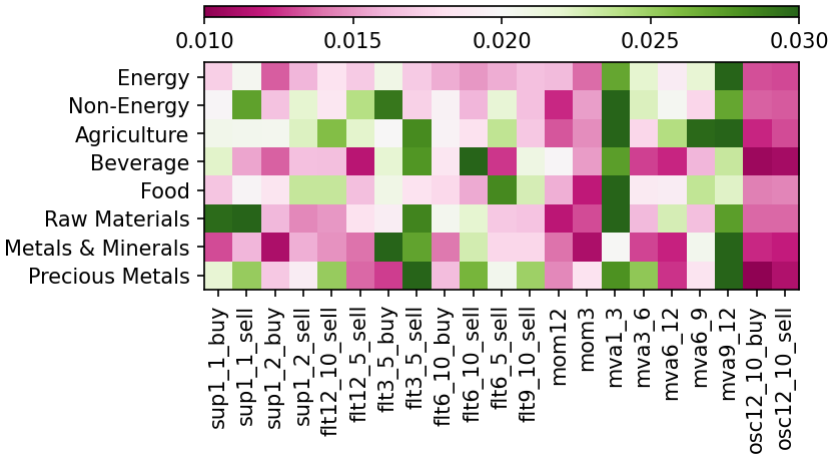

Figure 4: Non-binary feature importance heatmap, where the color white is the threshold for SVM feature selection.

5.2 Hyperparameter Selection

Table 6a shows the out-of-sample accuracy for the binary Random Forest, varying between 55.04% and 56.59%. We find the highest out-of-sample accuracy for 12 features and 100 trees with 56.59%. However, when looking at the training accuracy in Table 6b we observe that the binary Random Forest displays an almost constant training accuracy regardless of hyperparameter setting, at approximately 94.50%. For the non-binary Random Forest model, we look at Table 7a and 7b. We see that the out-of-sample accuracy fluctuates between 55.16% and 56.07%, which is lower than its binary variant, indicating a worse fit. The best-performing hyperparameter setting appears to be for 18 features and 250 trees with an accuracy of 56.07%. The training accuracy is consistently 100.00% for all settings, further indicating an overfit.

Table 6: Binary Random Forest accuracy (f = features, t = trees)

(a) Out-of-sample

f/t	100	250	500
6	56.07	56.33	55.65
12	56.59	56.01	55.39
18	55.04	55.59	56.04

(b) Training

f/t	100	250	500
6	94.38	94.50	94.50
12	94.38	94.50	94.50
18	94.38	94.50	94.50

Table 7: Non-binary Random Forest accuracy (f = features, t = trees)

(a) Out-of-sample

f/t	100	250	500
6	55.39	55.23	55.33
12	55.16	55.65	55.81
18	55.16	56.07	55.85

(b) Training

f/t	100	250	500
6	100.00	100.00	100.00
12	100.00	100.00	100.00
18	100.00	100.00	100.00

The binary SVM shows a consistent training accuracy of 70.65% for all hyperparameter settings in Table 8b. The out-of-sample accuracy is relatively constant as well, ranging from 54.55% to 55.62% in Table 8a. The results indicate that altering the hyperparameters has a negligible effect on the model's performance. Contrariwise, the non-binary SVM does show variable training accuracies depending on the settings in Table 9b. The training accuracy is near-perfect for higher C values. Here, we also find that this indicates overfitting on the training set, as the out-of-sample accuracy in Table 9a is consistently lower than those of the binary SVM, with values from 52.69% to 54.19%.

The Random Forest achieves higher out-of-sample accuracy than the SVM models, for both

Table 8: Binary SVM accuracy

(a) Out-of-sample

C/γ	2	2.5	3	3.5	4
1	55.62	55.29	54.64	54.55	54.61
10	55.42	55.23	55.10	54.74	54.68
100	55.42	55.23	55.10	54.74	54.68

(b) Training

C/γ	2	2.5	3	3.5	4
1	70.65	70.65	70.65	70.65	70.65
10	70.65	70.65	70.65	70.65	70.65
100	70.65	70.65	70.65	70.65	70.65

Table 9: Non-binary SVM accuracy

(a) Out-of-sample

C/γ	2	2.5	3	3.5	4
1	53.77	53.60	53.51	53.12	52.69
10	54.19	53.67	53.70	53.77	53.41
100	54.19	53.64	53.70	53.77	53.41

(b) Training

C/γ	2	2.5	3	3.5	4
1	96.98	98.21	98.95	99.29	99.50
10	99.99	100.00	100.00	100.00	100.00
100	100.00	100.00	100.00	100.00	100.00

binary and non-binary encoding. Among the Random Forest models, there was no great performance difference. For the SVM, increasing complexity by raising the C leads to higher training accuracy but not necessarily improved out-of-sample performance. The non-binary SVM performs worse than its binary counterpart on out-of-sample predictions, indicating overfitting with more complex models. The high training accuracies all imply the potential benefit of more sensitive hyperparameter tuning. All models are significant with $p < 0.001$, see Appendix C.1 for full statistics. For all further analysis, we consider the best-performing Random Forest and SVM models for each encoding and compare this to OLS as reported in Table 10.

Table 10: Accuracy of OLS and best-performing machine learning models

OLS		RF		SVM	
		$(f = 12, t = 100)$	$(f = 18, t = 250)$	$(C = 1, \gamma = 2)$	$(C = 10, \gamma = 2)$
Binary	Non-Binary	Binary	Non-Binary	Binary	Non-Binary
56.75	58.28	56.59	56.07	55.62	54.19

5.3 Overall Accuracy

Tables 11 and 12 show the confusion matrices of the binary and non-binary models. Comparing OLS with the machine learning models, we see that the former has more false positives than false negatives. This suggests that OLS may overestimate price increases, potentially leading to unnecessary buy decisions which lead to financial loss. On the other hand, the Random Forest and SVM have a more balanced distribution between false positives and negatives. Compared to OLS, machine learning models make fewer false predictions of price increases but also correctly predict fewer instances of price decreases. Despite this, machine learning models have a higher number of true positives than OLS, indicating that they may be more reliable for producing buy signals. This discrepancy is less pronounced for the non-binary models.

We zoom in on the commodities and find the three models' predictive accuracy in Table 13. For both binary and non-binary scenarios, we see that the OLS performs relatively well for non-energy, raw material and metals & minerals. Similarly, the Random Forest performs well

Table 11: Confusion matrices of binary models

(a) OLS

True/Pred.	1	0
1	680	791
0	529	1080

(b) Random Forest

True/Pred.	1	0
1	833	638
0	699	910

(c) SVM

True/Pred.	1	0
1	777	694
0	673	936

Table 12: Confusion matrices of non-binary models

(a) OLS

True/Pred.	1	0
1	712	759
0	543	1066

(b) Random Forest

True/Pred.	1	0
1	798	673
0	680	929

(c) SVM

True/Pred.	1	0
1	792	679
0	732	877

on raw material and metals & minerals. Despite exhibiting overall lower accuracy, the SVM also performs reasonably well on these two commodities.

The non-binary OLS outperforms its binary counterpart, barring the metals & minerals and precious metals commodities. Interestingly, precious metals is the only commodity where the non-binary OLS does not yield any significant results. In general, binary input for the machine learning models leads to higher accuracy. Neither of the machine learning models produces significant predictions for the energy commodity. Another noteworthy observation is the underperformance of the non-binary SVM in the beverage commodity, the only sub-50 accuracy across all models and commodities.

We find that non-binary models consistently produce lower and in some cases even non-significant accuracies when applied to precious metals, which is the most volatile commodity. This implies that binary indicators may be more effective for volatile commodities. Yet, when we examine energy, a commodity with comparable volatility, we find no trend as all machine learning accuracies fail to be statistically significant. Curiously, only raw materials, the least volatile commodity, yields higher accuracies for the machine learning models compared to OLS.

Table 13: Acccuracy of each commodity per model

	OLS		RF		SVM	
	Binary	Non-Binary	Binary	Non-Binary	Binary	Non-Binary
Energy	54.60**	56.75***	52.99	52.73	51.69	51.17
Non-Energy	61.35****	62.27****	56.10***	55.84**	56.10**	54.55*
Agriculture	55.21***	57.67***	57.40***	58.18****	57.66***	55.84***
Beverage	56.13***	56.75**	52.99	53.70*	53.77*	49.87
Food	53.37**	56.44***	54.55**	52.21	51.95	53.77*
Raw Material	57.36***	61.66****	62.98****	62.10****	59.48****	58.14***
Metals & Minerals	61.96****	61.04****	59.59****	59.70****	57.66****	59.46****
Precious Metals	54.29**	53.68	56.10***	54.03*	56.62***	50.65

Note: *, **, ***, **** denote rejections of the null hypotheses at respectively 10%, 5%, 1% and 0.1% significance levels.

Table 14 provides an evaluation of the different models' predictive performance disparities, by means of McNemar's Chi-squared statistics, the exact p-values can be found in Appendix C.2. While the difference between the binary and non-binary OLS is not statistically significant, the non-binary OLS does show superior performance when compared to the other models. Specific-

ally, the non-binary OLS significantly outperforms the non-binary Random Forest, which is not achieved by the binary OLS. Furthermore, there appears to be no significant difference between binary OLS and both Random Forest models. The SVM performs worst, being significantly worse than both OLS and the binary Random Forest for the non-binary SVM. In general, we do not find evidence of a superior encoding. Although there is no direct evidence to say that the non-binary OLS outperforms its binary counterpart, the non-binary OLS does hold an edge in certain contexts. This suggests its robustness and potential versatility over the other models and specifically over the binary OLS.

Table 14: McNemar's Chi-squared values

		OLS		RF		SVM	
		Binary	Non-Binary	Binary	Non-Binary	Binary	Non-Binary
OLS	Binary	-	90	547	490	467*	554****
	Non-Binary	-	-	539	487*	435**	541****
RF	Binary	-	-	-	414	477	513***
	Non-Binary	-	-	-	-	485	443
SVM	Binary	-	-	-	-	-	577
	Non-Binary	-	-	-	-	-	-

Note: *, **, ***, **** denote rejections of the null hypotheses at respectively 15%, 10%, 5% and 1% significance levels.

5.4 Temporal Accuracy

The dynamics of the predictive performance over time can provide valuable insights into long-term utility and robustness under varying economic conditions. We look at each model's average 24-month rolling window accuracy, allowing us to compare its consistency and accuracy fluctuations to the average commodity price shifts over time.

Figure 5 shows the 24-month accuracy of the OLS predictions and the average commodity price. The binary and non-binary OLS closely follow each other in terms of accuracy, with the non-binary slightly outperforming its counterpart. The models display a 'shock' sensitivity to market conditions, with notable swings in accuracy as the commodity prices fluctuate. While the OLS has periods of high accuracy, particularly during the price surges in 2008 and 2020, they also encounter great performance dips during the subsequent economic downturn.

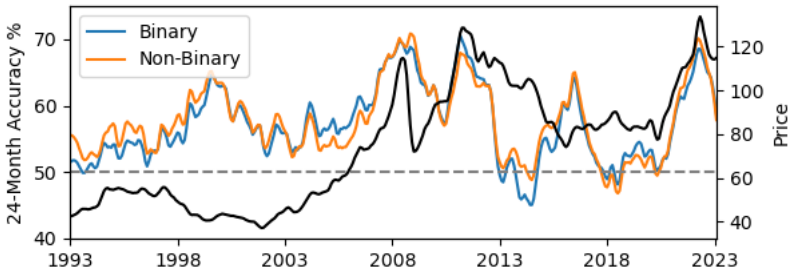

Figure 5: OLS average 24-month rolling window accuracy

Figure 6 and 7 show the accuracy over time for the Random Forest and SVM. The Random Forest and SVM present less rigid accuracy lines, implying less sensitivity to external changes. Although these models achieve lower average accuracy than OLS, their relatively smaller fluctuations indicate more stable and reliable performance. From the two machine learning models,

the Random Forest shows the least volatile accuracy indicating the most consistent performance. While the SVM has smoother accuracy than OLS, it still exhibits larger fluctuations compared to the Random Forest model.

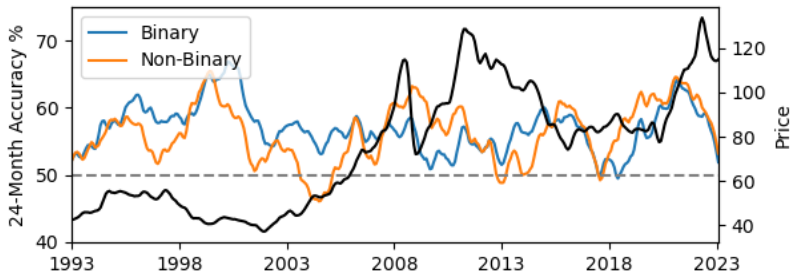

Figure 6: Random Forest average 24-month rolling window accuracy

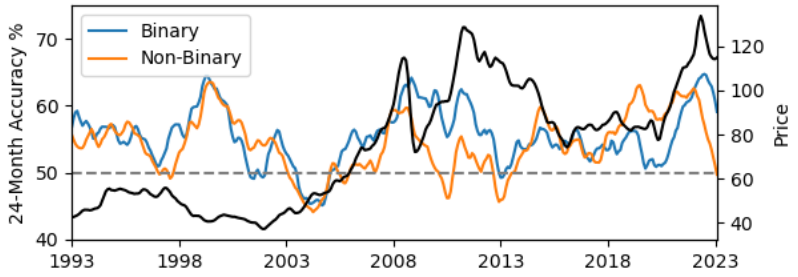

Figure 7: SVM average 24-month rolling window accuracy

Overall, this implies a potential trade-off between stability and peak performance, a characteristic reminiscent of the bias-variance trade-off concept in both statistics and machine learning. While the OLS model has relatively high accuracy during periods of economic growth, it is also more prone to significant dips during downturns. OLS regressions often have high bias but low variance, oversimplifying the problem and consequently exhibiting an increased sensitivity to market fluctuations, explaining the pronounced performance swings (Makridakis, Wheelwright & Hyndman, 2008). Conversely, the Random Forest and SVM are characterized by lower bias and higher variance. They are more capable of handling high-dimensional data. They offer more stability and are less sensitive to economic changes, albeit at the cost of a slightly lower average accuracy. This aligns with the findings of the confusion matrices in Subsection 5.3.

5.5 Financial Gain

Table 15 shows the annualized Sharpe ratios of the models. In both binary and non-binary encodings, we find that the OLS model has the highest Sharpe ratios indicating the best risk-adjusted performance compared among the three, despite having found a lower buy accuracy for OLS in Subsection 5.3. Similarly for the OLS, the non-binary Random Forest has a slightly better Sharpe compared to its binary counterpart. The SVM has the worst performance of all, with a very low Sharpe ratio of 0.143 for the binary SVM and even a negative ratio -0.041 for the non-binary SVM. The latter suggests that the risk-free asset outperforms the non-binary SVM in terms of risk-adjusted return. The order of Sharpe ratios is in line with the findings for accuracy as reported in Subsection 5.2.

Table 15: Annualized Sharpe ratio of OLS, Random Forest and SVM

OLS		RF		SVM	
Binary	Non-Binary	Binary	Non-Binary	Binary	Non-Binary
0.450	0.465	0.258	0.264	0.143	-0.041

6 Conclusion

Predicting the movement of commodity prices is important for the development of effective market trading strategies, potentially securing profitable returns for investors. This paper attempts to predict the movement of commodity prices, emphasizing the relevance of using machine learning models and highlighting the implications of binary and non-binary input encoding. We examine three models — OLS, Random forest and SVM — with different encodings. The predictions are tested on monthly data ranging from 1982 to 2023, incorporating 110 binary indicators and 37 non-binary indicators.

Based on this experiment, we find that the OLS model, with an average accuracy of 57.52%, surpasses the Random Forest (56.33%) and SVM (54.91%) in terms of overall predictive accuracy. However, the OLS model tends to overestimate upward movement and shows increased sensitivity to market fluctuations. On the other hand, the Random Forest and SVM display superior accuracy for upward movement and overall stability albeit with a slightly lower overall accuracy. This dichotomy hints at a trade-off between peak performance and stability, suggesting the need for more sensitive hyperparameter tuning to mitigate potential overfitting issues.

We find that short-term indicators, in particular the moving average rule, play a significant role in predicting commodity movement for both binary and non-binary models. The relevance of specific indicators varies between encoding, highlighting the need for indicator-specific encoding. Curiously, our findings hint that binary indicators may offer an advantage in modeling highly volatile commodities, warranting additional research.

While binary input seemingly enhances the accuracy of the Random Forest and SVM, there is no statistical evidence to support this. Although the binary and non-binary OLS model's accuracies (56.75% and 58.28%) were not significantly different, the non-binary variant demonstrates significantly superior predictability compared to other models. Indicating that the non-binary OLS may offer an advantage in a broader comparative sense. The use of non-binary indicators modestly improves the annualized Sharpe ratio for the OLS (0.465 versus 0.450) and Random Forest model (0.264 versus 0.258), but negatively affects the SVM (-0.041 versus 0.143). However, this does not take into account transaction costs which range from 0.0004% to 0.033% in the futures market (Locke & Venkatesh, 1997), which may affect potential profit.

Overall, this paper shows the relevance of machine learning models compared to OLS and the implications of technical indicator encoding for predicting commodity prices. To improve predictive performance, future research could focus on fine-tuning model parameters, incorporating macro-economic, financial or even sentiment-based indicators, exploring alternative binary to non-binary encoding strategies or even using mixed binary and non-binary inputs.

References

Aminimehr, A., Raoofi, A., Aminimehr, A. & Aminimehr, A. (2022, June). A comprehensive study of market prediction from efficient market hypothesis up to late intelligent market prediction approaches. *Computational Economics*, *60*(2), 781–815. Retrieved from https://doi.org/10.1007/s10614-022-10283-1 doi: 10.1007/s10614-022-10283-1

Bakas, D. & Triantafyllou, A. (2020, August). Commodity price volatility and the economic uncertainty of pandemics. *Economics Letters*, *193*, 109283. Retrieved from https://doi.org/10.1016/j.econlet.2020.109283 doi: 10.1016/j.econlet.2020.109283

Ballings, M., Van Den Poel, D., Hespeels, N. & Gryp, R. (2015, November). Evaluating multiple classifiers for stock price direction prediction. *Expert Systems with Applications*, *42*(20), 7046–7056. Retrieved 2023-06-12, from https://linkinghub.elsevier.com/retrieve/pii/S0957417415003334 doi: 10.1016/j.eswa.2015.05.013

Bhattacharjee, I. & Bhattacharja, P. (2019, December). Stock price prediction: A comparative study between traditional statistical approach and machine learning approach. In *2019 4th international conference on electrical information and communication technology (EICT)*. IEEE. Retrieved from https://doi.org/10.1109/eict48899.2019.9068850 doi: 10.1109/eict48899.2019.9068850

Breiman, L. (2001). Random forests. , *45*(1), 5–32. Retrieved from https://doi.org/10.1023/a:1010933404324 doi: 10.1023/a:1010933404324

Caballero, R. J., Farhi, E. & Gourinchas, P.-O. (2008). *Financial crash, commodity prices and global imbalances* (Tech. Rep.). National Bureau of Economic Research.

Choudhry, R. & Garg, K. (2008). A hybrid machine learning system for stock market forecasting. *International Journal of Computer and Information Engineering*, *2*(3), 689–692.

Cortes, C. & Vapnik, V. (1995, September). Support-vector networks. *Machine Learning*, *20*(3), 273–297. Retrieved from https://doi.org/10.1007/bf00994018 doi: 10.1007/bf00994018

Dash, R. & Dash, P. K. (2016, March). A hybrid stock trading framework integrating technical analysis with machine learning techniques. *The Journal of Finance and Data Science*, *2*(1), 42–57. Retrieved from https://doi.org/10.1016/j.jfds.2016.03.002 doi: 10.1016/j.jfds.2016.03.002

Deaton, A. & Laroque, G. (1992, January). On the behaviour of commodity prices. *The Review of Economic Studies*, *59*(1), 1. Retrieved from https://doi.org/10.2307/2297923 doi: 10.2307/2297923

Engle, R. F. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflation. *Econometrica: Journal of the econometric society*, 987–1007.

Fama, E. F. (1970, May). Efficient capital markets: A review of theory and empirical work. *The Journal of Finance*, *25*(2), 383. Retrieved from https://doi.org/10.2307/2325486 doi: 10.2307/2325486

Gu, S., Kelly, B. & Xiu, D. (2020, February). Empirical asset pricing via machine learning. *The Review of Financial Studies*, *33*(5), 2223–2273. Retrieved from https://doi.org/10.1093/rfs/hhaa009 doi: 10.1093/rfs/hhaa009

Hua, S. & Sun, Z. (2001, August). Support vector machine approach for protein subcellular

localization prediction. *Bioinformatics*, *17*(8), 721–728. Retrieved from https://doi.org/10.1093/bioinformatics/17.8.721 doi: 10.1093/bioinformatics/17.8.721

Kara, Y., Acar Boyacioglu, M. & Baykan, K. (2011, May). Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the Istanbul Stock Exchange. *Expert Systems with Applications*, *38*(5), 5311–5319. Retrieved 2023-06-12, from https://linkinghub.elsevier.com/retrieve/pii/S0957417410011711 doi: 10.1016/j.eswa.2010.10.027

Kilian, L. & Murphy, D. P. (2014). The role of inventories and speculative trading in the global market for crude oil. *Journal of Applied econometrics*, *29*(3), 454–478.

Kim, K.-j. (2003). Financial time series forecasting using support vector machines. *Neurocomputing*, *55*(1–2), 307–319. doi: 10.1016/s0925-2312(03)00372-2

Kumar, M. & M., T. (2006). Forecasting Stock Index Movement: A Comparison of Support Vector Machines and Random Forest. *SSRN Electronic Journal*. Retrieved 2023-06-12, from http://www.ssrn.com/abstract=876544 doi: 10.2139/ssrn.876544

Kwiatkowski, D., Phillips, P. C., Schmidt, P. & Shin, Y. (1992). Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root? *Journal of econometrics*, *54*(1-3), 159–178.

Lo, A. W., Mamaysky, H. & Wang, J. (2000, August). Foundations of technical analysis: Computational algorithms, statistical inference, and empirical implementation. *The Journal of Finance*, *55*(4), 1705–1765. Retrieved from https://doi.org/10.1111/0022-1082.00265 doi: 10.1111/0022-1082.00265

Locke, P. R. & Venkatesh, P. C. (1997, April). Futures market transaction costs. *Journal of Futures Markets*, *17*(2), 229–245. Retrieved from https://doi.org/10.1002/(sici)1096-9934(199704)17:2<229::aid-fut5>3.0.co;2-l doi: 10.1002/(sici)1096-9934(199704)17:2⟨229::aid-fut5⟩3.0.co;2-l

Makridakis, S., Wheelwright, S. C. & Hyndman, R. J. (2008). *Forecasting methods and applications*. John wiley & sons.

McNemar, Q. (1947, June). Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, *12*(2), 153–157. Retrieved from https://doi.org/10.1007/bf02295996 doi: 10.1007/bf02295996

Nabipour, M., Nayyeri, P., Jabani, H., Mosavi, A., Salwana, E. & S., S. (2020, July). Deep learning for stock market prediction. *Entropy*, *22*(8), 840. Retrieved from https://doi.org/10.3390/e22080840 doi: 10.3390/e22080840

Pan, J., Zhuang, Y. & Fong, S. (2016). The impact of data normalization on stock market prediction: Using svm and technical indicators. In M. W. Berry, A. Hj. Mohamed & B. W. Yap (Eds.), *Soft computing in data science* (pp. 72–88). Singapore: Springer Singapore.

Patel, J., Shah, S., Thakkar, P. & Kotecha, K. (2015, January). Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques. *Expert Systems with Applications*, *42*(1), 259–268. Retrieved from https://doi.org/10.1016/j.eswa.2014.07.040 doi: 10.1016/j.eswa.2014.07.040

Pesaran, M. H. & Timmermann, A. (2009, March). Testing dependence among serially correlated multicategory variables. *Journal of the American Statistical Association*, *104*(485), 325–

337. Retrieved from https://doi.org/10.1198/jasa.2009.0113 doi: 10.1198/jasa.2009.0113

Pindyck, R. S. (2004). Volatility and commodity price dynamics. *Journal of Futures Markets: Futures, Options, and Other Derivative Products*, *24*(11), 1029–1047.

Pindyck, R. S. & Rotemberg, J. J. (1990). The excess co-movement of commodity prices. *The Economic Journal*, *100*(403), 1173–1189.

Sezer, O. B., Gudelek, M. U. & Ozbayoglu, A. M. (2020, May). Financial time series forecasting with deep learning : A systematic literature review: 2005–2019. *Applied Soft Computing*, *90*, 106181. Retrieved from https://doi.org/10.1016/j.asoc.2020.106181 doi: 10.1016/j.asoc.2020.106181

Tay, F. E. & Cao, L. (2001, August). Application of support vector machines in financial time series forecasting. *Omega*, *29*(4), 309–317. Retrieved from https://doi.org/10.1016/s0305-0483(01)00026-3 doi: 10.1016/s0305-0483(01)00026-3

Wang, Y., Liu, L. & Wu, C. (2020). Forecasting commodity prices out-of-sample: Can technical indicators help? *International Journal of Forecasting*, *36*(2), 666-683. Retrieved from https://www.sciencedirect.com/science/article/pii/S0169207019302286 doi: https://doi.org/10.1016/j.ijforecast.2019.08.004

World Bank. (2023). *The World Bank*. Retrieved from https://thedocs.worldbank.org/en/doc/5d903e848db1d1b83e0ec8f744e55570-0350012021/related/CMO-Historical-Data-Monthly.xlsx (Accessed on 18 July 2023)

A Programming code

`github.com/ka-wah/commodity-forest-support`

The repository contains a readme file with instructions for replication. The data is divided into input and output, with input being the commodity prices and output all results obtained from each model and also evaluation measures. Data_Preprocess.py preprocesses the commodity prices. The Binary_Indicators.py and Nonbinary_Indicators.py create the technical indicators for their respective encoding. Least_Squares.py, Random_Forest.py and Support_Vector.py contain the models to create the predictions. Evaluation.py and Significance.py contain functions to evaluate the predictions and test for significance. Lastly, Plots.py has the functions to create the plots as seen in this paper.

B Algorithms

Algorithm 1: Random Forest with expanding window

Input:
- data: complete dataset
- *num_trees*: the number of decision trees in each forest
- $num\_features$: the number of features considered at each split

Output: *oos_predictions*: out-of-sample predictions

```
oos_predictions = empty list
for commodity in commodities do
    while windowsize ≤ length(data) do
        forest = empty list
        current_data = data[0:windowsize]
        test_date = data[windowsize]
        predictions = empty list
        for i ≤ num_trees do
            selected_features = random subset of num_features features
            bootstrap_samples = random subset of instances
            tree = decision_tree(bootstrap_samples, selected_features)
            forest.add(decision_tree)
        end
        tree_predictions = empty list
        for tree in forest do
            tree_prediction = tree.predict(test_date)
            tree_predictions.add(tree_prediction)
        end
        oos_predictions.add(majority_voting(tree_predictions))
        windowsize = windowsize + 1
    end
end
return oos_predictions
```

Algorithm 2: SVM with expanding window

Input:

- data: complete dataset
- C: the regularization parameter
- *gamma*: the constant

Output: *oos_predictions*: out-of-sample predictions

```
oos_predictions = empty list
for commodity in commodities do
    while windowsize ≤ length(data) do
        current_data = data[0 : windowsize]
        test_date = data[windowsize]
        svm = svm_model(rbf, C, gamma)
        svm.fit(current_data)
        prediction = svm.predict(test_date)
        oos_predictions.add(prediction)
        windowsize = windowsize + 1
    end
end
return oos_predictions
```

C Tables

C.1 Pesaran-Timmermann statistics

Table 16: Binary Random Forest

f/t	100	250	500
6	6.725	6.982	6.217
12	7.311	6.637	5.953
18	5.596	6.194	6.687

Note: all statistically significant with $p < 10^{-8}$

Table 17: Non-binary Random Forest

f/t	100	250	500
6	5.925	5.722	5.808
12	5.699	6.199	6.366
18	5.702	6.652	6.389

Note: all statistically significant with $p < 10^{-8}$

Table 18: Binary SVM

C/γ	2	2.5	3	3.5	4
1	6.107	5.783	5.093	5.008	5.084
10	5.873	5.684	5.578	5.200	5.149
100	5.873	5.684	5.578	5.200	5.149

Note: all statistically significant with $p < 10^{-6}$

Table 19: Non-binary SVM

	2	2.5	3	3.5	4
1	4.161	3.995	3.905	3.498	3.064
10	4.629	4.100	4.131	4.240	3.868
100	4.629	4.062	4.131	4.240	3.868

Note: all statistically significant with $p < 0.001$

C.2 McNemar

Table 20: Model p-values comparison

		OLS		RF		SVM	
		Binary	Non-Binary	Binary	Non-Binary	Binary	Non-Binary
OLS	Binary	-	0.227	0.631	0.315	0.142	0.009
	Non-Binary	-	-	0.308	0.118	0.036	0.002
RF	Binary	-	-	-	0.606	0.355	0.028
	Non-Binary	-	-	-	-	0.679	0.064
SVM	Binary	-	-	-	-	-	0.214
	Non-Binary	-	-	-	-	-	-