Stock Forecasting using Features Decomposition based Machine Learning Methods

Hieu Nguyen[1] , **Jihye Moon**[2]
[1]Department of Computer Science and Engineering
[2]Department of Biomedical Engineering
University of Connecticut, Storrs, CT
{hieu.nguyen, jihye.moon}@uconn.edu

Abstract

We propose a features decomposition method based on wavelet transform with machine learning models for forecasting price movements in the stock market. We augment the decomposition approach improves the performance in forecasting short-term stock movements as a novel approach that uses high-frequency signals (noise) as predictors. We collect popular three ETFs data sets, which are SP500 (SPY), Dow Jones Industrial Average (DIA), and NASDAQ (QQQ) recorded from January 1st, 2010 to January 1st, 2020. This approach decomposes the feature signals from those data sets into high and low frequencies, respectively. We exploit the decomposed features in forecasting the short-term stock movements using both traditional and deep machine learning models. Extensive results are presented on MAE, RMSE, and accuracy metrics for predicting either up or down in stock. The results clearly indicate that the prediction results using the feature decomposition approach outperform those using original signals.

1 Introduction

Time series analysis has been an appealing domain in many different fields such as economics, social sciences, medicine, physical sciences, and finances despite being challenging tasks. In recent years, accurately forecasting the movement of stocks/indices has become an ever-increasingly issue in investment/trading decision making. The goal was to analyze and understand historical data and potentially predict future outcomes. As many trading platforms are becoming commission-free, stock forecasting attracts interests not only for financial institutions and hedge funds for also for retail investors.

A well-known time series analysis method is the autoregressive moving average (ARIMA) model [1]. ARIMA has been a successful model in various time series applications. However, the model does not perform well on high dimensional data, and other variances of the ARIMA model was developed to tackle this problem such as the nonlinear autoregressive exogenous(NARX) models [2].

Due to the stock market being nonlinear, non-stationary, and contains lots of noise, predicting stock movements is a difficult task. Recently, machine learning models have become popular due to their ability to solve complex and nonlinear problems such as in computer vision or natural language processing [3][4][5]. As technology is growing with a rapid speed, applications in the financial domain are shifting toward incorporating machine learning methods into their models.

Researchers in time series forecasting has been using traditional machine methods such as decision trees, random forests, and support vector machines for more than 40 years. With the recent developments of deep learning, many studies has been published and showed that deep learning models outperform other machine learning counterparts [6] [7] [8] [9] [10] [11]. The long short-term memory (LSTM) model is a popular method in natural language application has been extended to time series application due to its forecasting nature [12] [13]. The advantage of machine learning methods is that it can learn the non-linear mapping between the features, and the model outperforms traditional methods when inputs are high-dimensional data. However, since many applications of time-series forecasting only have a small amount of data, the deep learning model tends to overfit. Thus, researchers come up with hybrid methods to combine the best of both worlds where traditional methods would learn the linear relationship, and deep learning methods learn the residuals [14] [15].

We take another approach and theorize that for short-term movement of the stock market, the high-frequency signals and noise are the causes for the short-term fluctuation. To tackle this challenge, we utilize wavelet transform as a signal processing method to decompose the feature signals into high and low-frequency signals, and model them separately using both traditional and advanced machine learning methods. We believe that using the high-frequency band signals would result in better performance in terms of predicting short-term market movement.

Existing researches use the wavelet transform and other signal processing methods to denoise the signal and sometimes results in an incremental improvement in forecasting using low-frequency band signal as predictor [16] [17]. To the best of our knowledge, we are the first to utilize high-frequency band signals of the features for short-term stock forecasting, and our proposed methodology yield a signifi-

cant improvement. The summary of our contribution is as follows:

- Using 4-band wavelet transform to get one low-frequency signal three different high-frequency signals.
- Better accuracy performance when using high-frequency signals of features.
- Using frequency signals yields a better convergence speed when training deep learning methods.

The remaining of the paper is as follows: Section 2 describes the proposed methodology, Section 3 presents results, and followed by a conclusion.

2 Proposed Method

Since financial time series data contain noise, this makes forecasting task a very challenging task. In our research, we utilize a signal processing method called wavelet transform to decompose the time series data into low and high frequencies. The low frequency contains the approximation of the original signal, and the high frequencies represent the details and noise.

Our proposed method mainly focus on decomposing the features into low and high-frequency signals, and use off-the-shelf machine learning models to each signal then compare their performance. We believe that short-term market movements tend to be influenced by the high-frequency signals where the details and the noise cause the market fluctuation.

2.1 Features Decomposition

One of many signal processing methods is the Fourier Transform. However, when apply Fourier Transform to get into the frequency-domain, all of the time information is lost. Since our data is time-series data and our signal is vary by time, we must preserve the time information within the frequency-domain.

Wavelet Transform is a solution for the aforementioned challenge as it fulfills both conditions of decomposing the signal into the frequency domain and preserve the time information [18][17]. For our experiment, we construct a 4-Band wavelet so that we can break down our features data. Below is the filter banks for our 4-Band Wavelet:

$$\alpha = [-0.067371, 0.094195, 0.405805, 0.567372, 0.567372, 0.405805, 0.094195, -0.067372] \quad (1)$$

$$\beta = [-0.094195, 0.067372, 0.567372, 0.405805, -0.405805, -0.567372, -0.067372, 0.094195] \quad (2)$$

$$\gamma = [-0.094195, -0.067372, 0.56737, -0.405805, -0.405805, -0.56737, -0.067372, -0.094195] \quad (3)$$

$$\delta = [-0.067372, -0.094195, 0.405805, -0.567372, 0.567372, -0.405805, 0.094195, 0.067372] \quad (4)$$

where α is the low pass filter bank, and β, γ, δ are the high pass filter banks such that they satisfy the following conditions:

$$\sum_{i=1}^{8} \alpha_i = 2 \quad (5)$$

$$\sum_{i=1}^{8} \beta_i = \sum_{i=1}^{8} \gamma_i = \sum_{i=1}^{8} \delta_i = 0 \quad (6)$$

$$|\alpha| = |\beta| = |\gamma| = |\delta| \quad (7)$$

$$\alpha \cdot \beta = \alpha \cdot \gamma = \alpha \cdot \delta = \beta \cdot \gamma = \beta \cdot \delta = \gamma \cdot \delta = 0 \quad (8)$$

An example where $S \in \mathbb{R}^{4^k} (k \in \mathbb{N}, k \geq 2)$, a 4-Band Wavelet Transform matrix T_1 (See Fig. 1) is constructed by shifting and wrapping around the filter banks. Let call our

$$\begin{bmatrix}
\alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & \alpha_5 & \alpha_6 & \alpha_7 & \alpha_8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & \alpha_5 & \alpha_6 & \alpha_7 & \alpha_8 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & \alpha_5 & \alpha_6 & \alpha_7 & \alpha_8 \\
\alpha_5 & \alpha_6 & \alpha_7 & \alpha_8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 \\
\beta_1 & \beta_2 & \beta_3 & \beta_4 & \beta_5 & \beta_6 & \beta_7 & \beta_8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \beta_1 & \beta_2 & \beta_3 & \beta_4 & \beta_5 & \beta_6 & \beta_7 & \beta_8 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \beta_1 & \beta_2 & \beta_3 & \beta_4 & \beta_5 & \beta_6 & \beta_7 & \beta_8 \\
\beta_5 & \beta_6 & \beta_7 & \beta_8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \beta_1 & \beta_2 & \beta_3 & \beta_4 \\
\gamma_1 & \gamma_2 & \gamma_3 & \gamma_4 & \gamma_5 & \gamma_6 & \gamma_7 & \gamma_8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \gamma_1 & \gamma_2 & \gamma_3 & \gamma_4 & \gamma_5 & \gamma_6 & \gamma_7 & \gamma_8 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \gamma_1 & \gamma_2 & \gamma_3 & \gamma_4 & \gamma_5 & \gamma_6 & \gamma_7 & \gamma_8 \\
\gamma_5 & \gamma_6 & \gamma_7 & \gamma_8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \gamma_1 & \gamma_2 & \gamma_3 & \gamma_4 \\
\delta_1 & \delta_2 & \delta_3 & \delta_4 & \delta_5 & \delta_6 & \delta_7 & \delta_8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \delta_1 & \delta_2 & \delta_3 & \delta_4 & \delta_5 & \delta_6 & \delta_7 & \delta_8 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \delta_1 & \delta_2 & \delta_3 & \delta_4 & \delta_5 & \delta_6 & \delta_7 & \delta_8 \\
\delta_5 & \delta_6 & \delta_7 & \delta_8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \delta_1 & \delta_2 & \delta_3 & \delta_4
\end{bmatrix}$$

Figure 1: An example of $4^2 \times 4^2$ Wavelet Transform matrix

wavelet matrix $W \in \mathbb{R}^{4^k} \times \mathbb{R}^{4^k}$ and it is an orthonormal matrix since the column and row vector of W form a set of the orthonormal basis for $\mathbb{R}^{4k}$. To get into the frequency domain (F), we apply wavelet transformation to the signal by simple perform a matrix multiplication:

$$F = W \cdot S = [a \;\; d_1 \;\; d_2 \;\; d_3]^T \in \mathbb{R}^{4^k}$$

where $a = [a_1, a_2, a_3, ..., a_{4^{k-1}}]$ is the low frequency component, and $d_i = [d_{i,1}, d_{i,2}, d_{i,3}, ..., d_{i,4^{k-1}}] (i = 1, 2, 3)$, are the high frequency components. We separate each frequency component by setting other frequency components to zero.

$$A = [a \;\; 0 \;\; 0 \;\; 0]^T$$

$$D_1 = [0 \;\; d_1 \;\; 0 \;\; 0]^T$$

$$D_2 = [0 \;\; 0 \;\; d_2 \;\; 0]^T$$

$$D_3 = [0 \;\; 0 \;\; 0 \;\; d_3]^T$$

We apply the inverse of wavelet transform to the frequency component to get back to the signal domain.

$$S_a = W^T \cdot A$$

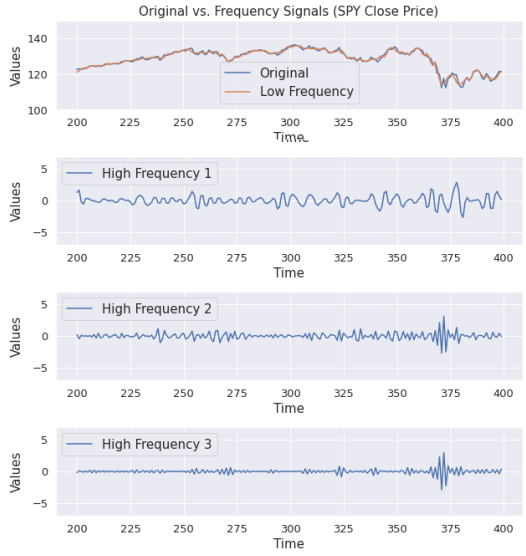

Figure 2: Signal Representations by high/low frequency components

$$S_{d_i} = W^T \cdot D_i$$

so that $S = S_a + \sum_{i=1}^{3} S_{d_i}$ where S_a approximate S and S_{d_i} represents the details and approximate noise of S. See Fig 2 for an example of this process.

2.2 Machine Learning Models

After we decompose our features into four sets of data, we run each data set separately using both traditional machine learning methods and deep learning methods to compare the performance. Below are short descriptions of the methods that are being used in this study:

Linear Regression: Linear Regression (LR) is to find the curve that best fits the data with the combination of coefficients and variables, which describes the relationship between the dependent and independent variable. The model estimates the best curve by minimizing the residual sum of squares between the labels provided by the data and the targets predicted by the linear approximation [19].

Decision Tree: A decision tree (DT) is constructed start from the root node. Each node represents an output class and every branch represents the process that leads to the decision output, and the end node is the result [20].

Random Forest: Random forest (RF) is an extension of Bootstrap aggregating by using decision tree where it helps improves the stability accuracy of the algorithm. Due the the fact the decision trees have a high variance and the model tend to over-fit if the training data is complex and present irregular pattern or contain a lot of noise [21] [10].

eXtreme Gradient Boosting: The eXtreme Gradient Boosting (XGBoost) is an ensemble scalable end-to-end tree boosting algorithm, which applies the boosting for weak learners to convert them to strong learners[22]. The XGBoost generates individual trees using multiple cores, organize each data to minimize the lookup times in order to get better performance and speed. The model has provided in-built cross-validation ability, efficient handling of missing data, regularization for avoiding over-fitting, catch awareness, tree pruning, and parallelized tree building [23].

Support Vector Machine: Support Vector Machine (SVM) is a non-parametric kernel-based regression method used for extrapolating future values [24] [25] [26]. More specifically, we shall be focusing on ϵ-SVR, a form of SVR in which a hyperplane is constructed with a loss function within precision. The SVR function can be expressed as below:

$$f(x) = w^T \phi(x) + b$$

where $\phi(x)$ maps data from the input space to the feature space, w is a weight vector, and b is a bias constant. w and b are estimated by satisfying as follows:

Minimizing: $\frac{1}{2}\|x\|^2$

Subject to:

$$y_i - (\langle w, \phi(x_i)\rangle + b) \leq \epsilon$$
$$(\langle w, \phi(x_i)\rangle + b - y_i \leq \epsilon$$

where x_i and y_i represent input and target values obtained from the training set. To address points outside this ϵ-insensitive band, we introduce slack variables ξ_i, ξ_i^*:

Minimizing: $\frac{1}{2}\|x\|^2 + C\sum_i^n = 1(\xi_i + \xi_i^*)$

Subject to:

$$y_i - (\langle w, \phi(x_i)\rangle + b) \leq \epsilon + \xi_i$$
$$(\langle w, \phi(x_i)\rangle + b - y_i \leq \epsilon + \xi_i^*$$
$$\xi_i, \xi_i^* \leq 0$$

The constant $C > 0$ is used to represent the trade off between model complexity and training error. After taking the Lagrangian and optimizing with the above constraints, we are left with:

$$f(x) = \sum_{i=1}^{n} (\alpha_i - \alpha_i^*) K(x_i, x) + b$$

where $K(\cdot, \cdot)$ is a kernel function, α_i and α_i^* are nonzero Lagrangian multipliers and solutions to the dual problem.

Long Short Term Memory: The Long Short Term Memory(LSTM) contains special units called memory blocks in the recurrent hidden layer. These memory blocks contain memory cells with self-connections storing the temporal state of the network in addition to special multiplicative units called gates to control the flow of information. Each memory block in the original architecture contained an input gate and an output gate. The input gate controls the flow of input activation into the memory cell. The output gate controls the output flow of cell activation into the rest of the network. Later, the forget gate was added to the memory block. This addressed a weakness of LSTM models preventing them from processing continuous input streams that are not segmented into sub-sequences. The forget gate scales the internal state of the cell before adding it as an input to the cell through the self-recurrent connection of the cell, therefore adaptively forgetting or resetting the cell's memory. In addition, the modern LSTM architecture contains peephole connections from its internal cells to the gates in the same cell to learn the precise timing of the outputs [5]. An LSTM network maps

a sequence of input vectors $x = (x_1, ..., x_t)$ to a sequence of output vectors $y = (y_1, ..., y_t)$ by iteratively calculating the network unit activation using the following equation from $t = 1...T$:

$$
\begin{array}{lll}
\text{Block input} & : & z_t = g(W_z x_t + V_z y_{t-1} + b_t) \\
\text{Input gate} & : & i_t = \sigma(W_i x_t + V_i y_{t-1} + b_i) \\
\text{Forget gate} & : & f_t = \sigma(W_f x_t + V_f y_{t-1} + b_f) \\
\text{Memory cell} & : & c_t = i_t \odot z_t + f_t \odot c_{t-1} \\
\text{Output gate} & : & o_t = \sigma(W_o x_t + V_o y_{t-1} + b_o) \\
\text{Block output} & : & y_t = o_t \odot g(c_t)
\end{array}
$$

where W and V denote input and recurrent weight matrices, respectively (e.g. W_i the weight matrix from the current input step to input gate i and V_i the recurrent weight matrix from the output of the previous step to input gate i the b terms denote bias weight vectors (e.g.b_i is the input gate bias weight vector), σ is the logistic sigmoid function, $g(x) = tanh(x)$, and i, f, o, c are output gate and memory cell activation vectors, respectively, all of which are the same size as the cell output activation vector m, $\odot$ is the element-wise product of the vectors.

Convolutional Network-Long Short Term Memory: The convoultional Network-Long Short Term Memory (CNN-LSTM) is one of variants of the LSTM model [27]. The CNN-LSTM utilizes the output of the convolutional networks as the input of the LSTM. A CNN network is formed using convolutional layers which perform convolutional operation. The model plays a huge role in reducing the parameters using filters, max-pooling, dropout and fully connected layers so that the network only extract the most meaningful features from input data.

3 Experiments

3.1 Data Collection

For our experiments, we are collecting three U.S. ETFs which are **SPY, DIA, and QQQ** which track the major indices of the SP 500, Dow Jones Industrial Average, and NASDAQ respectively. We collect the daily quote data from January 1st, 2010 to January 1st, 2020 using Yahoo! Finance API [28]. The features used in these datasets consist of $Open, High, Low, Close, Volume$ and technical indicator computed from $Close$ prices which are *Moving Average Convergence Divergence (MACD), Bollinger bands (BBANDS), Relative Strengh Index (RSI)*. Our target is the percentage change of $Close$ price for the next m days. Thus, the goal is to use n-previous days of features to predict the percentage change of $Close$ price for the next m days. The baseline formulation before wavelet transforms is below:

$$T_m = f(X_0, X_{-1}, ..., X_{-(n-1)}, X_{-n})$$

Where T_m is the the percentage change of $Close$ price for the next m days, X is the set of all the features, and $f(\cdot)$ is the non-linear mapping function. We then split then data in the sequential order to training/testing sets. For this experiment, 70% of the data was utilized as the training sets. The remaining 30% was used for test sets. Technical indicators are often used by many traders to identify the change of patterns. The most popular ones are Moving Averages, Bollinger bands, and Relative Strength Index. Many existing research only use Open-High-Low-Close (OHLC) values, but we believe that technical indicators are types of feature engineering methods, and they would be more useful than OHLC. Below is the description of our features:

Features	Description
MACD	A trend-following momentum indicator.
RSI	An oscillator that indicates the internal strength of a signal
BBAND	Standard deviation level above/below a simple moving average
Volume	Number of shares traded
High	Highest price reached in the day
Low	Lowest price reached in the day
Open	The price of the stock opened at market
Close	Close price adjusted for splits
Adj Close	Adjusted close price adjusted for both dividends and splits

Table 1: List of features

3.2 Feature Importance

This paper utilizes the Random Forest Regression model to estimate the importance of those features. The feature importance analysis is used to determine which features are more useful. It is calculated as the decrease in node impurity weighted by the probability of reaching that node. The node probability is calculated by the number of samples that reach the node, divided by the total number of samples. The higher the value the more important the feature. As shown in Fig. 3, the technical features are sorted as the most contributing in forecasting for the stock movements.

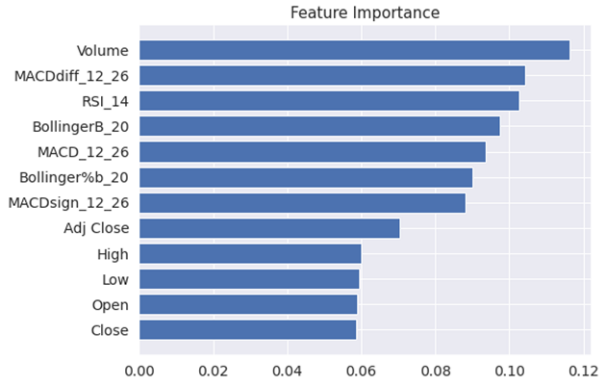

Figure 3: The Feature Importance using the Random Forest Regression Model

From the figure above, all of the technical indicators we use are more important compare to the OHLC features. Since there are hundreds of technical indicators out there, it is important that we pick and design the right ones that best fit the model. Further research will explore and include more technical indicators and external features.

3.3 Model Training

We train eight machine learning models using all stock features including decomposed features, respectively; LR, DT,

RF, GB, XGB, SVR, LSTM, and CNN-LSTM. All parameters of the machine learning models are set by the default values of the sklearn Library. The LSTM consists of four LSTM layers with 50 units and four drop-outs of 0.2. The drop-out is located after each LSTM layer. For CNN-LSTM, three 1-dimensional CNNs which consist of 5 kernel sizes, 1 stride size, and one LSTM layer with 30 units, and one drop-out of 0.2 are utilized. The filter size of the CNN layers varies 64, 64, and 32 for each layer. As hyper-parameters, for both LSTM and CNN-LSTM models, Adam optimizers are selected with 0.001 of learning rate. They are trained using Python3 on Google Colab.

3.4 Evaluation Metrics

We evaluate our approach by using multiple metrics: Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and accuracy. The MAE method represents the difference between the actual and the predicted values over the data set by averaging the absolute difference. The RMSE is the error rate by the square root of the mean squared error between actual and predicted values.

$$MAE = \frac{\sum_{i=1}^{n} |\bar{y_i} - y_i|}{n}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\bar{y_i} - y_i)^2}$$

where n is the number of data points, y_i is the observed values, and $\bar{y_i}$ is the predicted values.

This paper utilizes the accuracy as the binary classification, which estimates either down or up of the stock movements. We define the down as 0 when $\bar{y_i}$ and y_i are less or equal to 0, up as 1 and $\bar{y_i}$ and y_i are more than 0 like below equations. The accuracy is calculated by the sum of true negative (TN) and true positive (TP) over test sets. The TN is the number of that $\bar{y_i}$ and y_i are correctly identified as down, and the TP is the number of that $\bar{y_i}$ and $y_i$are correctly identified as up.

$$y_i, \bar{y_i} = \begin{cases} 0 & y_i, \bar{y_i} \leq 0 \\ 1 & y_i, \bar{y_i} > 0 \end{cases}$$

$$Accuracy = \frac{True\ Negatives(TN) + True\ Positives(TP)}{All\ Samples}$$

3.5 Numerical Experiments

In this experiment, the input features for the baseline models are MACD, RSI, BBAND, and Volume from the feature importance analysis. For our proposed method, we decompose each feature into low and high-frequency signals using wavelet transformation. We then run a series of statistical and machine learning models for each set of features and calculate their performance using the metrics mentioned earlier.

Predicting Price One Day Ahead

From our results (See Tables 2, 3, 4), high-frequency signals 2 perform the best when using Support Vector Regression. For all three ETFs, the accuracy is above 72%, and the errors (MAE and RMSE) are the lowest compare to other traditional statistic and machine learning models. It is also worth pointing out that when using frequency signals, these models also outperform the baseline models.

Predicting Price Multiple Days Ahead

To confirm our hypothesis that high-frequency signals have a more significant influence on the short-term stock movement, we run experiments for multiple days ahead. Below are the comparisons using the best methods (Fig. 4). From

Figure 4: Accuracy of predicting multiple days ahead using Support Vector Machine

the plot above, we learn that the performances of the models that use high-frequency features drop when trying to predict more than three days. The results gave us another confirmation short-term stock movement is more influenced by noise which is explained in the high-frequency signals. From the earlier experiment using deep learning methods, the accuracy for one-day movement prediction is almost a coin-toss. However, as we predict multiple days, the accuracy increases. We do not have an explanation for this behavior; this is left for future research. Also, noted that as we try to predict a long term period, the errors increase due to a higher level of uncertainty.

Method	Baseline			Low Frequency Signals			High Frequency Signals #1			High Frequency Signals #2			High Frequency Signals #3		
	MAE	RMSE	Acc.(%)	MAE	RMSE	Acc.(%)	MAE	RMSE	Acc.(%)	MAE	RMSE	Acc.(%)	MAE	RMSE	Acc.(%)
LR	0.544	0.816	48.32	0.542	0.804	55.57	0.550	0.824	57.99	**0.496**	**0.723**	**69.53**	0.527	0.772	64.16
DT	0.572	0.935	55.97	0.584	0.981	56.78	0.538	0.824	47.38	**0.507**	**0.713**	**67.79**	0.561	0.836	64.56
RF	0.848	1.247	49.80	0.805	1.165	53.15	0.758	1.100	52.48	**0.701**	**0.995**	**65.37**	0.736	1.073	58.12
GB	0.606	0.948	48.46	0.569	0.880	55.17	0.591	0.905	58.79	**0.538**	**0.797**	**69.53**	0.555	0.839	65.10
XGB	0.558	0.834	55.84	0.558	0.832	56.24	0.554	0.823	55.57	**0.523**	**0.785**	**59.60**	0.541	0.806	57.58
SVR	0.547	0.830	52.48	0.534	0.805	58.66	0.543	0.820	58.26	**0.444**	**0.683**	**72.35**	0.515	0.771	64.16
LSTM	0.539	0.813	52.37	0.550	0.805	49.19	0.537	0.805	50.97	0.507	0.735	50.55	**0.523**	**0.761**	**52.59**
CNN LSTM	**0.551**	**0.841**	**55.11**	0.573	0.877	52.73	0.572	0.876	53.05	0.704	1.409	50.75	0.557	0.854	52.11

Table 2: SPY RESULTS COMPARISON

Method	Baseline			Low Frequency Signals			High Frequency Signals #1			High Frequency Signals #2			High Frequency Signals #3		
	MAE	RMSE	Acc.(%)	MAE	RMSE	Acc.(%)	MAE	RMSE	Acc.(%)	MAE	RMSE	Acc.(%)	MAE	RMSE	Acc.(%)
LR	0.556	0.834	47.38	0.549	0.822	57.99	0.562	0.842	55.57	**0.518**	**0.750**	**69.66**	0.537	0.785	65.37
DT	0.596	0.972	55.70	0.554	0.831	57.05	0.576	0.875	54.23	**0.520**	**0.725**	**69.66**	0.540	0.799	66.31
RF	0.811	1.185	52.08	0.764	1.114	55.84	0.802	1.105	52.89	**0.668**	**0.949**	**64.56**	0.698	0.979	60.94
GB	0.623	0.983	49.80	0.592	0.949	57.72	0.609	0.941	57.18	**0.561**	**0.844**	**70.34**	0.561	0.842	65.91
XGB	0.565	0.847	55.44	0.573	0.857	55.84	0.561	0.830	55.17	**0.534**	**0.803**	**59.73**	0.550	0.829	57.45
SVR	0.565	0.853	55.03	0.550	0.832	57.05	0.554	0.832	58.26	**0.463**	**0.708**	**72.08**	0.519	0.777	65.50
LSTM	**0.554**	**0.832**	52.26	0.556	0.821	49.67	0.560	0.841	**52.76**	0.554	0.779	49.92	0.524	0.780	50.79
CNN LSTM	0.579	0.919	**53.01**	0.578	0.849	49.15	**0.571**	**0.873**	51.29	0.680	1.366	50.22	0.557	0.850	52.14

Table 3: DIA RESULTS COMPARISON

Method	Baseline			Low Frequency Signals			High Frequency Signals #1			High Frequency Signals #2			High Frequency Signals #3		
	MAE	RMSE	Acc.(%)	MAE	RMSE	Acc.(%)	MAE	RMSE	Acc.(%)	MAE	RMSE	Acc.(%)	MAE	RMSE	Acc.(%)
LR	0.755	1.096	50.07	0.542	0.804	55.57	0.550	0.824	57.99	**0.496**	**0.723**	**69.53**	0.527	0.772	64.16
DT	0.760	1.138	57.85	0.584	0.981	56.78	0.538	0.824	47.38	**0.507**	**0.713**	**67.79**	0.561	0.836	64.56
RF	0.940	1.320	50.87	0.805	1.165	53.15	0.758	1.100	52.48	**0.701**	**0.995**	**65.37**	0.736	1.073	58.12
GB	0.819	1.199	51.95	0.569	0.880	55.17	0.591	0.905	58.79	**0.538**	**0.797**	**69.53**	0.555	0.839	65.10
XGB	0.746	1.096	57.72	0.558	0.832	56.24	0.554	0.823	55.57	**0.523**	**0.785**	**59.60**	0.541	0.806	57.58
SVR	0.747	1.109	56.64	0.534	0.805	58.66	0.543	0.820	58.26	**0.444**	**0.683**	**72.35**	0.515	0.771	64.16
LSTM	0.736	1.087	**57.99**	0.541	0.800	51.21	0.547	0.806	49.38	**0.487**	**0.710**	51.48	0.522	0.761	51.82
CNN LSTM	0.796	1.244	57.69	0.586	0.916	49.72	**0.541**	**0.825**	**51.98**	0.660	1.277	49.49	0.585	0.884	49.94

Table 4: QQQ RESULTS COMPARISON

LSTM Convergence Speed

Another discovery we found while running experiments with LSTM was that the training error converges much faster and consistent when using the frequency-based signal. An example comparison is between the results using original signals vs. high-frequency signal 2. From Fig. 5, we can observe that the convergence speed when training with high-frequency signals 2 is faster. This can potentially be applied to other applications where the data is noisy.

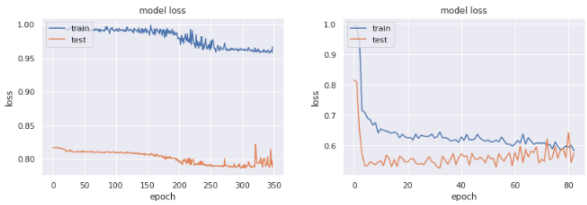

Figure 5: Training loss using original signal (left) vs. high frequency signals 2 (right)

4 Conclusion & Future Work

In this paper, we study the use of wavelet transformation to the features to decompose them into low/high-frequency signals and use them as input for our forecasting models. As demonstrated in our experiments, high-frequency signals of the features yield better short-term prediction results of the three major (ETFs SPY, DIA, and QQQ) in terms of errors and accuracy in comparison to using the original signals. The experiment results show that short-term market movements are caused by many external noises lie within the high-frequency signals. Thus, being able to separate the noise for the signal and use it to forecast the short-term stock market could lead to higher capital gain when combined with a good method of risk management.

Another discovery we have made while running the experiment was that by using the low-frequency signal which closely represents the original signal, the training of deep neural networks converges much faster. This would potentially be a potential data denoising method that can be applied to other applications.

Future studies will include more data and features and explore the potential of machine learning methods. In the experiment, we observe that traditional statistical and machine learning methods outperform deep learning. The main reason is our data is not big and complex enough to fully utilize a deep machine learning model, and there is a potential for overfitting. When including more features into the model, feature selection/extraction will be also included to avoid the curse of dimensional and reduce the cost of computation. Furthermore, we would also like to include portfolio optimization based on the proposed prediction methods.

References

[1] Jamal Fattah, Latifa Ezzine, Zineb Aman, Haj Moussami, and Abdeslam Lachhab. Forecasting of demand using arima model. *International Journal of Engineering Business Management*, 10:184797901880867, 10 2018.

[2] Zina Boussaada, Octavian Curea, Remaci Ahmed, Haritza Camblong, and mrabet bellaaj Najiba. A nonlinear autoregressive exogenous (narx) neural network model for the prediction of the daily direct solar radiation. *Energies*, 11:620, 03 2018.

[3] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[5] Haşim Sak, Andrew Senior, and Françoise Beaufays. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *arXiv preprint arXiv:1402.1128*, 2014.

[6] X.; Su J.; Tang B.; Wu S. Wu, D.; Wang. A labeling method for financial time series prediction based on trends. *Journal of Entropy*, 22:1162, 2020.

[7] Eunsuk Chong, Chulwoo Han, and Frank C Park. Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies. *Expert Systems with Applications*, 83:187–205, 2017.

[8] Kai Chen, Yi Zhou, and Fangyan Dai. A lstm-based method for stock returns prediction: A case study of china stock market. In *2015 IEEE international conference on big data (big data)*, pages 2823–2824. IEEE, 2015.

[9] Lin Chen, Zhilin Qiao, Minggang Wang, Chao Wang, Ruijin Du, and Harry Eugene Stanley. Which artificial intelligence algorithm better predicts the chinese stock market? *IEEE Access*, 6:48625–48633, 2018.

[10] Christopher Krauss, Xuan Anh Do, and Nicolas Huck. Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the s&p 500. *European Journal of Operational Research*, 259(2):689–702, 2017.

[11] Omer Berat Sezer, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu. Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Applied Soft Computing*, 90:106181, 2020.

[12] Abdul Hasib Rahimyar, Hieu Quang Nguyen, and Xiaodi Wang. Stock forecasting using m-band wavelet-based svr and rnn-lstms models. In *2019 2nd International Conference on Information Systems and Computer Aided Education (ICISCAE)*, pages 234–240. IEEE, 2019.

[13] Qun Zhuge, Lingyu Xu, and Gaowei Zhang. Lstm neural network with emotional analysis for prediction of stock price. *Engineering letters*, 25(2), 2017.

[14] Ping-Feng Pai and Chih-Sheng Lin. A hybrid arima and support vector machines model in stock price forecasting. *Omega*, 33(6):497–505, 2005.

[15] Manish Kumar and M Thenmozhi. Forecasting stock index returns using arima-svm, arima-ann, and arima-random forest hybrid models. *International Journal of Banking, Accounting and Finance*, 5(3):284–308, 2014.

[16] Hieu Q Nguyen and Xiaodi Wang. Pseudo quantum steganography with "color barcode" in m-band wavelet domain. *International Journal of Signal Processing*, 1:160–168, 2016.

[17] Zuohong Pan, Xiaodi Wang, et al. A wavelet-based nonparametric estimator of the variance function. *Computational Economics*, 15(1/2):79–87, 2000.

[18] Peter Steffen, Peter N Heller, Ramesh A Gopinath, and C Sidney Burrus. Theory of regular m-band wavelet bases. *IEEE Transactions on Signal Processing*, 41(12):3497–3511, 1993.

[19] Nicola Uras, Lodovica Marchesi, Michele Marchesi, and Roberto Tonelli. Forecasting bitcoin closing price series using linear regression and neural networks models, 01 2020.

[20] Lior Rokach and Oded Maimon. *Decision Trees*, volume 6, pages 165–192. 01 2005.

[21] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[22] Fei Z et al. Liu L, Yu Y. An interpretable boosting model to predict side effects of analgesics for osteoarthritis.

[23] M. Nabipour, P. Nayyeri, H. Jabani, A. Mosavi, E. Salwana, and Shahab S. Deep learning for stock market prediction. *Entropy*, 22(8):840, Jul 2020.

[24] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf. Support vector machines. *IEEE Intelligent Systems and their Applications*, 13(4):18–28, 1998.

[25] Francis EH Tay and Lijuan Cao. Application of support vector machines in financial time series forecasting. *omega*, 29(4):309–317, 2001.

[26] Wei Huang, Yoshiteru Nakamori, and Shou-Yang Wang. Forecasting stock market movement direction with support vector machine. *Computers & operations research*, 32(10):2513–2522, 2005.

[27] Shuanglong Liu, Chao Zhang, and Jinwen Ma. Cnn-lstm neural network model for quantitative strategy analysis in stock markets. In *International Conference on Neural Information Processing*, pages 198–206. Springer, 2017.

[28] Yahoo finance - stock market live, quotes, business finance news.