1.

combination	window size	step	MSE
1	10	15	322.857758
2	15	10	225.862183
3	20	5	17.956873

Combination 1 with a short window of 10 and a large step of 15 suffers the highest MSE (322.86). Increasing the window to 15 and reducing the step to 10 lowers MSE by about 30% to 225.86 in combination 2. Finally, using the longest window of 20 and the smallest step of 5 yields the greatest MSE of 17.96 (combination 3).

It makes sense because a larger window size provides the model with more historical context, and a smaller step produces more overlapping sequences—i.e., denser coverage of the time series—which helps the model better capture temporal dependencies. However, with a larger time window and a smaller step, it will need more computational resources.

2.

(i) I retained the Combination 3 settings and added Volume as a feature, which caused MSE to jump to 1095.31, and the validation loss and predictions were horizontal lines in plots. This might be because Volume values (in the millions) dominated the LSTM's gradients while price features were only in the tens or hundreds. I then applied a StandardScaler to all features and retrained, and MSE dropped dramatically to 0.002955. Although this is a huge improvement, I needed to compare fairly (since I had not normalized in the first experiment). So, I ran the model again without Volume and achieved an even better MSE of 0.001171—suggesting that Volume may not be a helpful feature.

(ii) I kept the Combination 3 settings and first added the High–Low and Close–Open features. I then computed the Pearson correlation of each feature with the next-day High. I performed a forward-selection procedure, adding features one at a time until the MSE no longer improved with the descending order of correlation features: Close, High, Low, Open, Close–Open, Volume, and High–Low. As soon as I added High, validation MSE increased, so the optimal feature set contained only Close. However, since Close, High, Low, and Open are all highly collinear, I also tested the sets {Close, Close–Open} and {Close, Close–Open, Volume}, but neither outperformed the single-feature model using only Close.

Features	MSE
Close	0.000640
Close, High	0.001198
Close, Close-open	0.004302
Close, Close-open, Volume	0.005104

3.

Since in Question 2, we already know that applying normalization can improve model performance, I want to test whether normalization still makes a difference if I use only close as the sole feature. The experimental results show that the MSE is 31.768358 without normalization, which is significantly different from 0.000640.

In Andrew Ng's machine learning class[1], he mentioned that feature scaling will make gradient descent go faster. Therefore, I ran an experiment to see how many epochs are needed for the validation loss to stably below 0.01 : with normalization it took only 20 epochs, whereas without normalization it required 322 epochs.

4.

My window size is greater than the step size! I don't think this description is correct. I believe that if the window size is smaller than the step size, gaps will appear between windows, and data in between will be skipped, which should be negative to the model's learning.

5.

According to Iglesias *et al.* (2023), there are three families of time-series augmentation. First, simple signal transforms—including time slicing window, jittering, scaling, rotation, permutation, and channel permutation—expose varied segments or distort signals in controlled ways. Second, deep

[1] https://www.coursera.org/learn/machine-learning

generative models—VAEs (e.g., CVAE, GlowImp) and GANs (e.g., WaveGAN, TimeGAN)—learn the data distribution and synthesize entirely new sequences. Third, alignment-based synthesis uses Dynamic Time Warping (DTW) to average or recombine multiple series into realistic hybrids.

Among the above, the magnitude warping, which belongs to scaling, specifically applies to time-series data. In magnitude warping, one samples random scaling factors at several time-points (knots), fits a smooth cubic spline through them, and then multiplies each original value xt by its corresponding spline interpolation value at , yielding $x'_t = \alpha_t x_t$. This smoothly distorts amplitude over time while preserving overall shape.

6.

(i) Convolution-based models

Convolutional models use a fixed receptive field set by kernel size, dilation, and depth. At inference, you slide the same-sized window (or maintain a rolling buffer of that length) across the input with identical stride and padding, ensuring each output sees the exact context as seen during training.

(ii) Recurrent-based models

Recurrent models (e.g., RNNs or LSTMs) naturally process one time-step at a time—carrying forward a hidden state—and can optionally reset or truncate that state every N step to match the truncated-BPTT length used in training.

(iii) Transformer-based models

Transformer models operate over a fixed attention window: you break a long sequence into segments no larger than the trained context length, and—if you need longer-range context—cache past key/value pairs so that each new segment can attend both to its tokens and to the remembered representations of earlier segments.

Reference

Iglesias, G., Talavera, E., González-Prieto, Á., Mozo, A., and Gómez-Canaval, S. (2023), "Data augmentation techniques in time series domain: a survey and taxonomy," *Neural Computing and Applications*, Vol. 35, No. 14, pp. 10123-10145.