1. (15 points) Experiment with different window sizes and steps. Train the model using **3** different combinations of window size and step. Evaluate the Mean Squared Error (MSE) for each configuration. Report the MSEs using a table and analyze the results. (Approximately 100 words.)

| window_size | step | Train MSE loss | Val MSE loss | best Val loss |
|---|---|---|---|---|
| 10 | 15 | 193.3793 | 203.3557 | 203.3557 |
| 25 | 30 | 468.849 | 888.804 | 888.804 |
| 5 | 15 | 89.5488 | 55.9626 | 55.9626 |
| 5 | 10 | 29.9707 | 11.6229 | 11.6229 |
| 10 | 5 | 7.1129 | 33.0624 | 33.0624 |
| 15 | 10 | 93.1427 | 162.029 | 162.029 |

When window size increases, the MSE on both training and validation also increases, which shows larger window size may be capturing more complexity or noise that makes higher Val MSE loss. When window size = 5 and step = 10, the model had the nice loss on Val sets and train sets, indicating that this parameter had the best performance and have no overfitting.

2. (Approximately 200 words.)
   (i) (15 points) Include 'Volume' as an additional input feature in your model. Discuss the impact of incorporating 'Volume' on the model's performance.
   (ii) (15 points) Explore and report on the best combination of input features that yields the best MSE. Briefly describe the reasons of your attempts and analyze the final, optimal input combination.

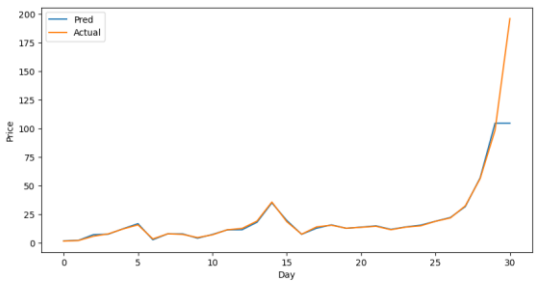
(i)

```
Epoch 100/100, Train loss: 922.3184, Val loss: 1460.8666, Best Val loss: 1438.8607
```

Volume's value is too big that cause high deviation to model and have a very bad performance.

But after normalization of Volume, we can have better loss while window size = 10 and step = 15

```
Epoch 100/100. Train loss: 91.2884. Val loss: 59.4928. Best Val loss: 59.4928
```



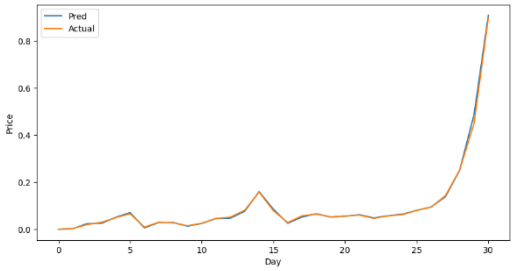

(ii) Choose all features.

```
features = df[['Open', 'High', 'Low', 'Close', 'Volume']]
```

The reason is because choose all features can give more information to model.

And with normalization, we can get very nice performance

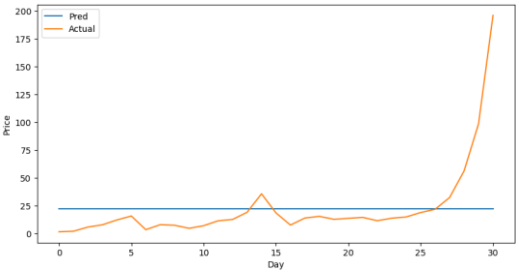
```
Epoch 100/100, Train loss: 0.0001, Val loss: 0.0000, Best Val loss: 0.0000
```

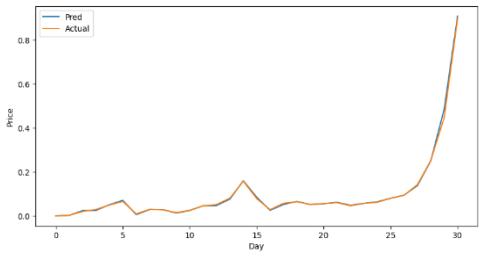



3. (15 points) Analyze the performance of the model with and without normalized inputs in Lab 4. You can use experimental results or external references (which must be cited) to support your conclusions on whether normalization improves the model's performance. (Approximately 100 words.)

Without normalization.

```
Epoch 100/100, Train loss: 650.3993, Val loss: 1331.5419, Best Val loss: 1328.1795
```




With normalization.

```
Epoch 100/100, Train loss: 0.0001, Val loss: 0.0000, Best Val loss: 0.0000
```

Normalization highly improves the model's performance.

4. (10 points) Why should the window size be less than the step size in Lab 4? Do you think this is correct? If you use external sources, please include references to support your response. (Approximately 50 words.)

No, I think is not correct. Window size and step size in LSTM model is independent parameter. And the size of both is determined by what you want. It can be seen in question 1.

5. (15 points) Describe one method for data augmentation specifically applicable to time-series data. Cite references to support your findings. (Approximately 100 words.)

The transforms in the time domain are the most straightforward data augmentation methods for time series data. For example, Window warping or Noise injection.

Window warping selects a random time range, then compresses (down sample) or extends (up sample) it, while keeps other time range unchanged.

Noise injection is a method by injecting small amount of noise/outlier into time series without changing the corresponding labels

References: Time Series Data Augmentation for Deep Learning: A Survey Qingsong Wen , Liang Sun

6. Discuss how to handle window size during inference in different model architectures (approximately 150 words):
   (i) (5 points) Convolution-based models
   (ii) (5 points) Recurrent-based models
   (iii) (5 points) Transformer-based models

(i) Convolution-based models typically handle inference with a fixed window size determined during training, often dictated by the architecture's layer configurations and the dimensionality of input data.

(ii) Recurrent-based models excel in handling varying window sizes naturally due to their sequential data processing capability.

(iii) Transformer-based models are inherently adaptable to different window sizes because they process inputs based on self-attention mechanisms without relying on the sequential processing inherent to recurrent models.