Time Series Forecasting of Sugar Production Yield

2023-1nd Semester
ProjectLab Final Report

Member1	20181128	Byeongyeon So
Member2	20191349	Jiwon Kim
Member3	20201181	Jihwan Oh

Time Series Forecasting of Sugar Production Yield

Byeongyeon So
School of Industrial Engineering
UNIST
booker1024@unist.ac.kr

Jiwon Kim
School of Industrial Engineering
UNIST
kjiwon0219@unist.ac.kr

Jihwan Oh
School of Industrial Engineering
UNIST
dhwlghks00@unist.ac.kr

Abstract

There is an effort to control the quality/production of the process by using a predictive model in the manufacturing process. Even in the case of multi-stage manufacturing processes that move to different processes at each stage and proceed sequentially, the quality and production of the next stage can be predicted using intermediate labels. However, in the case of our study, the continuous multi-stage liquid manufacturing process, research is insufficient because there is no intermediate label. In this report, we propose models for learning the sequential properties of multi-stage processes without intermediate labels called Multi-Stage AutoEncoder and Multi-Stage Linear. We use sugar yield data from Samyang Corporation. Based on EDA, we made an arranged merged data and applied multiple existing time-series forecasting models. Since these models do not reflect our data characteristics of multi-stage process, we propose our own models: Multi-Stage AutoEncoder and Multi-Stage Linear. Experimental results on the Samyang sugar yield dataset show that our models are superior to other applied methods. And we state the limitations of our model and propose other ways to further improve the model for use in real-world processes.

1. INTRODUCTION

Fig 1. Multi-stage process and monitoring in manufacturing process

Yield prediction is a critical challenge in manufacturing processes. Effective control of manufacturing process variables and optimization of yield through raw material variables is essential for successful control, which requires accurate yield prediction. In recent years, there has been an increase in the use of machine learning and AI to predict yield using control data and raw material variables.

The production of most products now involves multiple steps, and many factories are adopting continuous processes. These continuous processes have the advantage of increasing yield compared to discrete manufacturing. As shown in Figure 1, real-world companies are implementing continuous processes and monitoring them.

Most traditional yield prediction AI models consider all variables simultaneously, but this approach fails to account for multi-step processes. For example, if there are three stages of a process to get the final target as shown in Figure 2, the variables of stage 1 can affect stage 2, and stage 2 can affect stage 3, but not vice versa. A model that considers all variables simultaneously is unlikely to reflect a multi-stage process because the variables of all stages become intertwined during the learning process.

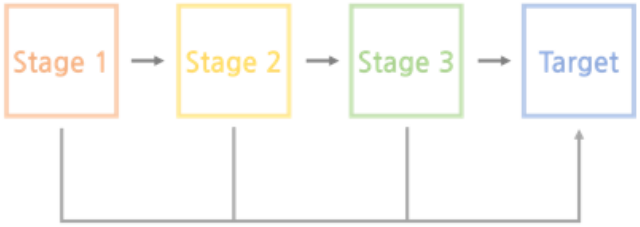

Fig 2. Example of Multi-stage process

An additional difficulty with our task is that we don't have intermediate outputs for each stage. If we had intermediate outputs, we could use each stage variables to predict the intermediate output, and then in the next Stage, we could think of the previous intermediate output as the representation variables of the previous Stage and use them as inputs of next stage, along with the next stage variables, to predict the intermediate output of that Stage, and so on. However, since there are no intermediate outputs, we need to make sure that when we produce a representation variable for that Stage, it is not affected by the next Stage.

In this project, we focused on yield prediction that takes into account multi-step processes. We applied our data to AI models, analysed the yield prediction results, and developed our own multi-stage model.

2. LITERATURE REVIEW

In fact, time-series models that take multi-stage into account are still rare. In particular, we were interested in applying a

multi-stage time-series forecasting model to a manufacturing process with no intermediate labels, but there is limited research in this area. Therefore, we prioritized resources on multivariate time series forecasting. The studies below are models that do not take multi-stage into account.

2.1 Causal Discovery with Attention-Based Convolutional Neural Networks *(MDPI 2019)*

This paper proposes AD-DSTCNs, which is a CNN-based model for temporal causal discovery discovery. AD-DSTCNs learns a causal graph structure in time-series data. This model use attention-based convolutional neural network. It use input as each variable multiplied with attention score and use dilated TCN(Temporal Convolutional Network) with depth-wise convolutions for each input. Then it is followed by a 1*1 point-wise convolution that merges together the resulting output channels for each variable. It can explain the time lag occurred between cause and occurrence of the effect.

2.2 Are transformers effective in time-series forecasting? (*AAAI 2023*)

This is the linear based deep learning model in the field of time series prediction made on 2023. It examines whether the existing transformer-based model used for time series prediction is suitable for time series models. They argue that the permutation-invariant self-attention mechanism of the transformer-based model inevitably loses information. As evidence, they introduce a linear model of a very simple neural network structure called LTSF-Linear. It performs better than any sophisticated transformer-based model ever built. Therefore, we would like to implement the model according to our data by referring to this paper.

2.3 Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting *(AAAI, 2021)*

In this paper, Informer is proposed as a specialized forecasting model in long sequence time-series data. Informer use transformer, but in more improved way in computational and spatial complexity. It overcomes the limitations of existing Transformers in LSTF data in several ways. First, using ProbSparse self-attention mechanism, it reflects the fact that a particular query responds to a particular key, which improve computational complexity. So, the self-attention is performed using only a specific query with sparsity. Second, using self-attention distilling operation, the model extract adjoint timestamp information through 1d convolution and further reduce the feature map with max pooling. And the model finally forms a more focused self-attention feature map with superior features. This can improve spatial complexity since the traditional transformer keeps the sequence length after passing the attention layer. Lastly, through generative style decoder, sequence forecasting is performed in one-shot using time information and time-series values as one input vector.

3. DATA

3.1 Data Description

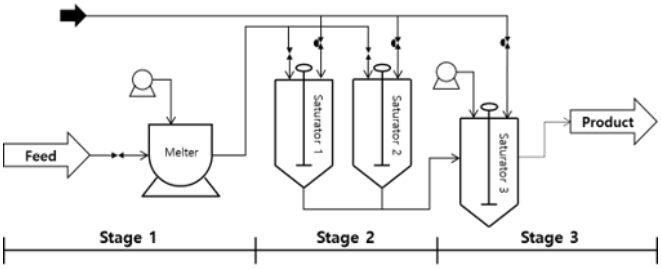

Fig 3. Continuous multi-stage liquid manufacturing process

The Figure.2 shows the continuous multi-stage liquid manufacturing process of making sugar in factory. We have multivariate time series dataset on the sugar manufacturing of Samyang Corporation, spanning seven years. This dataset includes a surrogate measure of production yield and production control variables.

There are total of three types of datasets: feed data, process operation data, and equipment data. Feed data is experimental value indicating the components of raw sugar injected into the process. This data is measured once a day. It includes 1278 days and 7 variables. Process operation data is sensor data including the operation state of process and control values. This is measured every 15 minutes. It includes 122640 time points and 17 variables. Equipment data is state of equipment related to yield of sugar production. This is measured once a day. It includes 1278 days and 3 variables.

	Variable Name
Stage1	'Pol', 'IS', 'Moisture', 'Ash', 'Filterability', 'Color', 'Alcohol_Floc', 'MELTER.MELTER_CONTROL_BX.Value', 'MELTER_ML.TEMP_PV_Value'
Stage2	'CO2.CO2_GAS_F.Value', 'CO2.CO2_GAS_PRESS.Value', 'CO2.CO2_GAS_concentration_Value', 'SATURATOR1_PH_PV.Value', 'SATURATOR1_ST_TEMP.Value', 'SATURATOR1_CALCIUM_F.Value', 'SATURATOR1_RATE_SV.Value', 'Saturator1_exp_CaO', 'SATURATOR2_PH_PV.Value', 'SATURATOR2_ST_TEMP.Value', 'SATURATOR2_CALCIUM_F.Value', 'Saturator2_exp_CaO'
Stage3	'SATURATOR3_PH_PV.Value', 'SATURATOR3_ST_TEMP.Value', 'Saturator3_exp_CaO'
Target	'SATURATOR_ML_SUPPLY_F_PV.Value'

Fig 4. Variable name per Stage in data of Samyang Corporation

The time intervals were not the same, with some datasets being measured once a day and others every 15 minutes, and

for the feed and equipment data, which were measured once a day, the time of measurement was different: 7:00 and 9:00, respectively. We merged the data with a common time interval and common time point (15 minutes). We will talk about this again in Data Preprocessing part. So, within each dataset, variables were also mixed by stage. Later, to facilitate modelling, we organized the merged dataset into stage1, stage2, stage3, and target.

3.2 Exploratory Data Analysis

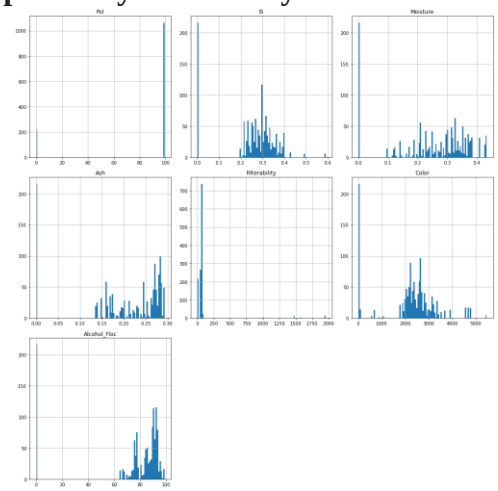

Fig 5. Distributions of feed data variables

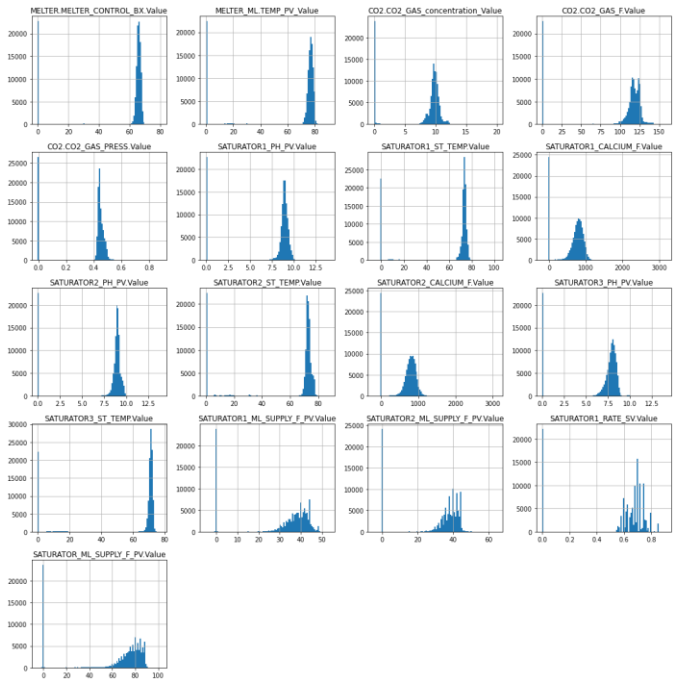

Fig 6. Distributions of process operation data variables

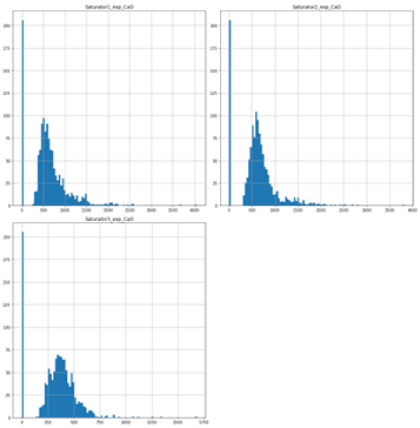

Fig 7. Distributions of equipment data variables

We checked the distribution of data in each dataset. Figure.3 to Figure.5 show that the histogram of each variable. As they are shown, there are lots of missing data which is recorded as 0 and extreme outliers. Most of them do not look like they are following a normal distribution. When we checked the scale of each variable, there were two variables with overwhelmingly large values, which seems to need regularization for variables: SATURATOR1_CALCIUM_F.Value and SATURATOR2_CALCIUM_F.Value.

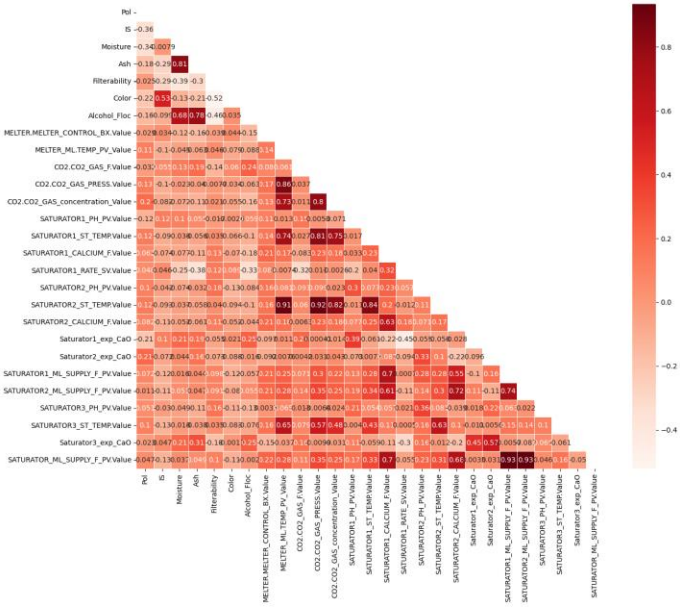

Fig 8. Heatmap correlation matrix

And we checked correlation between variables. This figure.4 shows that correlation heatmap. As they are shown, there are highly correlated features with target with the value over 0.9. This was due to the sum of these two variables is the value of target. We had to remove them.

3.3 Data Preprocessing

3.3.1 Data Merge & Extension

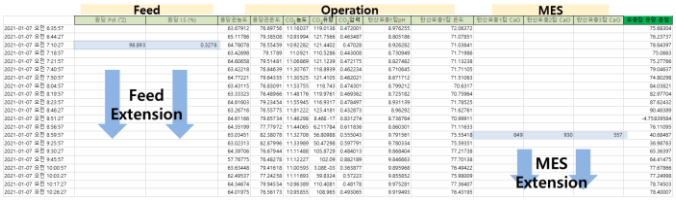

Fig 9. Data merge and extension method

Each data has a different time-frequency. Feed and equipment data are measured once a day, and operation data is measured every 15 minutes. To merge the data into one dataset, we extended feed and equipment data by copying and pasting with the same data for every 15 minutes.

3.3.2 Time lag

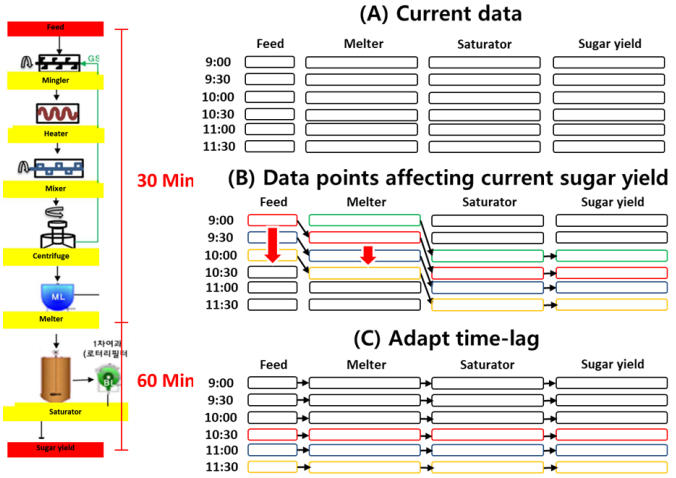

Fig 10. Time lag shift method

There exists time lag in which the process data collection time and the actual influenced time were different. It takes about 30 minutes and 60 minutes more for the currently measured value to be measured by moving the stage. To make the data the same column at the same time point, we adjusted the time lag with 'shift' function.

3.3.3 Remove rows with Null (NA) or Zero (0)

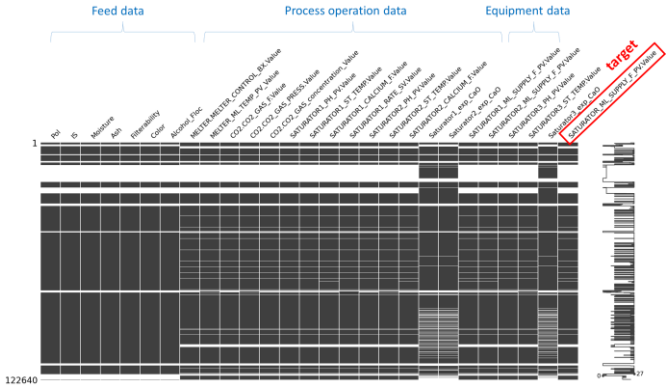

Fig 11. Visualization of missing values

This figure shows that the missing value distribution in each variable. Row means time points. Column means variables. The white parts are missing data. For the missing values, majority of them were missing value for all rows. We determined that was the sensor error that we removed rows filled with all missing values which is recorded as 0 and null value. But for the ones that were only missing for certain columns, we did not drop and left them.

3.3.4 Remove Outliers

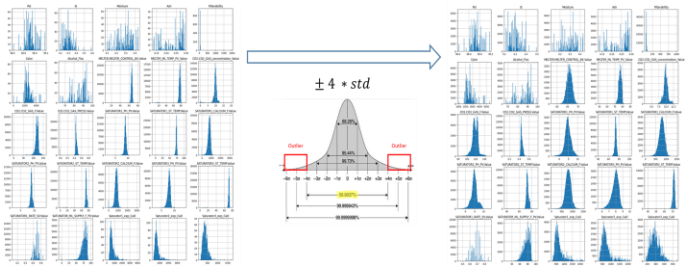

Fig 12. Before/After data distribution. (Left: Before preprocessing)

When we checked distribution of our data, we found extreme outliers. To drop extreme outliers, we dropped data out of four standard deviations from the mean of each variable. Since we thought all the negative values would be outliers that we set four standard deviations. This figure shows the distribution (histogram) of data before and after the remove of the outlier. Many variables were in the form of normal distribution.

4. METHOD

As mentioned earlier, there is not much research on multi-stage time-series forecasting without intermediate outputs. We first applied existing time-series prediction and forecasting models without considering multi-stage process: linear regression, random forest, LSTM, AD-DSTCNs, LSTF-Linear, and Informer. And we introduce our own models which reflect multi-stage process. Before describing our model, we want to explain why RNN and MLP models fail to

reflect multi-stage features. When RNN or MLP is trained, the error term is updated through backpropagation. In this backpropagation process, subsequent stages unavoidably influence the preceding stages. Mathematically speaking, x_{n-1} will be affected by x_n. Thus, the existing models that we applied cannot reflect our data characteristics of multi-stage process.

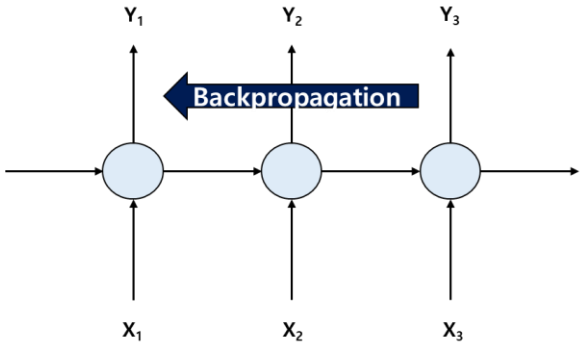

Fig 13. Backpropagation in RNN

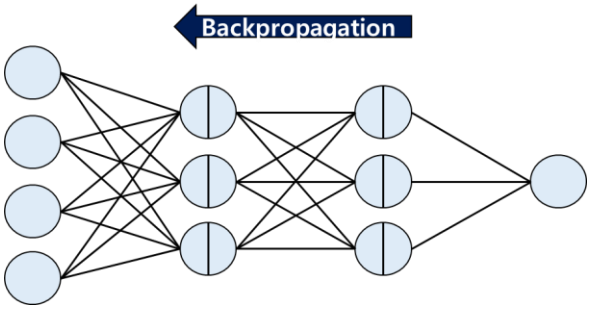

Fig 14. Backpropagation in MLP

We need a model in which the previous stage can affect the next stages, but not the previous stages. Therefore, we develop our own model, multi-stage linear model that restricts the next stage from affecting the previous stage during the backpropagation.

The Method part describes the process by which our model was developed. Section 4.1 introduces two models made with linear modules. However, these models violate the multi-stage property. We have developed two models that follows the multi-stage process. Section 4.2 introduces the model using autoencoder. Section 4.3 introduces the multi-stage linear model, which complement previous proposed methods.

4.1 Linear

To reflect multi-stage process characteristic, we developed two linear modelling methods. For the first linear method, we reduce the dimension of each stage variable. Reduced output variables are concatenated with the variables in next stage and again create new predicted output. The number in parentheses is the number of columns. More specifically, we make 9 variables in the first stage into 3 variables. The whole prediction process is working through the linear module. Then, we have 3 variables, and we will call this output1. Output1 is concatenated with 12 variables in the second stage. Using linear model, we make them into 5 variables, which is output2. Again, output2 is concatenated with the 3 variables in the third stage. Then, we have output3 with reduced 4 variables. Finally, we concatenate the output3 and target at t-1 time point. With these final 5 variables, we make it into one variable. This means that we first make the output in the same dimension with the target dimension. In this manner, we train our output to be a good predictor of target within the same dimension.

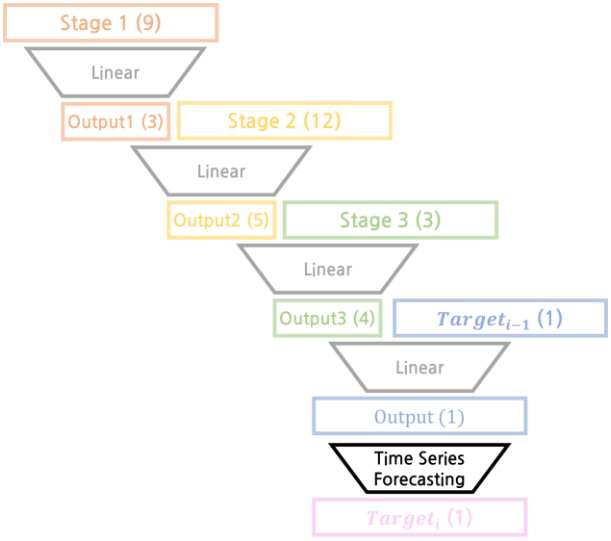

Fig 15. Multi-Stage model with linear layers

For the second method, the modelling method is exactly same. The only difference is last part where we made 5 variables. In method1, we made the with the 5 variables and reduce the window size (8) of variables into forecast size (1). But, for method2, we trained each 5 variables to fit the target. We concatenate these trained 5 variables and make it into forecast size. Again make the last forecast-sized output to fit the target.

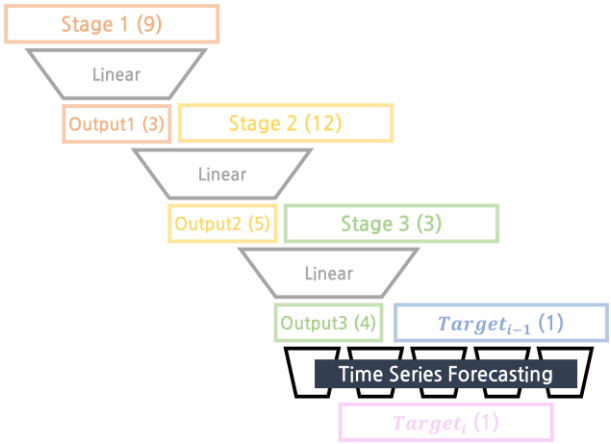

Fig 16. Multi-Stage model with linear layers adding 5 variables

However, though we extracted variables in each stage and concatenated for forecasting, we found the same problem occur in the process in backpropagation. Here is our formula for linear model. Our model simply multiplies the weight and x to make z. Then, the variables in the z and the next stage are concatenated and multiplied by W. However, since the concatenated variable is eventually recognized as one node, the previous stage variable will be affected by the subsequent stage variable during backpropagation.

$$z_1 = W_1^T x_1,$$

$$z_n = w_n^{injected} \times concat([z_{n-1}, x_n], axis = 1)$$

Fig 17. Formula of Linear

Suppose concatenated vector as A, then it goes in as one input and will be affected by z_n during the backpropagation process. Eventually, x_{n+1} affect x_n.

Fig 18. Simplification of Backpropagation in Linear

4.2 Multi-Stage AutoEncoder

AutoEncoder is a model for making a good representation of input itself that when the representation is entered to decoder, the input itself can be made. In this way, we used this autoencoder to make good representation for each stage.

When we create a representation variable with AutoEncoder in the first Stage, the next Stage will use the previous representation variable and the next Stage's variables to produce that Stage's representation variable with AutoEncoder. After repeating this process, the final representation variable with previous target variable is used as input in time series forecasting model and predict the target.

The significance of this model is that the representation variable of each stage variable does not change: the first representation variable is a good representation of the first stage variables, and starting from the second stage, the representation variable of the next stage is created with the contents of the previous representation variable and its stage variables. We can say that this represents Multi-Stage Forecasting.

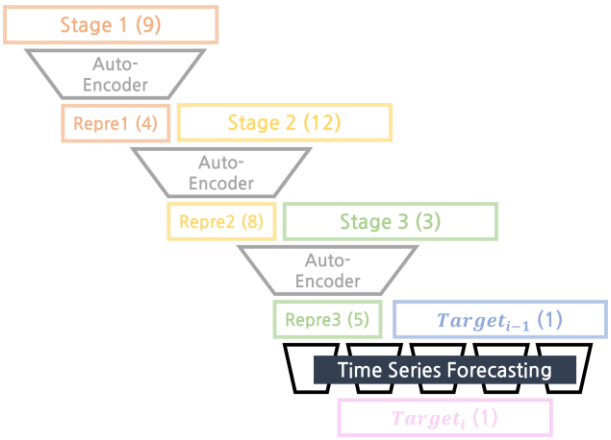

Fig 19. Multi-Stage model with AutoEncoder.

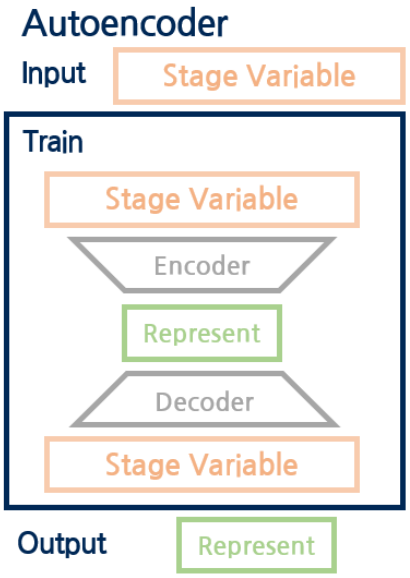

Fig 20. Representation module structure in Multi-Stage Linear

4.3 Multi-Stage Linear

This multi-stage linear model is developed in a way that improves shortcomings with the above models. We made small modifications in second linear model in Section 4.1. With this Multi-Stage Linear, the subsequent stages cannot affect preceding stages in the backpropagation perspective.

We apply a linear module so that the previous representation and the next stage variables have the same dimension. Then, we create the following representation by summing them together. Here, we sum the two variables in the same dimension, not concatenate two of them that subsequent stages do not affect former stages.

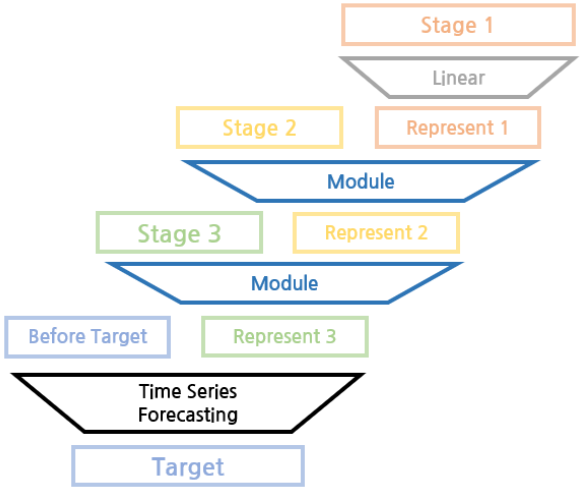

Fig 21. Multi-Stage model with Multi-Stage Linear

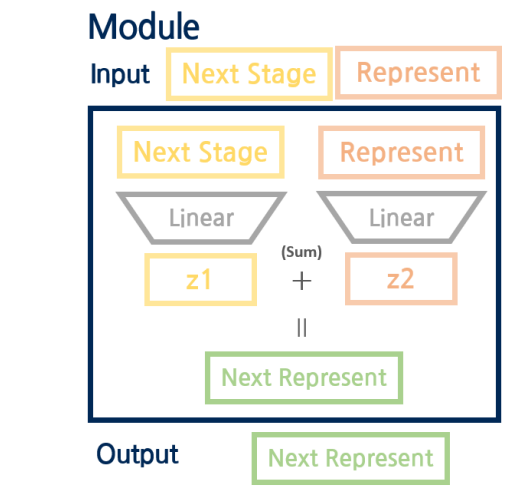

Fig 22. Representation module structure in Multi-Stage Linear

This is the formula of Multi-Stage Linear.

$$z_1 = \sigma(W_1^T x_1),$$
$$z_n = \sigma\left(W_n^{hidden} z_{n-1} + W_n^{injected} x_n\right),$$
$$y_n = MLP(z_n)$$

Fig 23. Formula of Multi-Stage Linear

Here is how the backpropagation work in the multi-stage linear model. Since z_{n-1} and x_n are added in the same dimension, the latent variables of the previous stage, z_{n-1} and variables of the next stage, x_n are recognized as two nodes. Accordingly, x_{n+1} does not affect x_n, resulting in reflecting multi-stage process.

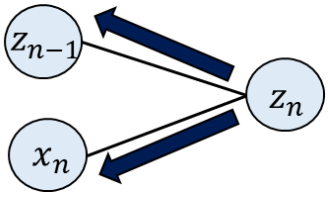

Fig 23. Simplification of Backpropagation in Multi-Stage Linear

In fact, when we take derivatives with respect to W_n^{hidden} and $W_n^{injected}$, we get the value regardless of x_{n+1} which means that only previous stages affect next stages.

$$\frac{\partial \mathcal{L}}{\partial W_n^{hidden}} = \frac{\partial z_n}{\partial W_n^{hidden}} \cdot \frac{\partial \mathcal{L}}{\partial z_n} = z_{n-1} \cdot \frac{\partial \mathcal{L}}{\partial z_n}$$

$$\frac{\partial \mathcal{L}}{\partial W_n^{injected}} = \frac{\partial z_n}{\partial W_n^{injected}} \cdot \frac{\partial \mathcal{L}}{\partial z_n} = x_n \cdot \frac{\partial \mathcal{L}}{\partial z_n}$$

Fig 24. Derivatives of loss with respect to W_n^{hidden} and $W_n^{injected}$

5. Experiments & Result

We split our data into train, validation, test set as 0.8, 0.1, and 0.1. Time-series data at a particular point is heavily influenced by ambient point. It is important to do chronological data sampling without shuffling. Also, the loss is set to MSE.

$$MSE = \frac{1}{n}\sum(y - \hat{y})^2$$

Fig 25. Mean squared error formula.

For time-series prediction input data, linear regression and random forest used a single row without target as input and target as output. For time-series forecasting input data, we used a sliding window. The sliding window technique is a method that define a fixed-size window that moves through the data, allowing for localized computations or analysis to be performed on the data within that window. This technique enables efficient and effective predictions in sequential data, capturing temporal patterns or trends within a specified window of observations. We used input data for independent 24 variables at t period and target at t-1 period and forecast target at t period for training our own models. We set window size as 8, forecast size as 1, and batch size as 64. This is the result of existing models: Linear Regression, Random Forest, LSTM, AD-DSTCNs, LSTF-Linear, and Informer.

	LR	RF	LSTM	**AD-DSTCNs**	**LSTF-Linear**	Informer
MSE	23.689	35.949	66.553	**12.911**	**14.665**	39.433

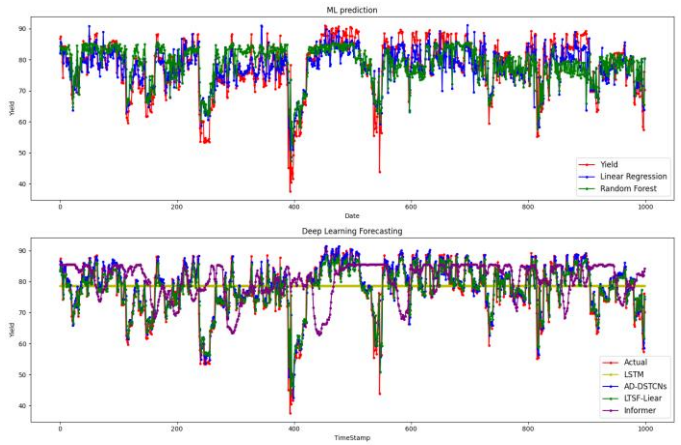

Fig 25. Experimental result of non-multi-stage models.

5.1 Linear

When we checked the loss of each linear model, it was 17.39 and 12.35 for each. Since the second linear model was more complex, it showed better predictions than the first one. So, we chose Linear 2 model for another experiments.

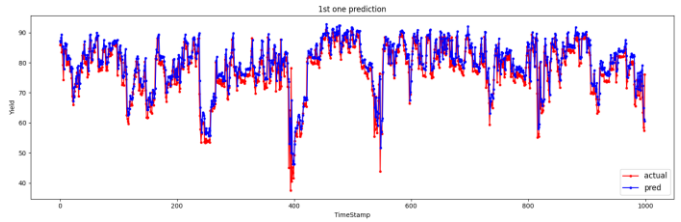

Fig 26. Experimental result of Multi-Stage-Linear 1 model.

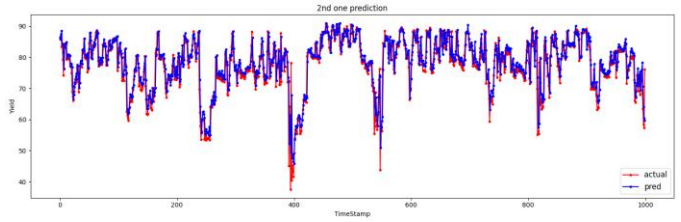

Fig 27. Experimental result of Multi-Stage-Linear 2 model.

To check whether our model reflect our multi-stage process, we experimented with shuffling the order of processes. We made a hypothesis that if our model reflects the multi-stage sequence well, then our model will perform when the data is put in its original order. We ran the model three times with different random seed for each different order and calculated the average MSE loss. This is the result of the experiment, and the loss was similar which means our model is defective.

Model	MSE
1 → 2 → 3	**13.5092**
1 → 3 → 2	14.3461
2 → 1 → 3	**13.0092**
2 → 3 → 1	16.3090
3 → 1 → 2	**13.3864**
3 → 2 → 1	**13.1483**

Fig 28. Experimental result of Multi-Stage-Linear 2 model with shuffling the order of processes.

5.2 Multi-Stage AutoEncoder

We tried different method using AutoEncoder in the similar way but to follow the multi-stage process. The MSE was 13.42. The loss is average MSE loss of different three random seed. Our model performed close to the latest existing model.

Model	MSE
AutoEncoder	13.4170

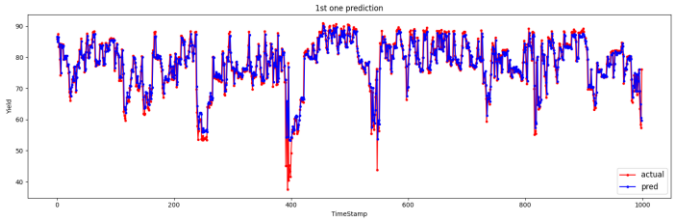

Fig 29. Experimental result of Multi-Stage AutoEncoder model.

5.3 Multi-Stage Linear

We finally develop a model that complements the deficiencies in former models. Multi-Stage Linear follows multi-stage process, also can be trained in end-to-end. This proposed model showed the best performance, reflecting the original manufacturing order with the loss of 12.99.

Model	MSE
1 → 2 → 3	**12.9894**
1 → 3 → 2	14.0530
2 → 1 → 3	13.7356
2 → 3 → 1	15.3857
3 → 1 → 2	13.1917
3 → 2 → 1	14.9665

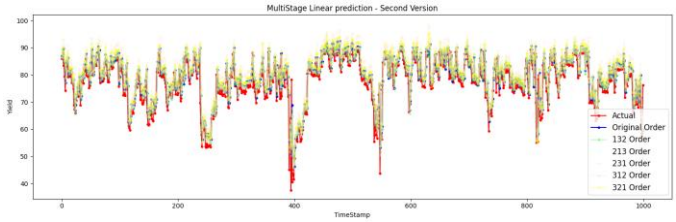

Fig 30. Experimental result of Multi-Stage AutoEncoder model with shuffling the order of processes.

6. Conclusion

We applied traditional machine learning and deep learning methods. Especially, through applying recent deep learning methods, we improved understanding of new models. Also, our data was a new data format that we have never encountered, but which is critical to consider these various stages of a manufacturing process. We proposed a new model framework that considers multi-stage beyond simple time-series forecasting.

Although our proposed multi-stage model is structurally simple, it performed well enough to be comparable to the performance of state-of-the-art AI models. This suggests that further refinement and improvement of models reflecting our multi-stage process could lead to better results on this task.

	AD-DSTCNs	LSTF-Linear	Multi-Stage AutoEncoder	**Multi-Stage Linear**
MSE	**12.911**	14.665	13.417	**12.989**

Fig 31. Comparison of our model(multi-stage) to the models that performed well in existing models(not multi-stage)

7. Future work

For our model to be used in the field, it needs to be further improved. First, it needs to automatically preprocess the time lag. As previously introduced in Data part, our process has a time-lag problem in which the process data collection time and the actual effect time of each stage variable are different. We preprocessed time lag based on the domain knowledge, but still, it is not accurate and there is more often no prior knowledge in the field. So, we need a model that can automatically adjust time lags. Second, we need a model that takes account of more temporal information. Our model considers the temporal information, but it needs to do it in more sophisticated way. We can further apply Transformer, RNN, or TCN to our Multi-Stage Linear. Finally, for the case of Multi-Stage AutoEncoder, each of latent variable are not made for better forecasting, just for good represent of each stage. So, we can consider the way how to make good latent variables that contains of both good forecasting the yield and representation of each stage variables.

For other model to apply, we can use self-supervised learning model to construct each stage variables. To put it more concretely, 'Deep Infomax' can be used. We referred paper "Learning Deep Representations by Mutual Information Estimation and Maximization". Deep Infomax is the first model to introduce MI (Mutual Information) in contrastive learning. Mutual Information is an index that determines how independent the two distributions are from each other. The detail of methodology is shown below. This algorithm allows the inner product of the positive feature vector and the context vector become smaller (closely mapped) and negative feature vector and the context vector bigger (mapped far). However, we can also use this summarize vector with supervised task such as regression or classification (self-supervised). Then, the summarize vector s will be more representative (upgraded). So, we can get summarize vectors that represent the original data. Then, we can use this vector to multi-stage time series forecasting as shown in figure. We expect this model to perform well on our data as a result. We hope to implement this algorithm if we have a chance in the future.

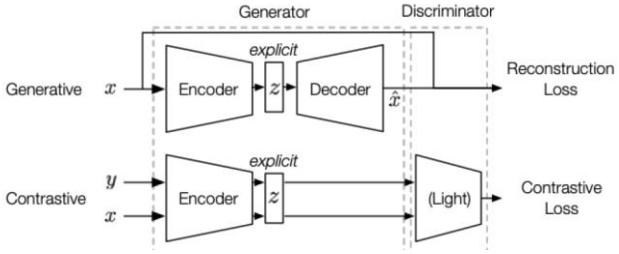

Fig 28. Generative and contrastive model structure.

References

[1] Nauta, M., Bucur, D., Seifert, C. Causal Discovery with Attention-Based Convolutional Neural Networks. MDPI, 2019

[2] Ailing Zeng, Muxi Chen, Lei Zhang, Qiang Xu. Are transformers effective in time series forecasting? *AAAI*, 2023

[3] Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., Zhang, W. (2021). Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. AAAI, 2021

[4] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, Yoshua Bengio. Learning Deep Representations by Mutual Information Estimation and Maximization. *ICLR*, 2019