European Research Studies Journal
Volume XXVII, Special Issue 2, 2024

pp. 38-48

Developing an IoT and Machine Learning-Based Monitoring System for Discrete Production Processes

Submitted 18/02/24, 1st revision 16/03/24, 2nd revision 20/04/24, accepted 16/05/24

Krzysztof Król[1], Michał Oleszek[2], Grzegorz Bartnik[3], Olena Ivashko[4], Marek Rutkowski[5], Adam Hernas[6]

Abstract:

Purpose: *This paper aims to develop a tool to support discrete manufacturing process monitoring using IoT sensors and machine learning systems.*
Design/Methodology/Approach: *Machine learning was used to prepare and analyze data from the production line. In discrete manufacturing, measurements from sensors throughout the line at various locations are read for objects moving on the line. The measurements and related research allow for ongoing data analysis and earlier reactions to multiple critical situations.*
Findings: *The study's result was the measurement data analysis in a discrete manufacturing process. Data was obtained from continuous monitoring of technological processes. It also shows how to classify components on the production line, allowing for better decision-making under uncertainty.*
Practical Implications: *The presented method of preparation and analysis of measurement data will allow for better production management and observation of the quality of this production.*
Originality/Value: *A novelty is using an approach to data preparation and processing, neural network systems preparation, and element classification on the production line.*

Keywords: *Neural network, positive predictive, negative predictive, RMSE.*

JEL codes: *C45, C61, E20, L23, O14.*

Paper type: *Research article.*

[1]*Corresponding Author: Netrix S.A./WSEI University , Lublin, Poland, e-mail: krzysztof. krol@netrix.com.pl;*
[2]*Netrix S.A./WSEI University, Lublin, Poland, e-mail: michal.oleszek@wsei.lublin.pl;*
[3]*WSEI University, Lublin, Poland, e-mail: Grzegorz.Bartnik@wsei.lublin.pl;*
[4]*WSEI University, Lublin, Poland, e-mail: Olena.Ivashko@wsei.lublin.pl;*
[5]*WSEI University, Lublin, Poland, e-mail: Marek.Rutkowski@wsei.lublin.pl;*
[6]*Wyższa Szkoła Biznesu - National Louis University, e-mail: ahernas@wsb-nlu.edu.pl;*

1. Introduction

In today's industrial era, effective monitoring of production processes is critical to ensuring high product quality, optimizing productivity, and minimizing production costs. Process monitoring plays a vital role in discrete manufacturing, where products are manufactured as separate units due to the complexity and dynamics of operations (Król, 2021). One of the commonly used production systems is production based on production and assembly lines, which allows products to move continuously through individual stages of production (Król, 2023).

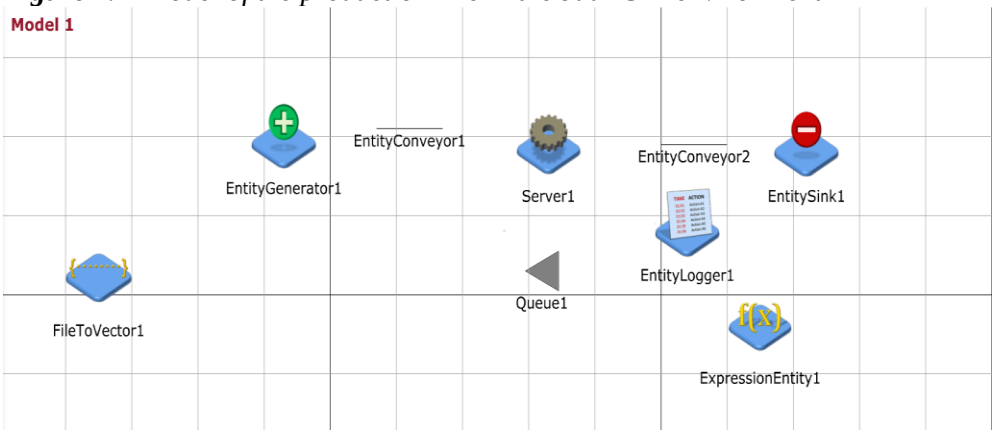
This article will analyze and develop a strategy for monitoring the discrete manufacturing process, particularly on belt lines. Our goal is to provide a comprehensive methodology that allows you to effectively track various aspects of the production process, identify potential problems, and ensure a quick response during irregularities.

2. Prepare Measurement

The first model is the production line model created in the JaamSim environment. JaamSim (JaamSim, 2016) is a free discrete event simulation software that includes, but is not limited to, a drag-and-drop user interface, interactive 3D graphics, input and output processing, and modeling tools and editors.

The designed model reads data from a file and a number and then squares it. The result and other important information, such as the time the line has been running, are stored in the output file. The model includes FileToVector1, EntityGenerator1, EntityConveyor1, Server1 and Queue1, EntityConveyor2, ExpressionEntity1, EntityLogger1, and EntitySink1. The discussed model of the production line is visually presented in Figure 1.

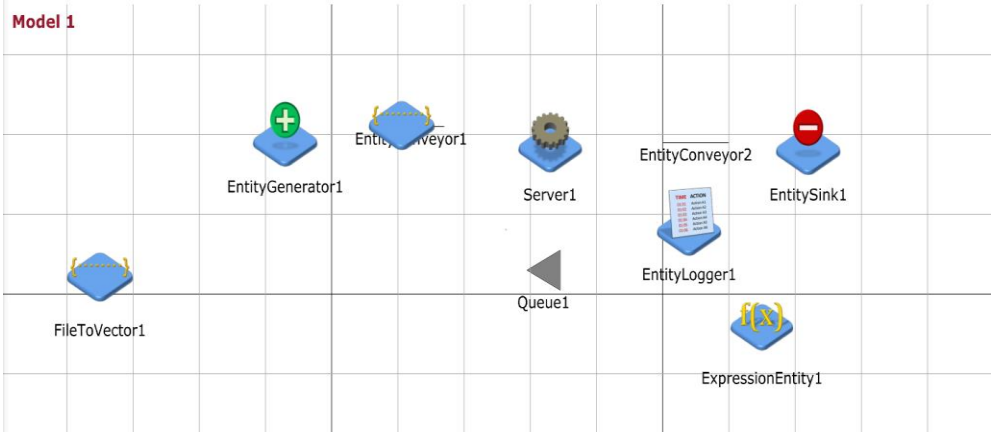
Figure 1. *A model of the production line in the JaamSim environment*



Source: *Own creation.*

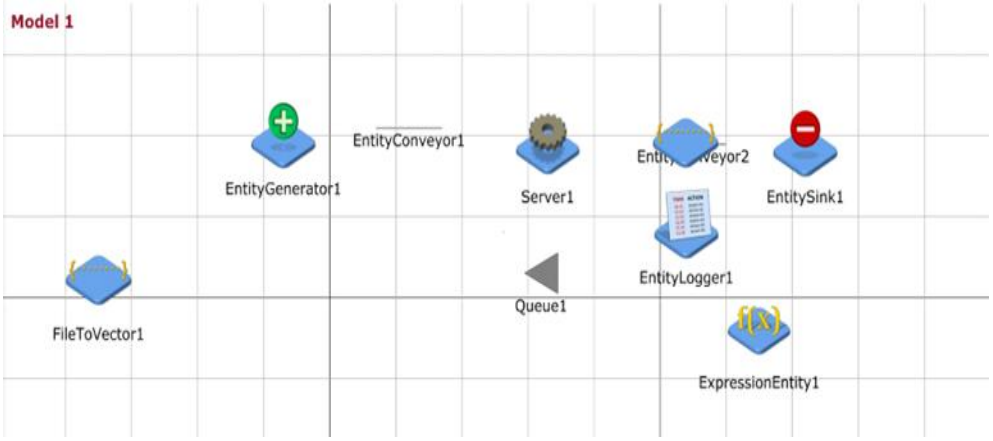
Once the simulation starts, the production line starts working (Figures 2 and 3). We can observe how objects move around EntityConveyor1 and EntityConveyor2.

Figure 2. *Starting the simulation in the JaamSim environment*



Source: *Own creation.*

Figure 3. *Simulation in JaamSim*



Source: *Own creation.*

The individual elements of the model are appropriately connected. Creating a production line model in the SimPy environment (SimPy). One potential tool for modeling production lines is the SimPy Python library. Among other things, it enables the simulation of discrete production processes.

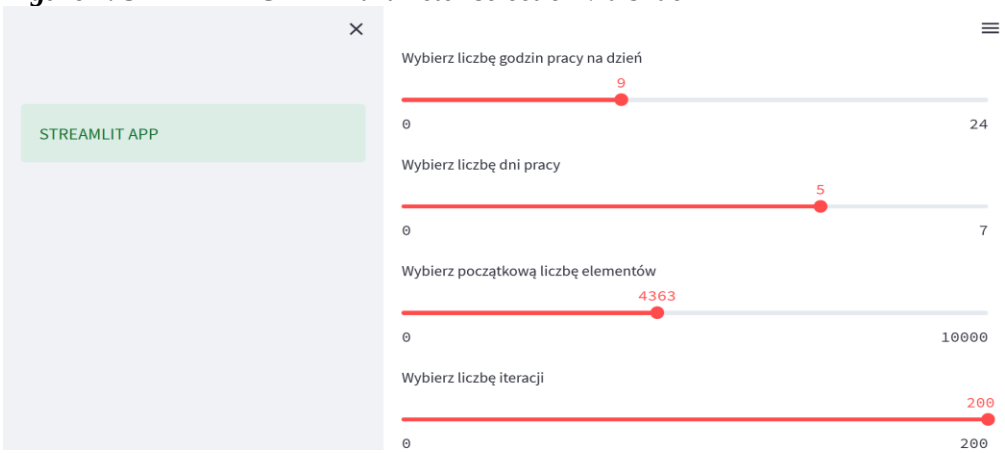
Certain parameters were set at the beginning, such as the possible running time of the production line ('total_time' was set at 40 working hours) or the initial number of elements (1000).

The designed line model is based on the Factory class. Items from the initial container go to Workstation 1 and then to Container 1, from where they are retrieved further. Individual steps are carried out using different time parameters.

3. Implementation of the Solution – STREAMLIT (Streamlight)

The above solution can be implemented as a web application using the Streamlit library. Streamlit is a free, open-source platform for quickly building and sharing web applications. It is a Python-based library specifically designed for machine learning engineers. Streamlit turns data scripts into web applications that can be shared in minutes. You can choose from the sliders to select parameters such as the number of working hours per day, the number of working days, the initial number of elements, and the number of iterations of the production line simulation.

Figure 4. *STREAMLIGHT - Parameter selection via slider*



Source: *Own creation.*

In Figure 5, we can observe the parameters that have been selected.

Figure 5. *STREAMLIGHT - selected parameters*

Source: *Own creation.*

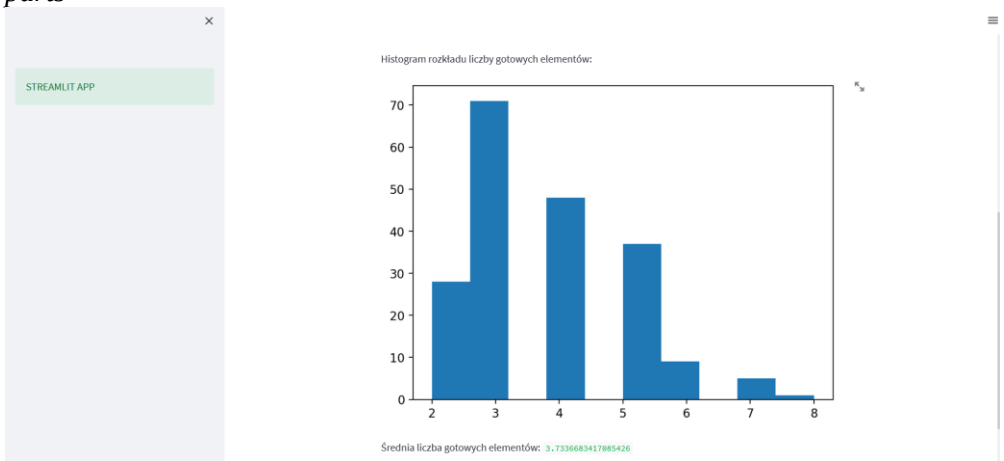
STREAMLIT also allows you to implement graphs in a web application (histogram of the number of finished elements in Figure 6). This window can be further enlarged.

4. Another Program that you Can Use is Dyslexia

The data generated from the Flexsim program was downloaded to monitor the process. The variable motor load was used for monitoring. Since the process itself did not assume the existence of failures in the form of significant deviations from the ordinary course of the process, such disturbances were introduced randomly. A function was prepared to add random disturbances with a given frequency, amplitude, and direction.

As a result of the function's operation, a process is characterized by certain deviations from regular operation. For the tests, the interference frequency was 0.001; the magnitude was 50, 75, 100, 125, and 150, and the variable direction of the inclusions (i.e., the deviation in the process could be positive or negative - the direction was chosen randomly).

Figure 6. *STREAMLIT - histogram of the distribution of the number of finished parts*



Source: *Own creation.*

The idea of detecting introduced disturbances in the time series of the engine load was to calculate the moving averages of the process so that the moving average process predicted the trend. Different moving averages (25, 50, 75, and 100 epochs backward) were selected to see which setup worked best.

Then, a confidence interval was added for the moving average with different multipliers to filter out outliers. A classic three was adopted, which gives the probability that the mean is within the interval with 99.77% certainty.

Still, since it was estimated based on observations from the sample, the confidence interval was extended to 3.5 and 4 times.

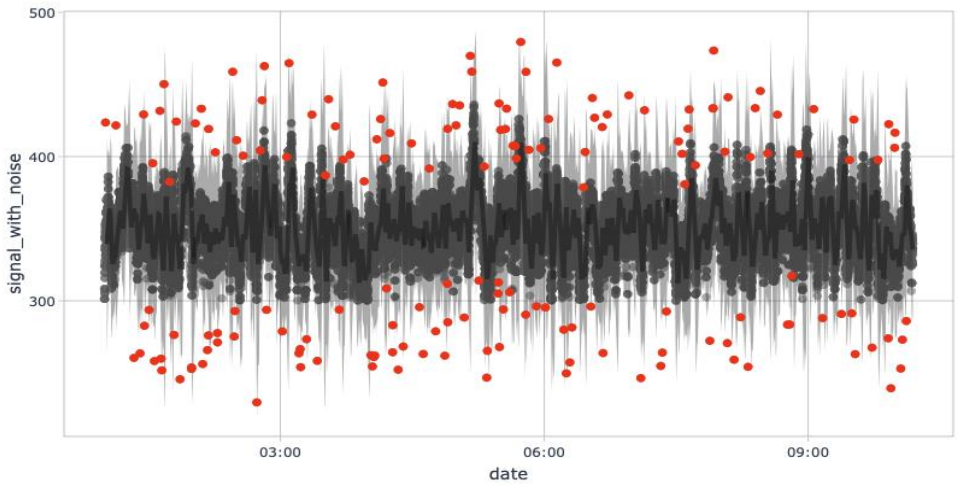
The following measures were used to assess the quality of the model:

- accuracy,
- kappa,
- positive predictive value,
- negative predictive value,
- balanced accuracy.

To illustrate how the classification of whether a point is an out-of-control signal was carried out, the process flow was drawn with confidence intervals for one specific example of parameter configuration (Hastie, 2009). The points drawn in red are inclusions, the points in gray are the points of the process without failure, and the gray box is the confidence interval for the moving average. The moving average is marked in black.

The algorithm used is surprisingly good at detecting this type of interference. The results are excellent if we use a parameter to create a confidence interval in the model (all measures show a perfect fit).

Figure 7. *Process course separated with confidence intervals.*



Source: *Own creation.*

5. Work on Real Data

Because the analytical engine was supposed to be used to control the process on the production line, the dependence of readings from temperature, humidity, pressure, gyroscope, and acceleration of moving elements on the belt speed was analyzed. For

this purpose, tests were carried out with different belt speeds ranging from 4 to 20 m/s. This was to see how the speed settings affected the readings from the sensors.

6. Clean Up your Data

The readings had different time stamps because all available sensors recorded the data automatically and independently. As a result, almost every database record contained deficiencies caused by the fact that there was no change in the sensor value at a given moment, and only then were the values recorded. The data was supplemented (filled) as follows to remove this inconvenience. If there were no values in a given row, they were supplemented with the next non-empty value of the readings of a given sensor, which de facto reflects the actual state.

This section presents models of the relationship between temperature, pressure, humidity readings, and belt speed.

Table 1. *Fitting models on a training set*

model.	**RMSE**	**R2**
temp_engine_rf	0.6875365	0.7828791
Hum_engine_rf.	0.6724879	0.7158981
Temp_central_rf	0.1753435	0.7069418
Hum_central_rf	1.6751874	0.4874685
Temp_rf	0.1753435	0.7069418
Hum_rf	0.3917242	0.5144561
Press_central	0.2368255	0.6674590
Press_engine	0.2397195	0.6512764
Press_rf	0.2851091	0.6475081

Source: *Own creation.*

On the training set, the best model in context was for the engine temperature, where the coefficient of determination is 84%. On the other hand, in the context of RMSE, the best model is for the temperature, where the error is 0.17 degrees.

Table 2. *Fitting models on a test set*

model.	**RMSE**	**R2**
temp_engine_rf	0.3251846	0.8428052
Hum_engine_rf.	0.6889340	0.7826453
Temp_central_rf	0.6732453	0.7149582
Hum_central_rf	1.6983192	0.4810081
Temp_rf	0.1752657	0.7080059
Hum_rf	0.3964886	0.5083236
Press_central	0.2364094	0.6690395
Press_engine	0.2391754	0.6490734
Press_rf	0.2845377	0.6527929

Source: *Own creation.*

The results of matching the models on the test set show that the best model in the context of R^2 is the model for the engine temperature and, due to the RMSE, for the jaw temperature.

7. Neural Network for Parcel Classification

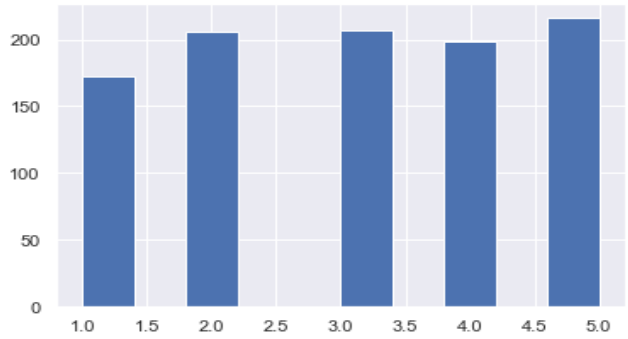
A model built based on measurement data taken during the process on the belt classifies the type of package (Pham *et al.*, 2022).

Loading and reviewing data:
The data comes from a simulation of a production line. Each record in the loaded table from the file contains the complete set of measurement data for that package. Important data are the object's humidity, volume, and mass, which are measured in the process.

The measurement data contains more than one million records. This is far too much for training a model with three predictors. That is why we randomize samples of 1000 records from the entire data pool for the training, test, and validation sets. Then, after determining that the extracted sets are disjoint, we examine the correlation of the predictors. There is a correlation between the weight of the box and the humidity and volume. This is natural and in line with basic knowledge of physics.

However, from a statistical point of view, this correlation is not so significant that we should be concerned about it and take appropriate measures. Before training the model, we check if the training set has an equal representation of classes.

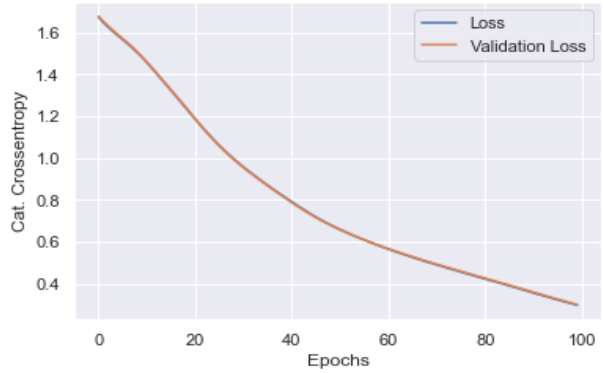
Figure 8. *Checking whether the training set has an equal representation of classes.*



Source: *Own creation.*

Feature extraction and scaling:
We create a function that extracts the necessary data from the sets and returns it as the neural network expects (Kłosowski, 2023).

Figure 9. *Loss and validation Loss*

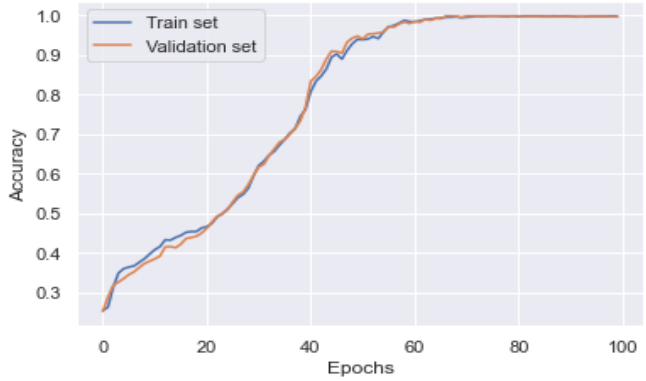


Source: *Own creation.*

The prepared data sets are scaled with the StandardScaler object, i.e., transformed to a normal distribution with an expected value and standard deviation $\mu = 0\sigma = 1$.

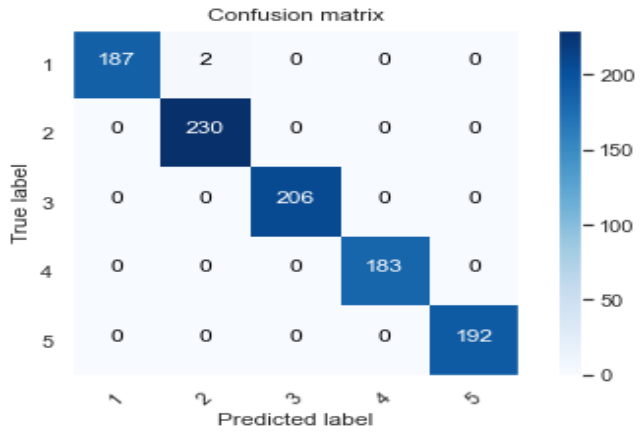
- Checking the data after scaling on set X and one hot encoding on flags Y.
- Building an artificial neural network for classification.
- Neural network in a 3-5-5 system.
- Optimizer: adam, learning_rate=0.001.
- Loss: categorical cross-entropy.
- Metric: Accuracy.
- Train the model.
- batch_size: 32.
- eras: 100.
- Validation every epoch.
- callback to save the best weights.

Figure 10. *Train set and validation set*



Source: *Own creation.*

Figure 11. *Confusion matrix*



Source: *Own creation.*

8. Conclusions, Proposals, Recommendations

The prepared model classifies boxes on the industrial conveyor belt very well. An accuracy of 99.8% gives us only two misclassified boxes in a set of 1,000 sizes. This is an excellent result.

The paper presents various approaches supporting the monitoring of the discrete production process, indicating solutions that allow the identification of solutions that can help employees. To sum up, there are tools on the market that support monitoring industrial lines.

The proposal of the authors of the article shows an individual approach to the analysis of results from different sensors. Proprietary solutions can be adapted to a specific type of production. We plan to implement our solution in further development. Considering the specificity of a particular manufacturer, it will be necessary to adapt the solutions to the sensors installed on the lines, and thus, there will be a need for communication between the sensors and the database.

References:

Hastie, T., Tibshirani, R., Friedman, J. 2009. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer.

JaamSim Development Team. 2016. JaamSim: Discrete-Event Simulation Software. Version 2016-14. URL: http://jaamsim.com. doi:10.5281/zenodo.57118.

Kłosowski, G., Rymarczyk, T., Niderla, K., Kulisz, M., Skowron, Ł., Soleimani, M. 2023. An LSTM network is used to monitor industrial reactors using electrical capacitance and impedance tomography – a hybrid approach. Eksploatacja i Niezawodność – Maintenance and Reliability, 25(1), 11. https://doi.org/10.17531/ein.2023.1.11.

Krol, K., Marciniak, A., Gudowski, J., Bojanowska, A. 2021. Intelligent Sensor Platform with Open Architecture for Monitoring and Control of Industry 4.0 Systems. European Research Studies Journal, Volume XXIV, Special Issue 2, 597-606.

Krol, K., Niderla, K., Kozłowski, E. 2023 Multisensor platform using industrial tomography for monitoring and control of technological processes. Przegląd Elektrotechniczny 1(2), 165-168.

Norena-Chavez, D., Thalassinos, E.I. 2023. Impact of big data analytics in project success: Mediating role of intellectual capital and knowledge sharing. Journal of Infrastructure, Policy and Development, 7(3).

Pham, A.H.T., Pham, D.X., Thalassinos, E.I., Le, A.H. 2022. The application of sem–neural network method to determine the factors affecting the intention to use online banking services in Vietnam. Sustainability, 14(10), 6021.

Simply. https://simpy.readthedocs.io/en/latest/.

Streamlight. https://docs.streamlit.io.