Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Modified Synthetic Dataset-Driven Multiple Linear Regression for Industrial Production Forecasting and Cost Estimation

Md. Abu Salman

abus02480@gmail.com

Northern University of Business and Technology Khulna https://orcid.org/0000-0003-4439-1317

Sumon Sarker

Northern University of Business and Technology Khulna

Krishnapada Mondal

University of Information Technology and Sciences

Md Al Mamun Sk

Northern University of Business and Technology Khulna

Litu Gain

Northern University of Business and Technology Khulna

Mehedi Hassan

Jashore University of Science and Technology

Md. Sohel Rana

Northern University of Business and Technology Khulna

Research Article

Keywords: Manufacturing automation, multiple linear regression, Machine learning, Production cost estimation, GUI-based prediction, Industrial decision-making

Posted Date: June 18th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-9131129/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Modified Synthetic Dataset-Driven Multiple Linear Regression for Industrial Production Forecasting and Cost Estimation

Md. Abu Salman[1,2*|0000-0003-4439-1317], Sumon Sarker[1|0009-0009-7344-4730], Krishnapada Mondal[3|0009-0005-2512-1604], Md Al Mamun Sk[1|0009-0008-5948-0420], Litu Gain[1|0009-0005-2766-3474], Mehedi Hassan[2|0000-0002-0223-0390], Md. Sohel Rana[1|0000-0001-5999-9435]

[1] Dept. of Electrical and Electronic Engineering (EEE), Northern University of Business and Technology Khulna (NUBTK), Khulna, Bangladesh
[2] Dept. of Electrical and Electronic Engineering (EEE), Jashore University of Science and Technology (JUST), Jashore, Bangladesh
[3] Dept. of Electrical and Electronic Engineering (EEE), University of Information Technology and Sciences (UITS), Dhaka, Bangladesh
*Corresponding author's email: abus02480@gmail.com; m241104.eee@student.just.edu.bd

Abstract. In the era of fast industrialization and urbanization manufacturing sectors need sophisticated automation to guarantee effective production monitoring and cost estimation. This work presents a Python-based graphical user interface (GUI) using multiple linear regression (MLR) in machine learning to maximize decision-making in industrial production. To precisely forecast production quantity (PQ), unit production cost (PC), total PC per item, and total manufacturing expenses on a fixed profit margin (PM) set up by the industry, the proposed approach incorporates theoretical assumption data nominated as modified synthetic dataset (MSD) of proportional industrial variables based on the economics of industrial operation. Embedded with a GUI, the suggested Python-based computational model provides high-accuracy production prediction and real-time cost estimation. Precise predictions are ensured by MSD of experimental data showing an exceedingly low mean square error (MSE) of 1.210139×10^{-23} and coefficient of determination (R^2) values close to 1.0. Moreover, a significant comparison of comparable machine learning (ML) techniques, featuring a combination of linear, regularized, kernel-based, and ensemble learning methods, was conducted on the same platform to assess the proposed MLR model as the optimal choice. By lowering hand computations and errors, this automated approach greatly improves manufacturing cost estimation and planning. For cost engineers and production managers, the GUI is a useful tool since it helps enable complex industrial decision-making and resource control in a very straightforward way. The results of this study show that the suggested GUI model is a dependable and useful tool for contemporary industrial sectors, therefore promoting better operational efficiency and cost-effectiveness in the case of the modern manufacturing sector.

Keywords: Manufacturing automation; multiple linear regression; Machine learning; Production cost estimation; GUI-based prediction; Industrial decision-making.

1 Introduction

The fast changes in modern manufacturing systems that come with Industry 4.0 have changed how people plan industrial production, estimate costs, and make decisions about how to run a business. Industries can now look at a lot of manufacturing data thanks to

advanced technologies like machine learning, big data analytics, and cyber-physical systems. This helps them make their operations more efficient and cut costs. Being able to accurately predict production output and manufacturing cost is especially important for staying competitive in global markets because it has a direct impact on pricing strategies, supply chain planning, and investment decisions. Most of the time, traditional methods for estimating industrial costs depend on manual calculations and historical data records, which are not always easy to find and not always accurate (Cho & Park, 2013). As manufacturing systems get more complicated, these old methods have a hard time capturing nonlinear relationships between different economic and production factors. As a result, data-driven predictive methods have become important tools for making better predictions about costs and production in today's factories (Więcek et al., 2019; Soleimani et al., 2023).

Recent studies have shown that machine learning algorithms can greatly improve the accuracy of predicting production and estimating manufacturing costs. For example, (Ning et al., 2020a) suggested a cost estimation model based on deep learning that uses convolutional neural networks (CNN) to guess how much it will cost to make something based on its geometric features and voxel-based representations of its parts. Their research showed that deep learning models can find complicated links between the shape of a part and its manufacturing cost. This makes it easier to predict how well complicated industrial parts will work. But these kinds of methods usually need a lot of data and computing power, which not all manufacturing companies can afford. (Ning et al., 2020c) subsequently developed a feature-based and process-based methodology for manufacturing cost estimation that amalgamates machining characteristics and process parameters to enhance the accuracy of production cost assessments. The method improved estimation accuracy, but it needs detailed feature recognition and modeling for each process, which makes it less useful for general manufacturing situations.

Along with deep learning methods, some researchers have looked into machine learning and statistical methods for predicting costs in engineering and manufacturing systems. For instance, (Soleimani et al., 2023) created a machine learning framework that uses data fusion to predict production costs in Industry 4.0 settings by combining data from different sources. This shows how important it is to combine data from different sources to make accurate cost predictions. Some studies have looked at machine learning models to help with product design and figuring out how much it will cost to make during the early stages of design. (Ma'ruf et al., 2024) put forward a machine-learning framework that uses Artificial Neural Networks (ANN) and Support Vector Regression (SVR) to figure out how much it will cost to design a product for make-to-order industries using CAD features and design parameters (Ning et al., 2022). These methods show promise for making accurate predictions, but they often depend on detailed design data and complicated feature extraction processes that may not be available during the early stages of production planning. Additionally, a lot of current models need big historical datasets to work well, which is a problem for industries that don't have a lot of data. Recent research also shows that structured datasets are necessary for training machine learning models well. However, the fact that there aren't many organized industrial datasets is still a big problem for research into estimating manufacturing costs (Hennebold et al., 2022).

1.1 Related Works and Research Gap

Regardless of machine-learning-based cost estimation has come a long way (Mandolini et al., 2024), there are still some problems with current research. A lot of methods depend on complicated algorithms, like deep learning models (Ning et al., 2020b, 2020c), even by using regression-based analysis model (Machfiroh & Maulida, 2025), some are depend on another ML model including ensemble based models (L. Chen et al., 2025; Machfiroh & Maulida, 2025) that need a lot of training data and are expensive to run. Other studies mostly look at theoretical modeling and don't give businesses the tools they need to make decisions. Also, most of the work that has already been done focuses only on predicting costs and doesn't include predicting production, economic relationships, optimization techniques (G. Zhang & Zhang, 2025) or easy-to-use computer interfaces. Table 1 summarizes a few examples of studies on estimating manufacturing costs and points out their imperfections compared to the method proposed.

Table 1. Comparison of Related Works in Manufacturing Cost Estimation.

Authors	ML / AI Model	Production Forecasting	Cost Estimation	Large Dataset Required	Model Interpretability	GUI / Tool Implementation
Cho & Park (2013)	Rule-based	×	✓	✓	✓	×
Więcek et al. (2019)	Activity-based model	×	✓	~	✓	×
Ning et al. (2020a)	Deep Learning (CNN)	×	✓	✓	×	×
Ning et al. (2020c)	DL + Machining features	×	✓	✓	×	×
Ning et al. (2022)	Feature-based ML	×	✓	✓	~	×
Soleimani et al. (2023)	Multi-source ML	×	✓	✓	~	×
Ma'ruf et al. (2024)	ANN / SVR	×	✓	✓	×	×
Mandolini et al. (2024)	ML Cost Modeling	~	✓	✓	~	×

Authors	ML / AI Model	Production Forecasting	Cost Estimation	Large Dataset Required	Model Interpretability	GUI / Tool Implementation
Construction ML studies	Ensemble ML	×	✓	✓	×	×
Proposed Work	**MSD-MLR Model**	✓	✓	×	✓	✓

1.2 Contribution of the Proposed MSD-MLR-GUI System

To overcome the shortcomings found in previous research, this study introduces a Modified Synthetic Dataset (MSD)-driven Multiple Linear Regression (MLR) framework, complemented by a graphical user interface (GUI) for forecasting industrial production and estimating costs. The proposed approach creates structured synthetic data based on basic industrial economic relationships between the required production quantity, market price, and profit margin. This is different from traditional machine learning models, which rely heavily on large historical datasets. This method makes it possible to create predictive models even when there aren't many or any historical industrial datasets.

This work's other important contribution is that it combines the predictive model with an interactive GUI-based industrial decision-support system. A lot of the research that has already been done only looks at how to make algorithms, and it doesn't give engineers or production managers any practical tools to use the model directly in real-world industrial settings. With the proposed GUI, users can enter industrial parameters like the number of units needed, the market price, the hours worked, and the profit margin. They will then get instant predictions of the number of units produced, the cost of producing each unit, and the total cost of manufacturing. The proposed system turns complicated industrial analytics into an easy-to-use computational platform by using statistical modeling, automated data generation, and user-friendly visualization. In addition, a prominent comparison of various machine learning techniques that are comparable was carried out on the same platform in order to evaluate the proposed MLR model as the best option. This comparison included a combination of linear, regularized, kernel-based, and ensemble learning tools.

The proposed MSD-MLR-GUI framework has a number of benefits over other machine-learning-based cost estimation methods:

- Reduced computational complexity through interpretable regression modeling.
- Synthetic dataset generation to overcome industrial data scarcity.
- Integrated production and cost prediction within a single analytical framework.
- User-friendly GUI implementation for real-time industrial decision support.

These characteristics render the proposed system especially well-suited for implementation in contemporary manufacturing settings that adhere to the tenets of Industry 4.0 and smart manufacturing.

The following describes how the remainder of the paper is set up: Section 2 gives a full explanation of the research methodology. Section 3 explores about how the algorithm and flowchart were made and how they fit with the suggested GUI. Section 4 discusses the outcomes that come from the GUI and compares different machine learning algorithms in a distinctive manner. Section 5 explores the platform's hardware and software features and how the GUI user interface works. Section 6 addresses the study's limitations and possible future directions, while Section 7 talks about the study's wider conclusions. The authors' statement about this study is in Section 8.

2 Methodology

2.1 System Architecture and Industrial Data Modeling

The suggested approach uses a machine-learning assisted GUI to create a data-driven framework for estimating the cost of industrial output. The technology combines synthetic manufacturing data generation of MSD, MLR modeling, cost computation, and visualization modules into a single decision-support system. The architecture is in line with the Industry 4.0 model, which uses big data analytics (BDA) and predictive models to help with planning and lowering costs in production. This kind of integration makes it possible to estimate the cost of manufacturing in real time depending on demand, market price, and operational time (Salman et al., 2025; Lee et al., 2015; Tao et al., 2018; Kusiak, 2018). There are five main steps in the proposed system's overall workflow is shown in figure 1.

i. ***Manufacturing Sample Dataset (e.g. MSD) generation***
ii. ***Data preprocessing and feature construction***
iii. ***Machine learning model development using MLR***
iv. ***Production cost estimation and operational calculations***
v. ***GUI-based visualization and reporting***



Fig. 1. The five main steps in the proposed system's overall workflow.

The MSD dataset is generated to simulate large-scale manufacturing production scenarios, following the concept of big-data-driven modeling in Industry 4.0 systems. The dataset contains **three primary industrial variables**:

- ***Quantity required per day,*** $\boldsymbol{Q_{req}}$
- ***Product market price,*** $\boldsymbol{MP}$
- ***Production quantity per day,*** $\boldsymbol{Q_{pro}}$

These parameters represent fundamental decision variables in manufacturing economics, where production output is influenced by demand level and market price fluctuations (Tao et al., 2018; Y. Zhang et al., 2017).

According to the Industry 4.0 paradigm by (Salman et al., 2025), big-data modeling uses enormous datasets to make smart production decisions and predictive analytics possible. So, the MSD dataset acts as big streams of industrial data to train predictive models that can help with planning production and estimating costs.

Mathematically, the dataset, D is represented as by equation (1).

$$D = \{(Q_{req}^{(i)}, MP^{(i)}, Q_{pro}^{(i)})\}_{i=1}^{N} \tag{1}$$

Where N denotes the total number of industrial samples used for training the predictive model.

The synthetic dataset specified in (2) is generated using linear numerical sequences:

$$\begin{cases} Q_{req} \in [100,10000] \\ MP \in [1,100] \\ Q_{pro} \in [120,10100] \end{cases} \tag{2}$$

This data modeling approach enables the system to simulate large-scale industrial production behavior and provides sufficient samples for machine learning training, which is essential for predictive manufacturing analytics (Bertsimas & Kallus, 2020; Kusiak, 2018) .

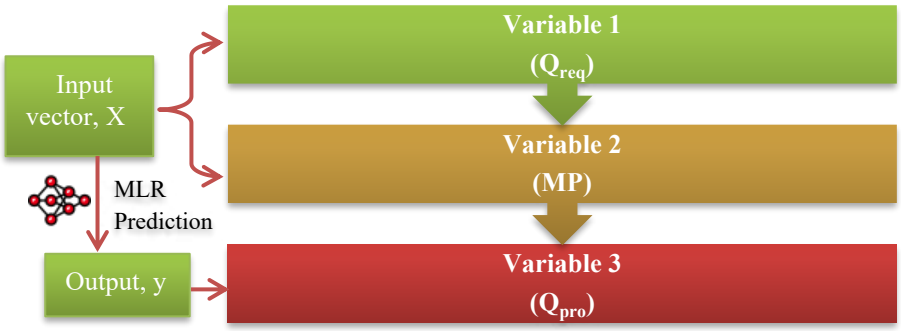


Fig. 2. MSD generation process with MLR integrated general workflow.

The NumPy linspace() function makes the MSD by creating evenly spaced numbers within a certain range. This method makes synthetic information about large-scale industrial production to train the predictive model.

1. Quantity Required Per Day (Demand)
The daily demand variable Q_{req} is generated by using (3):

$$Q_{req} = \text{linspace}(100,10000,10000) \tag{3}$$

This produces 10,000 uniformly distributed demand values ranging from 100 to 10,000 units per day, representing varying production requirements in an industrial environment.

2. Product Market Price
The market price variable MP is generated as the relation in (4):

$$MP = \text{linspace}(100,1,10000) \tag{4}$$

This creates 10,000 price samples decreasing from 100 BDT to 1 BDT, simulating dynamic market conditions where product prices vary across production scenarios.

3. Production Quantity Per Day
The production output variable Q_{pro} is generated using (5):

$$Q_{pro} = \text{linspace}(120,10100,10000) \tag{5}$$

This produces 10,000 production output values ranging from 120 to 10,100 units per day, representing different manufacturing capacity levels corresponding to demand variations.

The MSD generation process makes a lot of synthetic industrial data with 10,000 samples. In this data, demand, market price, and production output all change in a systematic way. This dataset is then used to train and test the Multiple Linear Regression (MLR) model, which is used to estimate production costs. Figure 2 shows three industrial parameters that are connected: Q_{req}, MP, and Q_{req}, MP. Where the Q_{req} and MP are the independent input variables as X, and depending on these variables the output production as Q_{pro} is calculated by the adoption of MLR. As the MP goes down along the horizontal axis, the Q_{req} goes up. This shows that when prices go down, demand goes up. The Q_{pro} also goes up at the same time to keep a buffer. This means that production has to be at least 15 to 20 percent more than what is needed to deal with changes and higher demand. This model curve shown in figure 3 shows that the price and quantity of production are related in the opposite way. This means that there will always be more supply than demand.

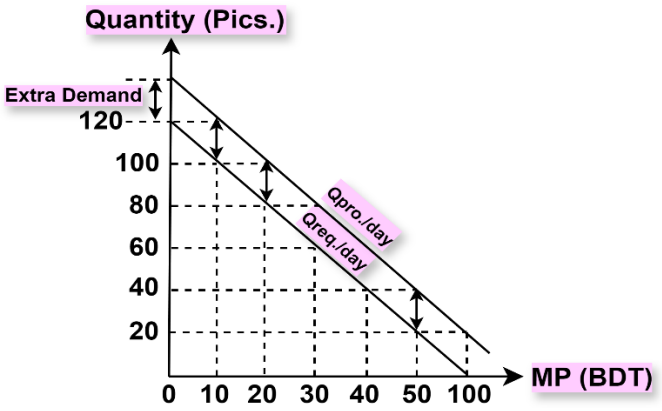


Fig. 3. The visual representation of the industrial correlation among the three variables.

2.2 Data Pre-Processing and Feature Construction

Data preliminary treatment is a very important stage in making predictive models in manufacturing analytics more reliable and accurate. The proposed methodology initially uses the Pandas framework to turn the obtained MSD dataset into structured CSV-based industrial datasets. After that, the three datasets are combined to make the whole feature matrix.

The input feature vector (X) is defined as by (6).

$$X = [Q_{req}, MP] \tag{6}$$

while the target output variable is y in equation (7).

$$y = Q_{pro} \tag{7}$$

The combined dataset as a matrix format can therefore be expressed by (8) to (9).

$$X = \begin{bmatrix} Q_{req}^{(1)} & MP^{(1)} \\ Q_{req}^{(2)} & MP^{(2)} \\ \vdots & \vdots \\ Q_{req}^{(N)} & MP^{(N)} \end{bmatrix} \tag{8}$$

$$y = \begin{bmatrix} Q_{pro}^{(1)} \\ Q_{pro}^{(2)} \\ \vdots \\ Q_{pro}^{(N)} \end{bmatrix} \tag{9}$$

To ensure impartial machine learning training and validation, the dataset is divided into **training and testing subsets** using the hold-out validation technique expressed by (10).

$$D = D_{train} \cup D_{test} \tag{10}$$

where

$$| D_{train} | = 0.8N, \qquad | D_{test} | = 0.2N$$

This 80–20 data partitioning strategy is often used in predictive analytics to keep models from overfitting and make sure they can generalize (Gray, 2002; Hastie et al., 2009). The training dataset is used to find the regression parameters, and the testing dataset is used to find the accuracy of the predictions using statistical performance metrics.

2.3 Multiple Linear Regression Model for Production Prediction

The proposed system uses MLR to figure out how much to produce based on market conditions and demand levels. MLR is a common way to use statistics to model the relationship between a dependent variable and several independent variables (Gray, 2002; Hastie et al., 2009; Witten et al., 2011). The MLR model can be written in general form as in (11).

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \epsilon \tag{11}$$

where

- y = dependent variable
- x_1, x_2= independent variables
- β_0= intercept term
- β_1, β_2= regression coefficients
- ϵ= error term.

In the proposed manufacturing model, the regression equation becomes the following format in (12). In real-world manufacturing systems, output depends on demand, market price, machine capacity, labor availability, operational time, and efficiency. But for the proposed predictive model, an idealized linear relation is assumed that MP is

where demand drives most of the production output and market price is a secondary factor remaining the others factors to be constant.

$$Q_{pro} = \beta_0 + \beta_1 Q_{req} + \beta_2 MP + \epsilon \tag{12}$$

The model parameters $\beta_0, \beta_1, \beta_2$ are estimated using the Ordinary Least Squares (OLS) optimization technique given in (13), which minimizes the sum of squared prediction errors (Gray, 2002).

$$\min_{\beta} \ \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{13}$$

where y_i denotes the true or real value of the dependent variable for the i^{th} sample in the dataset. It comes directly from the actual dataset. And $\hat{y}_i$ represents the predicted value generated by the regression model for the i^{th} sample. It is calculated using the regression equation in (14).

$$\hat{y}_i = \beta_0 + \beta_1 Q_{req}^{(i)} + \beta_2 MP^{(i)} \tag{14}$$

The trained MLR model then predicts the daily production quantity by (15) for any new manufacturing input parameters provided through the GUI interface.

$$\hat{Q}_{pro} = f(Q_{req}, MP) \tag{15}$$

Mean Squared Error (MSE) and the coefficient of determination (R^2) are two common performance metrics used to evaluate the predictive power of regression models in machine learning (Hastie et al., 2009; Witten et al., 2011). The difference between the actual value and the predicted value gives the prediction error shown by (16):

$$error_i = y_i - \hat{y}_i \tag{16}$$

This error is used in the MSE calculation. The MSE is defined as in (17). Where, RMSE is the square-root of MSE.

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{17}$$

where n is the number of testing samples.

The coefficient of determination by (18) measures the proportion of variance explained by the model.

$$R^2 = 1 - \frac{\sum (y_i - \hat{y}_i)^2}{\sum (y_i - \bar{y})^2} \tag{18}$$

Higher R^2 values (close to 1) indicate stronger predictive capability of the regression model for production estimation.

2.4 Production Cost Estimation Model

The system calculates the cost of production and the efficiency of operations for each item after using the MLR model to predict how many items will be made each day. To begin, the predicted daily production quantity is obtained through (19).

$$Q_{pro_day} = round(\hat{Q}_{pro}) \tag{19}$$

The unit production cost equation in (20) is calculated based on the market price and profit margin.

$$Unit_PC = MP\left(1-\frac{PM}{100}\right) \tag{20}$$

where

- PM = desired profit margin (%)
- MP = market price per unit.

The total production cost for each product item is then computed by (21).

$$Total_PC_{item} = Q_{pro_day} \times Unit_PC \tag{21}$$

If multiple product items are processed simultaneously, the overall manufacturing cost is obtained by summing all item costs given in (22).

$$Total\ PC_{All\ items} = \sum_{i=1}^{m} Total\ _PC_{item}^{(i)} \tag{22}$$

where m represents the number of product items entered by the user. The model also estimates the effective hourly production rate based on active working hours that is given in (23).

$$Production\ Rate_{hour} = \frac{Q_{pro_day} \times H_{active}}{24} \tag{23}$$

where, H_{active}= active working hours per day.

This formulation enables manufacturing managers to evaluate production efficiency in relation to operational time, which is essential for industrial scheduling and work-force allocation in smart manufacturing settings (Kusiak, 2018; Lee et al., 2015).

2.5 GUI-Based Decision Support and Visualization System

The predictive model is built into a Python-based GUI made with the Tkinter frame-work so that it can be used in real-world industrial settings. The GUI lets production

managers enter operational parameters and get cost estimates right away. Digital manufacturing and smart factory systems are using these kinds of human-machine interfaces extensively (Tao et al., 2018; G. Zhang & Zhang, 2025).

The GUI consists of the following functional modules:

1. Industrial Input Module

Users input the following parameters:

- Quantity required per day
- Market price per unit
- Active working hours
- Profit margin

The system allows multiple product entries to simulate multi-product manufacturing environments.

2. Prediction Engine

The trained MLR model processes the input parameters and generates the predicted production quantity by the following relation in (24)

$$\hat{Q}_{pro} = f(Q_{req}, MP) \tag{24}$$

3. Cost Calculation Engine

The system automatically calculates:

- Unit production cost
- Total item production cost
- Hourly production rate
- Overall production cost.

4. Visualization Module

A 3-dimensional (3D) visualization is generated to illustrate the relationship among demand, market price, and production output. The 3D representation is defined by (25).

$$Z = f(X, Y) \tag{25}$$

where

$$X = Q_{req}, Y = MP, Z = Q_{pro}$$

Such visualization helps decision-makers understand production dynamics under varying market conditions, which is a key aspect of predictive manufacturing analytics (Bertsimas & Kallus, 2020; Kusiak, 2018). The 3D illustration shows how daily demand (the amount needed each day), market price, and predicted production output are all related. Figure 4 shows how production changes with different market conditions and levels of demand by combining the surface and scatter representations.

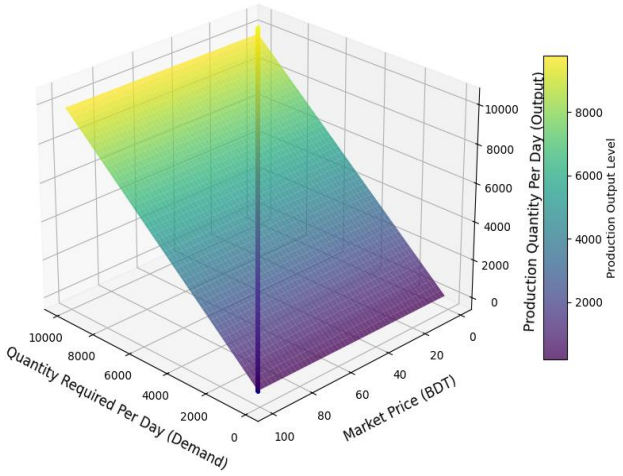


Fig. 4. Three-dimensional visualization showing the relationship among daily demand (quantity required per day), market price, and production output.

5. Automated Reporting Module

The system also generates PDF-based production cost invoices containing

- production inputs,
- model performance metrics,
- cost calculations,
- timestamped operational records.

This feature supports digital documentation and industrial record keeping, which are important components of smart manufacturing systems (Lee et al., 2015).

3 Algorithm and Flowchart Development

3.1 Algorithm: MLR-Based Production Cost Estimation

To facilitate data-driven decision-making in contemporary manufacturing settings, an algorithm utilizing MLR is created to estimate production output and the corresponding production costs. Predictive models based on machine learning are now an important part of smart manufacturing and Industry 4.0 systems. These systems use operational data to improve production planning, cost management, and resource allocation (Lee et al., 2015; Tao et al., 2018; Kusiak, 2018). The algorithm of MLR-based production cost estimation has the following inputs and outputs parameters and processing steps given in figure 5.

Input: Q_{req}, MP, H_{active}, PM
Output: Q_{pro}, Unit production cost, Total production cost, Production rate

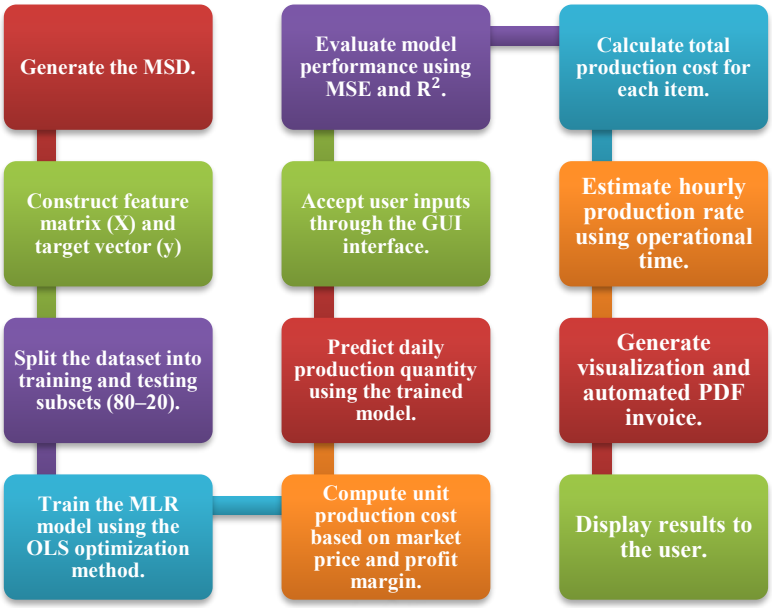


Fig. 5. Step-by-step procedures of the algorithm regarding MLR-based production cost estimation.

The suggested algorithm combines data generation, model training, prediction, and cost estimation into one computing framework. First, an MSD is made to show how things work in the real world. There are three main variables in the dataset: the amount of product needed each day (demand), the market price of the product, and the amount of product produced each day. These variables include both the economic and operational parts of the manufacturing process. Demand and market price are the independent input variables, and production quantity is the target output variable. In manufacturing analytics, similar data-driven modeling methods are widely used to find connections between production factors and system performance (Tao et al., 2018; Y. Zhang et al., 2017). After the dataset is made, the data are put into a feature matrix and a target vector so that the regression model can learn. The dataset is then split into two parts: a training set and a testing set. The split ratio is usually 80–20. This strategy lets the model learn how variables are related while keeping some of the data for performance testing. In machine learning, it is common to split up datasets to make sure that models can be used in other situations and to avoid overfitting during model development (Witten et al., 2011). Then, the MLR model is trained using the OLS optimization method. This method finds the best regression coefficients by making the squared difference between predicted and actual values as small as possible. Linear regression models are widely used in predictive analytics because they are easy to understand, easy to use, and quick to compute (Gray, 2002). After being trained, the model can figure out how much to make based on different levels of demand and market prices.

To check how reliable the trained model is, that are employed statistical performance indicators like the MSE and the coefficient of determination. These metrics give us

numbers that show how accurate our predictions are and how well our model fits the data. To make sure that the predictive model can reliably capture the relationship between variables, performance evaluation is an important part of machine learning workflows (Witten et al., 2011; Hastie et al., 2009).

After checking the regression model, the algorithm moves on to the user interaction stage. In this stage, production managers give operational inputs through the GUI. The trained MLR model uses these inputs to guess how much will be produced each day. The algorithm then does the necessary economic calculations to figure out the total and unit production costs for the manufacturing process.

The algorithm not only makes numerical predictions, but it also makes visual analytical outputs to help people understand how production changes over time. A three-dimensional image is made to show how demand, market price, and production output are all related. Visual analytics is an important part of industrial decision-support systems because it helps engineers and production planners understand how systems work and how they change over time (Kusiak, 2018).

The algorithm then puts together the predicted costs and production into an automated report that can be saved as a PDF-based production invoice. More and more, automated reporting systems are being used in digital manufacturing settings to make things more clear, easier to track, and more efficient (Lee et al., 2015; Tao et al., 2018).

The suggested algorithm gives a data-driven way to estimate production and analyze costs by combining machine learning methods with industrial economics. The system helps with efficient production planning and cost management in modern manufacturing settings by combining predictive modeling, visualization, and automated reporting.

3.2 Flowchart Development

The flowchart for the proposed system shows how the MLR-based production cost estimation algorithm works step by step. To start the process, the system is set up and the MSD is created. This step gets the industrial data ready for training the model. After that, the dataset is cleaned up and split into two parts: one for training and one for testing. Then, the MLR model is trained on the training dataset and checked for accuracy using metrics like MSE and R^2. After the predictive model has been tested and found to be accurate, the system moves on to the user interaction stage.

Here, production managers enter operational parameters through the GUI. After getting the inputs, the trained regression model makes a guess about how much will be made each day. The system uses this prediction to figure out the cost of making one unit, the total cost of making all the items, and the rate of production per hour. These calculations are based on the mathematical rules set out in the algorithm. After that, the system makes visualization outputs, like a 3D plot that shows how demand, market price, and production output are related.

Finally, the results are put together and sent out as a PDF-based production cost invoice, which ends the decision-support workflow. The flowchart shows the entire operational cycle of the suggested predictive production cost estimation system, which combines machine learning, economic modeling, and industrial decision support. The distinctive flowchart is given in figure 6.

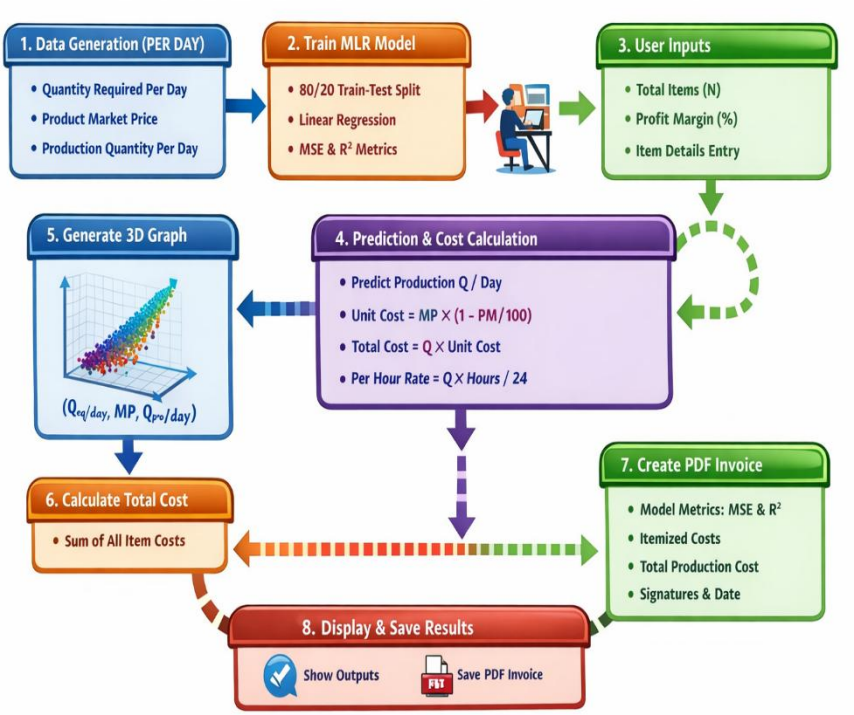


Fig. 6. Flowchart of the proposed whole GUI system.

4 Results and Distinctive Comparison

4.1 Results Analysis of the Proposed GUI System

The system used the MSD created in the Python environment to evaluate how well the proposed MLR-based production cost estimation GUI system worked. The results in Tables 2 and 3 show how well the model can predict what will happen in two different types of industrial markets: one where the MP goes up and one where it goes down. In both cases, the profit margin, PM stays the same at 15% for all production items. And the active working hours is 8 for all the items. These scenarios mimic authentic manufacturing settings characterized by fluctuating market prices and variable production demand driven by operational necessities. Smart manufacturing and Industry 4.0 systems are using more and more data-driven predictive models to look at these changes in production and help with decision-making (Kusiak, 2018; Lee et al., 2015; Tao et al., 2018).

4.1.1 Incremental Market Price Scenario with Fixed PM

Table 2 shows the results when the market price goes up slowly from 100 BDT to 1000 BDT and the number of pieces that need to be made each day ranges from 500 to 5000. The predicted daily production quantity (Q_{pro}) has a strong linear relationship with the demand variable (Q_{req}). For instance, if the demand goes up from 500 to 5000 pieces, the predicted number of pieces that will be made also goes up from 523 to 5050. This proportional trend shows that the trained MLR model accurately captures the relationship that is hidden in the MSD dataset.

From a statistical point of view, the differences between the required and predicted production amounts stay very small in all test cases. The difference between Q_{req} and Q_{pro} stays within a small range, which shows that the regression model is stable and reliable for estimating production tasks. This is what we expect because linear regression works best for modeling systems where variables have a strong linear dependency (Gray, 2002).

Table 2. Some results of the proposed GUI system program for incremental MP with fixed PM of 15% for all items.

Item No. (s)	$Q_{req./day}$ (Pieces)	MP (BDT)	$Q_{pro./day}$ (Predicted Pieces)	$Q_{pro./hour}$ (Predicted rate in 24 hours)	Calculated PC_{unit} (BDT)	Total PC_{item} = $Q_{pro./day}$× PC_{unit} (BDT)	GUI Performance Metrics	
							MSE	R^2
Item 1	500	100	523	174	85	44455	1.2101e-23	1.00
Item 2	600	200	623	208	170	105910	1.2101e-23	1.00
Item 3	700	300	723	241	255	184365	1.2101e-23	1.00
Item 4	800	400	823	274	340	279820	1.2101e-23	1.00
Item 5	900	500	922	307	425	391850	1.2101e-23	1.00
Item 6	1000	600	1022	341	510	511220	1.2101e-23	1.00
Item 7	1500	700	1525	508	595	907375	1.2101e-23	1.00
Item 8	2000	800	2028	676	680	1379040	1.2101e-23	1.00
Item 9	3000	900	3035	1012	765	2321775	1.2101e-23	1.00
Item 10	5000	1000	5050	1683	850	4292500	1.2101e-23	1.00

The total production cost values also clearly show how the results will affect the economy. The unit production cost goes up in direct proportion to the market price because the profit margin is set at 15%. This means that the total cost of making each item goes up a lot when demand and market price go up. For example, the total cost of making Item 1 goes up from 44,455 BDT to about 4.29 million BDT for Item 10. This trend shows that the proposed system accurately models the link between the size of production and the value of the product in the market.

The hourly production rate is another important operational indicator that the system gives us. As demand for the ten items goes up, the predicted hourly production rate goes up from 174 pieces/hour to 1683 pieces/hour. This trend shows that the system

can turn daily production forecasts into useful operational manufacturing metrics that can be used for scheduling production and planning resources.

One important thing to notice in Table 2 is that the performance metrics are the same for all test cases. The MSE stays very low at about 1.2101×10^{-23} for all items, and the R^2 stays at 1.00. These numbers show that the predicted and expected outputs are almost the same. The fact that these metrics are the same in different experimental situations suggests that the proposed GUI system keeps its predictive accuracy stable even when industrial parameters change. This is a big plus for real-world manufacturing applications (Hastie et al., 2009).

4.1.2 Decremental Market Price Scenario with Fixed PM

The second experimental scenario in Table 3 tests the system when market prices are falling. In this case, the price goes down from 1000 BDT to 100 BDT while keeping the same 15% profit margin. Even though the price trend has changed, the predicted production amounts are still very close to the demand amounts. This shows that the trained regression model is strong.

The total cost of production goes down a lot as the market price goes down, even though the amounts produced stay about the same as they were in Table 2. The total production cost for Item 1 is 436,900 BDT when the market price is 1000 BDT. However, when the price drops to 100 BDT, the total cost for Item 10 is about 430,015 BDT, even though the quantity produced is much higher. This observation shows that market price has the biggest effect on estimating total production cost.

The model performance metrics stay the same for all cases, which is another important finding. For every item, MSE is about 1.2101×10^{-23} and R^2 is 1.00. This consistency shows that the suggested MLR-based system keeps its high predictive accuracy and stability even when economic conditions change, which is a good thing for industrial decision-support systems.

Table 3. Some results of the proposed GUI system program for decremental MP with fixed PM of 15% for all items.

Item No. (s)	$Q_{req./day}$ (Pieces)	MP (BDT)	$Q_{pro./day}$ (Predicted Pieces)	$Q_{pro./hour}$ (Predicted rate in 24 hours)	Calculated PC_{unit} (BDT)	Total PC_{item} = $Q_{pro./day}\times PC_{unit}$ (BDT)	GUI Performance Metrics	
							MSE	R^2
Item 1	500	1000	514	171	850	436900	1.2101e-23	1.00
Item 2	600	900	616	205	765	471240	1.2101e-23	1.00
Item 3	700	800	718	239	680	488240	1.2101e-23	1.00
Item 4	800	700	820	273	595	487900	1.2101e-23	1.00
Item 5	900	600	921	307	510	469710	1.2101e-23	1.00
Item 6	1000	500	1023	341	425	434775	1.2101e-23	1.00
Item 7	1500	400	1528	509	340	519520	1.2101e-23	1.00
Item 8	2000	300	2033	678	255	518415	1.2101e-23	1.00
Item 9	3000	200	3042	1014	170	517140	1.2101e-23	1.00

Item 10	5000	100	5059	1686	85	430015	1.2101e-23	1.00

4.2 Distinctive Comparison with Other Machine Learning Models

Table 4 compares the performance of five machine learning models trained on the same MSD dataset: MLR, Ridge Regression (RR), Support Vector Regression (SVR), Random Forest (RF), and Extreme Gradient Boosting (XGBoost). This is to further prove that the proposed method is reliable. These algorithms use a mix of linear, regularized, kernel-based, and ensemble learning methods, which are often used in predictive analytics and industrial modeling (Breiman, 2001; Cortes & Vapnik, 1995; T. Chen & Guestrin, 2016).

The MLR model is the best at predicting things overall out of all the models that were looked at. It gets an MSE of 1.180323×10^{-23} and a very low RMSE of 3.435584×10^{-12} while keeping a perfect R^2 value of 1.000000. These very low error values show that the model does a great job of showing how demand, market price, and production quantity are related.

Even though Ridge Regression gets a perfect R^2 score, its error values are much higher than those of the MLR model. This indicates that the regularization utilized in Ridge Regression does not substantially enhance predictive performance for the MSD dataset, which already demonstrates a robust linear structure.

The Random Forest model also gives a value close to 1 for R^2, but it still has higher error values than MLR. Ensemble tree-based models are usually good for modeling complicated nonlinear relationships, but they can make things more complicated than they need to be when the relationship between the variables is basically linear.

Support Vector Regression, on the other hand, does much worse, with a huge MSE of 6.59×10^{6} and a R^2 value of only 0.203975, which means it can't predict the data well. This suggests that kernel-based models might not work well with the simple linear relationships in the MSD dataset.

The XGBoost model does pretty well, with a R^2 value of 0.999983, but its error values are still much higher than those of the MLR model. XGBoost is a powerful boosting algorithm, but it may not be worth the extra work if a simpler linear approach can model the problem just as well.

The comparative analysis shows that the MLR-based GUI system has better predictive accuracy with less computational complexity. This makes it a good choice for estimating production costs in real time and for decision-support applications.

Table 4. Comparison of different ML techniques on the same MSD.

ML Model Type	ML Techniques	MSE	R^2	Annotations of Fitness
Linear learning method	**MLR**	**1.180323e-23**	**1**	**Best fitted**
Regularized linear method	RR	1.937710e-15	1	Moderately fitted

Kernel-based learning method	SVR	6.592929e+6	0.203975	Lowest fitted
Ensemble learning method (bagging)	RF	2.838623e-01	1	Moderately fitted
Ensemble learning method (boosting)	XGBoost	1.402392e+02	0.999983	Lowermost fitted

5 GUI Interface with Hardware and Software Specifications

5.1 GUI Interface Modeling

The suggested system has a GUI based on Python that lets users interact with the production cost estimation model through a well-organized and easy-to-use interface. The GUI was made to look like a simple industrial production planning dashboard, where users can enter operational parameters, do calculations, see how data relates to each other, and make automated reports. The GUI has a number of functional parts that work together to support the workflow of the proposed framework for estimating production costs. Figure 7 shows the snapshot of the Python code in the PyCharm software's interface that makes the GUI system work, and Figure 8 shows what the developed GUI based user interface looks like from the outside.

```
from datetime import datetime
import numpy as np
import pandas as pd
import tkinter as tk
import matplotlib.pyplot as plt
from tkinter import messagebox, filedialog
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from mpl_toolkits.mplot3d import Axes3D
from fpdf import FPDF


# ====================================================
# Generate sample data (PER DAY VERSION)
# ====================================================
quantity_required_per_day = np.linspace( start: 100,  stop: 10000,  num: 10000)
product_market_price = np.linspace( start: 100,  stop: 1,  num: 10000)
production_quantity_per_day = np.linspace( start: 120,  stop: 10100,  num: 10000)

pd.DataFrame({'quantity_required_per_day': quantity_required_per_day}).to_csv( path_or_buf: 'quantity_required_per_day.csv',
                                                                               index=False)
pd.DataFrame({'product_market_price': product_market_price}).to_csv( path_or_buf: 'product_market_price.csv', index=False)
pd.DataFrame({'production_quantity_per_day': production_quantity_per_day}).to_csv( path_or_buf: 'production_quantity_per_day.csv',
                                                                                   index=False)

if __name__ == "__main__"
```

Fig. 7. A snapshot of the GUI based MLR program in PyCharm platform.

The **Total Items to Produce** input field is the first part of the interface. It lets the user say how many industrial items will be used in the calculation of production. This parameter tells the interface how many separate product entries to make. In real-world industrial situations, production planning often includes more than one type of product or batch. This input field gives it the opportunity to handle more than one production item in a single operation.

The GUI has a **Set Items** button that turns on this feature. When the user types a number into the **Total Items to Produce** field and clicks the "Set Items" button, the system automatically creates the right number of item input sections in the interface. If the user says they want two items, for instance, the GUI will automatically make two separate input panels where they can type in the information they need for each item. This dynamic layout system makes the system easier to use and makes sure it can easily adjust to different production situations.

The **Profit Margin (%)** field is another important input parameter in the GUI. It lets the user set the expected industrial profit margin before any calculations are done. The system uses this parameter in its industrial-grade cost estimation formula to figure out the unit production cost and the total production cost. The system lets users set the profit margin, which makes it easier to plan finances and analyze costs for different business strategies.

The **Calculate Production Cost** button is what makes the system work. When you click this button, the GUI gathers all the input parameters, such as the required production quantity, market price, and profit margin. It then uses the MLR model and industrial cost calculation formulas to process them. The system then figures out a number of important outputs, such as the expected number of items produced, the hourly production rate, the cost of producing one item, and the total cost of producing all items. The GUI also calculates the total production cost for all items, giving you a complete picture of the production costs.

The GUI also has a **Reset All** button that clears all input fields and returns the interface to its default state for ease of use. This function lets users quickly start the calculation process over again without having to delete old values by hand.

The interface has a **Print Invoice** button to make it easier to keep records and report. This feature makes a professionally formatted PDF invoice or report that includes all the calculated production parameters and results on its own. The report that is made usually has information about the items, how many were made, what the market prices are, what the profit margins are, what the predicted outputs are, and what the total production costs are. Automated reporting like this is especially helpful for keeping track of records in factories and making decisions as a manager.

The **Generate 3D Graph** button is another visualization tool in the GUI. It makes a 3D graph that shows how the three main industrial variables of demand, market price, and production output are related to each other. This graph helps users understand better how changes in demand and market price affect how much the system can produce.

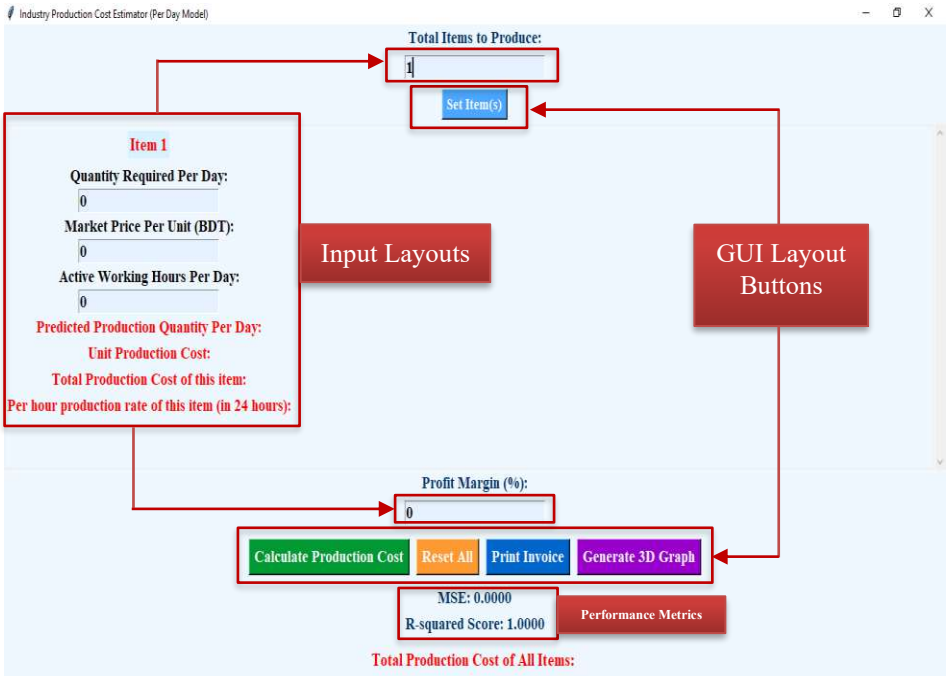


Fig. 8. A snapshot of the proposed GUI based user interface.

Lastly, the GUI interface shows model performance metrics right in the user interface. These include the MSE and the R^2. By showing these metrics, users can check the predictive model's accuracy and dependability while the system is running.

The developed GUI brings together input configuration, predictive computation, visualization, performance monitoring, and automated reporting into one interactive platform. This makes it useful for real-world industrial production planning and cost estimation tasks.

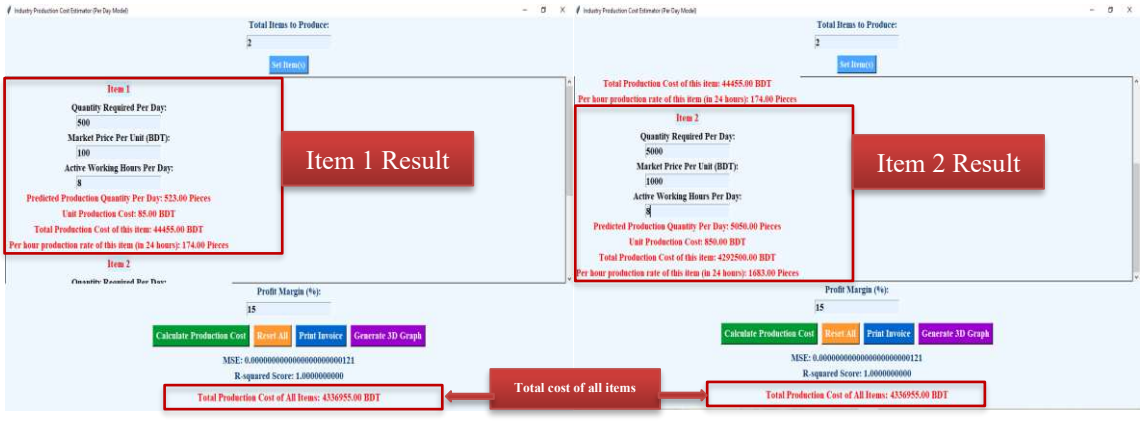


Fig. 9. A snapshot of the generated results in the proposed GUI based user interface.

Figure 9 shows an illustration of the results that were made and shown in the suggested GUI-based user interface. The figure shows how the system works by using two items from Table 2 as examples to show how the GUI does real-time calculations of production costs. In particular, the results match Item 1 with Q_{req} = 500 and Item 10 (correspondent to Item 2 in the snapshot) with Q_{req} = 5000, where the GUI calculates the expected production amounts, hourly production rates, and unit production costs based on the parameters that have already been set. The interface makes it easy to see the calculated production costs for both items. The total production cost for Item 1 is 44,455 BDT, and for Item 10 it is 4,292,500 BDT. The GUI system adds these numbers together on its own to find the total production cost, which comes to 4,336,955 BDT for the two items chosen. This number shows how useful the GUI system is for doing things like estimating industrial costs in real time, visualizing results, and automatically adding up costs. It also shows how well the proposed framework works for real-time production planning applications.

5.2 Specification of Hardware and Software Used in the System

This study's experiments were done on a personal computer with an Intel Core i5 processor, 8 GB of RAM, and a 64-bit operating system. This setup is a standard computing environment that you would find in most industrial offices and research labs. The PyCharm Integrated Development Environment (IDE) was used to run and test the GUI application that was made. This IDE has advanced debugging tools, code management features, and the ability to run projects quickly.

The system's implementation depends on a number of Python libraries and packages, each of which has a specific job to do in the application. The NumPy library was used to do mathematics and make the MSD, especially to make arrays and do math. It used the Pandas library to handle data and manage datasets. This included reading, organizing, and preprocessing production data.

The Tkinter package was the main framework for making windows, frames, labels, buttons, entry fields, and dialog boxes for GUI development. It used more Tkinter modules, like messagebox and filedialog, to add interactive notifications and file selection features.

The GUI used the Matplotlib library to make graphs, such as two- and three-dimensional plots that show how demand, market price, and production output are related. The mpl_toolkits.mplot3d module made it possible to create three-dimensional data visualizations in the GUI environment.

It used the Scikit-learn (sklearn) library to add machine learning features. The MLR model was made with the Linear Regression module, and the train_test_split function was used to split the dataset into training and testing sets. It used the mean_squared_error and r2_score functions to see how well the predictive model worked.

The datetime module was also used to make timestamps for system operations and to add to the automatically generated production reports. These libraries work together to make the Python GUI program fully functional. They let the generation of data, training of models, making of predictions, visualization of data, and automatic reporting of

results all happen in one software framework. Table 5 gives a tabular overview of all of these specifications.

Table 5. Overview of the Generic Hardware and Software Specifications.

Parts of the System	System Configurations	Packages Used	Functionality
Hardware	Intel Core i5 processor, a 64-bit operating system, and 8GB of RAM	Windows 10	System Software for the hardware
Software	Pycharm IDE (Community Edition)	Different packages	GUI Application

In the real world, a high-performance computer setup with advanced settings would speed up processing time by a lot, making sure that predictions and analysis happen in real time. The study can show how powerful the testing system is by going into detail about the computational setup. It can also make it clear that processing times could be even shorter with more powerful systems, making it suitable for large-scale industrial use (Salman et al., 2025).

Table 6. List of the all packages used in the proposed Python GUI system.

Package / Library	Purpose in the System	Functionality in the Proposed GUI Program
datetime	Date and time management	Generates timestamps for reports and PDF invoices.
NumPy (numpy)	Numerical computation	Generates numerical datasets and performs array operations.
Pandas (pandas)	Data handling and processing	Organizes and manages industrial data in tabular form.
Tkinter (tkinter)	GUI development framework	Builds the graphical user interface (windows, inputs, buttons).
Tkinter Messagebox	GUI notifications	Displays alerts, warnings, and user messages.
Tkinter Filedialog	File handling interface	Allows users to choose locations for saving generated files.
Matplotlib (matplotlib.pyplot)	Data visualization	Generates graphs and plots for result visualization.
mpl_toolkits.mplot3d	3D plotting toolkit	Creates 3D visualization of demand, price, and production output.
Scikit-learn (sklearn)	Machine learning framework	Implements regression models for prediction.

Package / Library	Purpose in the System	Functionality in the Proposed GUI Program
sklearn.model_selection	Dataset splitting	Divides data into training and testing sets.
sklearn.linear_model	Regression modeling	Implements the MLR model.
sklearn.metrics	Model evaluation	Calculates performance metrics such as MSE and R^2.

The table in 6 shows the main software libraries that make up the Python GUI system. Together, these libraries let it create data, build machine learning models, make graphs, interact with users, and automatically report within the proposed framework for estimating industrial production costs.

6 Research Limitations and Prospective Directions

6.1 Research Limitations

The proposed Python-based GUI system that works with a MLR model shows good predictive power and is easy to use, but there are a few things that could be improved. The current framework focuses on three main industrial variables: demand (how much is needed each day), market price, and production output. These are enough to show that the proposed system works. But in real factories, there are often more operational parameters to think about, like the cost of raw materials, labor, energy use, machine use, and logistics. Adding these kinds of variables to future versions could make the predictive framework even more realistic and adaptable for use in more industries (J. Zhang et al., 2025).

The study uses linear regression modeling (Nguyen, 2024), which is easier to understand, faster to compute, and easier to add to decision-support tools that are already in use. Many people use linear models for production planning and economic forecasting because they are easy to understand and explain how variables are related (Shnawa & Jali, 2025) . But sometimes, industrial production systems show more complicated or nonlinear relationships between variables. Investigating supplementary modeling techniques in conjunction with the existing framework may yield additional insights for managing varied manufacturing datasets (Sun et al., 2023; Li et al., 2015) .

Another thing to think about is the method used to generate data for model testing and demonstration. The dataset used in the GUI program was made to clearly show how demand, market price, and production output are related. This made it possible to systematically test the algorithm and visualization parts. Structured datasets like these are good for checking that a system works, but having access to large-scale historical industrial datasets could help with more thorough empirical evaluation and comparisons with real manufacturing situations (Sun et al., 2023).

Also, the proposed system is currently a standalone decision-support application that is mainly meant to show how machine learning predictions, cost estimation formulas,

and graphical visualization can work together in a user-friendly way. In real-world industrial settings, similar analytical tools are often linked to enterprise management systems, production databases, or Industry 4.0 monitoring platforms. This lets data be shared in real time and decisions be made automatically (Lee et al., 2015).

In general, these factors do not make the proposed framework less useful; instead, they point out possible ways to make the system more useful and useful in real life. By adding more data, more industrial parameters, and more advanced analytical methods, future developments could make the proposed GUI-based production cost estimation platform even more useful in modern manufacturing settings (Warke et al., 2021).

6.2 Prospective Directions

Future studies can build on the proposed framework in a number of technical ways to make it more useful and reliable in real life. Adding more industrial variables to the predictive model is one promising way to make it better. These could be things like the cost of raw materials, the rate at which machines are used, the productivity of workers, the amount of energy used, and the schedules for maintenance. Adding these kinds of variables would help the model better understand how production works and make its predictions more accurate in spite of the limitations (Porter et al., 1981).

Another important area of research is the use of advanced deep learning and machine learning algorithms. ANN, Random Forest, Gradient Boosting, and Deep Learning architectures (Ning et al., 2020b) are examples of models that have been very good at modeling complicated nonlinear relationships in manufacturing datasets. When working with large amounts of industrial data, these methods might work better than traditional regression models (Bhattacharjee et al., 2024).

Another important future direction is to combine the proposed system with Industry 4.0 technologies. For instance, linking the GUI-based decision-support system to IoT sensors, real-time production monitoring systems, and cloud-based industrial platforms could make it possible to estimate production costs in real time and make decisions on the fly. This kind of integration would turn the system into a cyber-physical production planning tool that could work in smart manufacturing settings (Lee et al., 2015).

In addition, future research may look into explainable artificial intelligence (XAI) methods to make predictive models easier to understand and see through. Explainability tools can help industrial managers figure out which factors have the biggest impact on production costs and output, giving them useful information (Tzionis et al., 2025).

Lastly, the suggested GUI framework can be turned into a fully integrated industrial decision-support platform by adding features like supply chain optimization, demand forecasting, inventory management, and scheduling production for multiple products. These improvements would let the system handle more complex manufacturing planning tasks while still being easy to use (Alonge et al., 2021; Itu, 2025).

In general, looking into this research areas will make the proposed production cost estimation system more scalable, accurate, and useful for industry.

7 Conclusions

This study created a Python-based system for estimating industrial production costs that uses machine learning to help with efficient production planning and cost analysis. The suggested framework combines MLR, an interactive GUI, and tools for visualizing data to make it easier for users to make decisions in manufacturing settings. Three key industrial variables daily demand, market price, and production output were used to create a predictive relationship in the methodological design. This relationship makes it possible to automatically estimate production quantities and their costs. The dataset that was created was used to train and test the regression model. The GUI platform let users change production parameters, set profit margins, and do industrial calculations in real time. The system makes it even easier to understand by using three-dimensional graphics to show how demand, market price, and predicted production output are all related. The experimental results show that the proposed framework has very consistent predictive performance, with very low error values and almost perfect R^2 scores in a variety of industrial settings, including both rising and falling market prices. When we compared the MLR model to several other machine learning techniques, such as Ridge Regression, Support Vector Regression, Random Forest, and XGBoost, the system found that it performed the most consistently and was the easiest to understand for the structured dataset used in the system. The developed GUI has more than just predictive modeling. It also has useful features like automatically calculating production rates, unit production costs, and total industrial costs for multiple items. It can also create 3D graphical analysis and professional PDF invoices. These features show that the proposed system could be a useful tool for estimating industrial costs, planning production, and keeping an eye on operations. The study shows how combining machine learning, interactive GUI systems, and automated visualization can make complicated industrial calculations easier and give a structured way to support data-driven manufacturing decisions. The proposed framework thus enhances the increasing utilization of computational intelligence and software-driven decision-support systems in contemporary industrial management and production analysis.

8 Declarations

8.1 Funding and Conflicts of Interests

The authors declare that they have not received any funding for this work. They also confirm that there are no conflicts of interest or competing interests related to the publication of this manuscript.

References

Alonge, E. O., Eyo-Udo, N. L., Ubanadu, B. C., Daraojimba, A. I., Balogun, E. D., & Ogunsola, K. O. (2021). Real-Time Data Analytics for Enhancing Supply Chain Efficiency. *International Journal of Multidisciplinary Research and Growth Evaluation.*, *2*(1), 759–771. https://doi.org/10.54660/.IJMRGE.2021.2.1.759-771

Bertsimas, D., & Kallus, N. (2020). From Predictive to Prescriptive Analytics. *Management Science*, *66*(3), 1025–1044. https://doi.org/10.1287/mnsc.2018.3253

Bhattacharjee, B., Mou, S. N., MACHINE LEARNING FOR COST ESTIMATION AND FORECASTING IN BANKING: A COMPARATIVE ANALYSIS OF ALGORITHMS. *Frontline Marketing, Management and Economics Journal*, *04*(12), 66–83. https://doi.org/10.37547/marketing-fmmej-04-12-06

Breiman, L. (2001). Random Forests. *Machine Learning*, *45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

Chen, L., Tiang, S. S., Chong, K. S., Sharma, A., Berghout, T., & Lim, W. H. (2025). Predicting Construction Costs with Machine Learning: A Comparative Study on Ensemble and Linear Models. *Journal of Electronics, Electromedical Engineering, and Medical Informatics*, *7*(3), 624–645. https://doi.org/10.35882/jeeemi.v7i3.799

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. https://doi.org/10.1145/2939672.2939785

Cho, H. J., & Park, J. I. (2013). An Introduction of a Cost Estimation System for Model Changes. *Applied Mechanics and Materials*, *437*, 981–984. https://doi.org/10.4028/www.scientific.net/AMM.437.981

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297. https://doi.org/10.1007/BF00994018

Gray, J. B. (2002). Introduction to Linear Regression Analysis. *Technometrics*, *44*(2), 191–192. https://doi.org/10.1198/tech.2002.s714

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning*. Springer New York. https://doi.org/10.1007/978-0-387-84858-7

Hennebold, C., Klöpfer, K., Lettenbauer, P., & Huber, M. (2022). Machine Learning based Cost Prediction for Product Development in Mechanical Engineering. *Procedia CIRP*, *107*, 264–269. https://doi.org/10.1016/j.procir.2022.04.043

Itu, A. (2025). Industrial Scheduling in the Digital Era: Challenges, State-of-the-Art Methods, and Deep Learning Perspectives. *Applied Sciences*, *15*(19), 10823. https://doi.org/10.3390/app151910823

Kusiak, A. (2018). Smart manufacturing. *International Journal of Production Research*, *56*(1–2), 508–517. https://doi.org/10.1080/00207543.2017.1351644

Lee, J., Bagheri, B., & Kao, H.-A. (2015). A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems. *Manufacturing Letters*, *3*, 18–23. https://doi.org/10.1016/j.mfglet.2014.12.001

Li, Z., Cao, X., Ding, X., & Chen, H. (2015). *Prediction Model of Multiple Linear Regression Analysis in Grain Production:* 5th International Conference on Information Engineering for Mechanics and Materials. https://doi.org/10.2991/icimm-15.2015.233

Machfiroh, I. S., & Maulida, A. (2025). Multiple Regression Model on Selling Price, Sales Volume, Raw Material Costs, and Direct Labor Costs on Profit. *Jurnal Varian*, *8*(2), 221–232. https://doi.org/10.30812/mqvzph91

Mandolini, M., Manuguerra, L., Sartini, M., Lo Presti, G. M., & Pescatori, F. (2024). A cost modelling methodology based on machine learning for engineered-to-order products. *Engineering Applications of Artificial Intelligence*, *136*, 108957. https://doi.org/10.1016/j.engappai.2024.108957

Ma'ruf, A., Leuveano, R. A. C., & Utama, R. (2024). Product Design Cost Estimation for Make-to-Order Industry: A Machine Learning Approach. *Emerging Science Journal*, *8*(3), 1167–1183. https://doi.org/10.28991/ESJ-2024-08-03-022

Nguyen, T. A. (2024). *Using Multiple Linear Regression (MLR) to predict the real cost of a car model*. https://doi.org/10.58445/rars.1211

Ning, F., Qu, H., Shi, Y., Cai, M., & Xu, W. (2022). Feature-Based and Process-Based Manufacturing Cost Estimation. *Machines*, *10*(5), 319. https://doi.org/10.3390/machines10050319

Ning, F., Shi, Y., Cai, M., Xu, W., & Zhang, X. (2020a). Manufacturing cost estimation based on a deep-learning method. *Journal of Manufacturing Systems*, *54*, 186–195. https://doi.org/10.1016/j.jmsy.2019.12.005

Ning, F., Shi, Y., Cai, M., Xu, W., & Zhang, X. (2020b). Manufacturing cost estimation based on a deep-learning method. *Journal of Manufacturing Systems*, *54*, 186–195. https://doi.org/10.1016/j.jmsy.2019.12.005

Ning, F., Shi, Y., Cai, M., Xu, W., & Zhang, X. (2020c). Manufacturing cost estimation based on the machining process and deep-learning method. *Journal of Manufacturing Systems*, *56*, 11–22. https://doi.org/10.1016/j.jmsy.2020.04.011

Porter, A. L., Connolly, T., Heikes, R. G., & Park, C. Y. (1981). Misleading indicators: The limitations of multiple linear regression in formulation of policy recommendations. *Policy Sciences*, *13*(4), 397–418. https://doi.org/10.1007/BF00146959

Salman, Md. A., Mondal, K., Kobir, M. S., Khusi, Mst. K. K., Rifat, M., & Ghani, S. I. (2025). Harnessing big data analytics for industry 4.0: A hybrid architecture of revolutionary technologies and predictive implementation in manufacturing. *International Journal of System Assurance Engineering and Management*. https://doi.org/10.1007/s13198-025-03045-3

Shnawa, W. aziz, & Jali, Z. falah. (2025). MACHINE LEARNING TO MANAGING MANUFACTURING COSTS AND REDUCE COSTS- AN APPLIED STUDY IN THE GENERAL COMPANY FOR FOOD PRODUCTS. *World Economics and Finance Bulletin*, *50*, 70–84.

Soleimani, M., Naderian, H., Afshinfar, A. H., Savari, Z., Tizhari, M., & Agha Seyed Hosseini, S. R. (2023). A Method for Predicting Production Costs Based on Data Fusion from Multiple Sources for Industry 4.0: Trends and Applications of Machine Learning Methods. *Computational Intelligence and Neuroscience*, *2023*(1), 6271241. https://doi.org/10.1155/2023/6271241

Sun, Y., Wang, X., Zhang, C., & Zuo, M. (2023). Multiple Regression: Methodology and Applications. *Highlights in Science, Engineering and Technology*, *49*, 542–548. https://doi.org/10.54097/hset.v49i.8611

Tao, F., Qi, Q., Liu, A., & Kusiak, A. (2018). Data-driven smart manufacturing. *Journal of Manufacturing Systems*, *48*, 157–169. https://doi.org/10.1016/j.jmsy.2018.01.006

Tzionis, G., Mouratidis, P., Kougka, G., Gialampoukidis, I., Vrochidis, S., Kompatsiaris, I., & Vlachopoulou, M. (2025). A review of explainable AI methods and their application in manufacturing systems. *Discover Applied Sciences*, *8*(1), 52. https://doi.org/10.1007/s42452-025-07908-z

Warke, V., Kumar, S., Bongale, A., & Kotecha, K. (2021). Sustainable Development of Smart Manufacturing Driven by the Digital Twin Framework: A Statistical Analysis. *Sustainability*, *13*(18), 10139. https://doi.org/10.3390/su131810139

Więcek, D., Więcek, D., & Kuric, I. (2019). Cost Estimation Methods of Machine Elements at the Design Stage in Unit and Small Lot Production Conditions. *Management Systems in Production Engineering*, *27*(1), 12–17. https://doi.org/10.1515/mspe-2019-0002

Witten, I. H., Frank, E., & Hall, M. A. (2011). *Data Mining: Practical Machine Learning Tools and Techniques*. Elsevier. https://doi.org/10.1016/C2009-0-19715-5

Zhang, G., & Zhang, Q. (2025). Machine Learning-Based Construction Cost Prediction Model and Optimization Design. *International Journal of High Speed Electronics and Systems*, 2540774. https://doi.org/10.1142/S0129156425407740

Zhang, J., Yang, S., & Wang, Z. (2025). Research on Prediction Methods Based on Multiple Linear Regression—Artificial Neural Network Model. *Mathematical Modeling and Algorithm Application*, *5*(2), 76–80. https://doi.org/10.54097/jammp895

Zhang, Y., Ren, S., Liu, Y., & Si, S. (2017). A big data analytics architecture for cleaner manufacturing and maintenance processes of complex products. *Journal of Cleaner Production*, *142*, 626–641. https://doi.org/10.1016/j.jclepro.2016.07.123