An International Journal of Optimization and Control: Theories & Applications
ISSN: 2146-0957 eISSN: 2146-5703
Vol.14, No.4, pp.308-321 (2024)
https://doi.org/10.11121/ijocta.1462

RESEARCH ARTICLE

Early prediction of fabric quality using machine learning to reduce rework in manufacturing processes

Sema Aydın, Koray Altun*

Department of Industrial Engineering, Bursa Technical University, Bursa, Turkey
semaydinn.16@gmail.com, koray.altun@btu.edu.tr

ARTICLE INFO

Article history:
Received: 1 October 2023
Accepted: 11 September 2024
Available Online: 9 October 2024

Keywords:
Fabric quality
Rework reduction
Machine learning
Artificial neural networks

AMS Classification 2010:
97R40, 60G25

ABSTRACT

The increasing competition and rapid technological advancements in today's business world have raised customer expectations. People now expect quick delivery, low prices, and high-quality products. As a result, companies must adapt to this competitive environment to survive. Rework, which is a significant cost in production, increases expenses, reduces production efficiency, and can lead to customer attrition. Research shows various efforts across different sectors to reduce rework, although there is still a gap in the textile sector's fabric dyeing units. Common problems in these units include non-retentive colors, customer dissatisfaction with shades, and repeated dyeing due to environmental factors or dye vat issues. This study uses logistic regression and artificial neural networks models from machine learning to predict which fabrics will need rework, using data from a textile company in Bursa. The analysis indicates that artificial neural networks models perform better.

(cc) BY

1. Introduction

In the textile industry, optimizing fabric dyeing processes is a pivotal challenge. Rework processes within fabric dyeing units are among the most critical factors contributing to cost escalation, low-quality production, and customer dissatisfaction.

“Rework” can be defined as the need for additional processing or corrective measures due to various quality issues. Rework represents a form of waste, driving research into the concept of Zero Defect Manufacturing [1,2]. This concept aims to eliminate defects before they necessitate rework. However, achieving this goal requires proactive measures to anticipate and prevent potential quality issues before they escalate into rework processes.

Machine learning techniques offer promising approaches to identifying patterns in historical data and predicting defects in manufacturing processes. Although there is growing interest in using machine learning for quality assurance and defect detection in manufacturing, a significant gap remains in research on applying these techniques specifically for "rework prediction" in the textile industry.

Correspondingly, this research aims to address this gap by evaluating the effectiveness of logistic regression and artificial neural networks (ANNs) in predicting rework instances in fabric dyeing processes.

In line with the study's objectives and the research landscape, the following contributions are emphasized:

- This study addresses a critical research gap in the textile industry by exploring the application of machine learning for predicting errors in fabric dyeing processes. While machine learning has been extensively applied in various industries, its use in the textile sector for rework prediction remains relatively underexplored.
- This study focuses on applying machine learning, specifically logistic regression and ANNs, to develop models that predict rework in fabric dyeing units within the textile industry. By leveraging data-driven methods, our goal is to enhance the early detection of potential quality issues and reduce the need for rework before it escalates a bigger problem.

The aim is to proactively predict and mitigate potential quality issues, thereby optimizing production processes and minimizing instances of rework. This study not only contributes to the expanding field of predictive

*Corresponding author

analytics and manufacturing optimization but also seeks to advance the application of machine learning within the textile sector.

2. Literature review

A review of the literature on early quality prediction and rework reveals numerous studies across various industries. However, the application of these techniques specifically within the textile sector appears to be less common.

A systematic effort was undertaken to conduct a comprehensive literature review. Prominent databases such as Science Direct, Web of Science, and Google Scholar were utilized to identify relevant materials. Two specific search strings were employed to reflect the core focus of the investigation. The first search string, "quality" AND "defect detection" AND "prediction" AND "manufacturing," aimed to cover the scholarly work on predictive methods for quality assurance and defect detection in manufacturing contexts. The second search string, "rework" AND "defect" AND "machine learning," was designed to explore research on the application of machine learning approaches to predict defects and prevent rework processes. By systematically using these search strings and examining the results from the selected databases, the literature review aimed to extract and synthesize relevant insights from the existing body of scholarly work.

A case study [1] was conducted within an automotive company, employing an Early Quality Prediction system grounded in a data-driven approach. In this study, the focus was on applying Convolutional Neural Network (CNN) techniques to time-series data to proactively predict and prevent defects. This approach aimed to minimize rework costs and optimize product quality through predictive insights. A data-driven mathematical model has been developed [3] for a dynamic manufacturing process with multiple rework lines, focusing on calculating production rates and machine efficiency for each machine. This model is designed to address scenarios involving preventive maintenance activities for each machine, with machine efficiencies computed based on real performance data for analysis. Taking into consideration the dynamic states of the production system, including rework productions, a mathematical model has been developed [4] to calculate efficiency of the process. With this model, rework strategies are predicted to ensure the production of products with the desired quality. In the context of assembly lines, rework stations are typically set up at the end of assembly lines for reprocessing faulty products. These rework stations operate within standard production processes when error rates are low. In cases where rework stations operate in dynamic conditions, a nonlinear mixed-integer programming model [5] has been proposed to enhance station efficiency. This model aims to increase the efficiency of rework stations in dynamic situations, thus improving the overall performance of the assembly line. In the automotive industry, various Machine Learning methods have been employed to predict errors in assembly lines. The results obtained from different methods have been compared [6]. To facilitate this comparison, specific metrics were established, and these metrics were contrasted among the six algorithms employed.

To provide a comprehensive overview of the current state of research on early quality prediction and rework processes, a detailed literature summary table (Table-1) has been compiled. This table highlights the key methodologies, applications, and findings from various studies, illustrating the broader landscape of machine learning applications in different sectors.

A review of the literature on the application of artificial neural networks in the textile sector reveals that they have been employed in various studies to predict yarn parameters, optimize weaving processes, enhance finishing stages, and assess fabric comfort parameters [7]. Artificial Neural Networks (ANN) has also been employed to detect fabric defects in weaving [8]. To predict the characteristics of woven fabric (width, weight, weft and warp tensile values), an advanced feedforward recurrent artificial neural network (ANN) model [9] was employed and compared with a linear regression model. Artificial Neural Network (ANN) methods have been employed [10] to predict weft errors that emerge during fabric production in a textile company. ANN methods have been employed [11] to predict the impact of chemical finishing processes on the CIELab value of the fabric color.

This study aims to develop alert systems for the identification of products with a high probability of rework and the implementation of specific actions for their mitigation. Early Quality Prediction systems employ deep learning methods as well as various techniques for this purpose. However, within the textile industry, specifically in a textile company with a fabric dyeing production process, there is a lack of research on using Machine Learning for the prediction of errors. This particular gap forms the distinct aspect of this study.

The remainder of this paper is organized as follows: Section 3 presents the problem statement, detailing the specific challenges addressed in this study. Section 4 provides the methodology. Section 5 discusses the results. Section 6 explores managerial insights and practical implementations, offering actionable recommendations based on the study's outcomes. Finally, Section 7 concludes the paper with a summary of key findings and an outlook on future research directions.

Table 1. Summary table of the literature.

Ref.	Methods used	Application area	Key findings
[1]	Deep learning, Time series data	Manufacturing	Successful use of deep learning techniques for early quality prediction.
[2]	Various machine learning methods	Zero defect manufacturing	Comprehensive review of current methods to achieve zero defect manufacturing.
[3]	Data-driven modeling and analysis	Multi-stage manufacturing systems	Used data-driven modeling to analyze quality rework cycles.
[4]	Product traceability and rework analysis	Manufacturing systems	Analyzed quality performance considering product traceability and rework.
[5]	Mixed-integer programming	Assembly lines	Developed a model for positioning rework stations to improve efficiency.
[6]	Machine learning	Assembly environment	Applied machine learning for error detection in low-automation assembly environments.
[7]	Artificial neural networks	Textile industry	Reviewed applications of artificial neural networks in the textile industry.
[8]	Artificial neural networks	Weaving technology	Comprehensive review of ANN applications in weaving technology.
[9]	Artificial neural networks	Woven fabric	Used advanced feedforward recurrent neural networks to predict woven fabric properties.
[10]	Artificial neural networks, Multiple linear regression	Fabric defects	Compared ANN and multiple linear regression models for predicting fabric defects.
[11]	Artificial neural networks	Chemical finishing processes	Predicted the impact of chemical finishing on fabric color using ANNs.
[12]	Robust mathematical model	Epidemic modeling	Developed a mathematical model to predict the course of the COVID-19 epidemic.
[13]	Machine learning	Agri-food production forecasting	Used robust and resilient machine learning methods to forecast agri-food production.
[14]	Machine learning, Deep learning	Healthcare	Applied machine learning techniques to improve early disease detection.
[15]	Support vector machines, Neural networks	Manufacturing	Compared the efficiency of SVM and ANN in predicting manufacturing defects.
[16]	Deep learning	Electronics manufacturing	Used deep learning to predict defects in electronics manufacturing.
[17]	Logistic regression, Decision trees	Automotive	Developed predictive models to reduce rework in automotive manufacturing.
[18]	Ensemble learning	Aerospace	Applied ensemble learning techniques for quality prediction in aerospace components.
[19]	Neural networks, Fuzzy logic	Food processing	Combined neural networks and fuzzy logic for defect prediction in food processing.
[20]	Gradient boosting machines	Pharmaceutical	Used gradient boosting machines to predict quality issues in pharmaceutical manufacturing.
[21]	Intelligent quality control system	Surface roughness	Enhanced surface quality control.
[22]	Artificial intelligence techniques	Production cycle	Rework control optimization.
[23]	Machine learning for quality prediction	Injection molding process	Prediction of defects in injection molding.
[24]	Real-time quality prediction	Serial-parallel manufacturing processes	Real-time quality identification and prediction.
[25]	Integration of multisource information	Manufacturing processes	Enhanced quality prediction using multisource information.
[26]	Soft computing techniques	Machining process	Intelligent quality prediction in machining.
[27]	BP neural network algorithm	Supply chain quality prediction	Optimized method for supply chain quality prediction

3. Problem statement

Fabric dyeing involves several key steps: preparing the fabric, mixing the dyes, dyeing the fabric, washing and rinsing, and drying. First, the fabric is cleaned and treated to ensure it absorbs the dye evenly. Next, the dyes are mixed to achieve the desired color. The fabric is then immersed in the dye mixture, ensuring it is thoroughly soaked. After dyeing, the fabric is washed and rinsed to remove any excess dye, and finally, it is dried.

Fabric dyeing involves several methods, each suited to different types of fabrics and production scales. Common methods include batch dyeing, where fabric is dyed in a single batch, and continuous dyeing, which is efficient for large quantities as fabric moves continuously through the dyeing process. Pad-dyeing uses rollers to ensure even dye absorption, while jet dyeing employs high-pressure jets for delicate fabrics. Beam dyeing immerses fabric wound on a beam into the dye bath, and tie-dyeing creates unique patterns by tying fabric before dyeing. Solution dyeing integrates dye into the polymer solution for synthetic fibers, and dip dyeing achieves gradient effects by submerging

fabric to different levels. Each method offers distinct advantages depending on the desired outcome and fabric type.

Rework processes are often needed in fabric dyeing due to various issues such as uneven dye distribution, incorrect color shades, or dye spots. These problems can arise from improper preparation, inaccurate dye mixing, or inconsistencies in the dyeing process. Reworking involves correcting these defects to meet the required quality standards, ensuring the final product is uniform and meets customer expectations.

The research was conducted within a textile company located in Bursa with a Fabric Dyeing Department. It was found that dyed fabrics frequently required re-dyeing or additional finishing processes due to various quality issues. These corrective processes are referred to as "Rework" within the operation. To address this issue, historical fabric dyeing job order data from the MRP software, which the company uses, was utilized. A dataset comprising 4,855 entries from the past year was collected for this purpose. The goal is to predict and minimize these rework instances to enhance efficiency and quality in the fabric dyeing process.

4. Methodology

As a result of the literature review, the decision of whether a fabric should undergo dyeing was framed as a "Classification" problem. To address this, an algorithm was developed using Logistic Regression and Artificial Neural Networks to determine whether rework is needed or not. Among the Artificial Neural Networks algorithms, the Multilayer Perceptron (MLP) algorithm was selected, specifically the MLPClassifier from the sklearn.neural_network library in Python. The parameters chosen during the construction of the Artificial Neural Networks model directly impact its accuracy. The steps involved in the study are illustrated in Figure 1.

Logistic regression

Regression is a statistical method used to determine the relationship between two variables, where one is dependent (y) and the other is independent (x). In this relationship, y is expressed as a function of x. Given the x attribute values, the continuous variable y is calculated. Regression is a supervised learning technique. Regression analysis helps identify the cause-and-effect relationship between variables.

Logistic regression is a statistical method used to predict binary outcomes. It predicts the probability of a result that can have only two values. The prediction is based on the use of one or several predictors (numerical and categorical). Linear regression is not suitable for values that can be expressed in a binary system such as yes/no or presence/absence because it can predict values outside the range of 0 and 1. Logistic regression produces a logistic curve that is limited to values between 0 and 1.

The logistic regression model aims to minimize the cost function by updating the parameters and learning the parameters that provide the best classification results [28]. In logistic regression analysis, the ratio of the probability of an event occurring to the probability of it not occurring is called the odds ratio, and the probability of not occurring is calculated as follows:

$$1 - P_i = 1 - 1/(1 + e^{-Z_i}) = (1 + e^{-Z_i} - 1)/(1 + e^{-Z_i})$$
$$= e^{-Z_i}/(1 + e^{-Z_i})$$

The odds ratio is obtained by dividing the probability of occurrence by the probability of non-occurrence. The explicit expression of the odds ratio is:

$$P_i/(1 - P_i) = 1/(1 + e^{-Z_i}) * ((1 + e^{-Z_i})/(e^{-Z_i}) = e^{Z_i}$$

By taking the natural logarithm of both sides of the logistic function, which becomes usable in linear regression analysis, a linear structure is obtained:

$$g(x) = \ln(P_i/(1 - P_i)) = \ln e^{Z_i} = Z_i = b_1 + b_2 X_i$$
$$P_i/(1 - P_i) = 1/(1 + e^{-Z_i}) * ((1 + e^{-Z_i})/(e^{-Z_i}) = e^{Z_i}$$

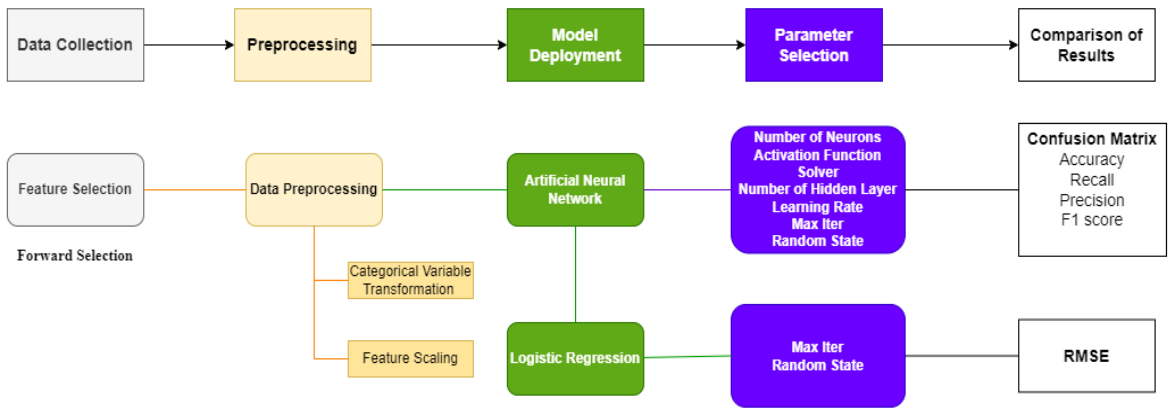

Figure 1. An overview of the methodology.

By taking the natural logarithm of both sides of the logistic function, which becomes usable in linear regression analysis, a linear structure is obtained:

$$g(x) = \ln (P_i /(1 - P_i)) = \ln e^{Z_i} = Z_i = b_1 + b_2 X_i$$

Artificial Neural Networks (ANNs)

ANNs are computer systems designed to automatically perform tasks such as generating and discovering new knowledge through learning. ANNs can be used for tasks like prediction, classification, data association, data interpretation, and data filtering. ANNs are nonlinear information and data processing systems. They consist of processing units called neurons and the connections between these neurons.

The three main components of ANNs are neurons, the connections between these neurons, and functions. ANNs are made up of layers; the input layer is where data enters the network, and hidden layers process the data received in the input layer. There can be multiple hidden layers. The output layer is where the processed data is expressed as output. When a structure has multiple hidden layers, deep neural networks are used [29]. Figure 2 shows the working principle of the ANN model.

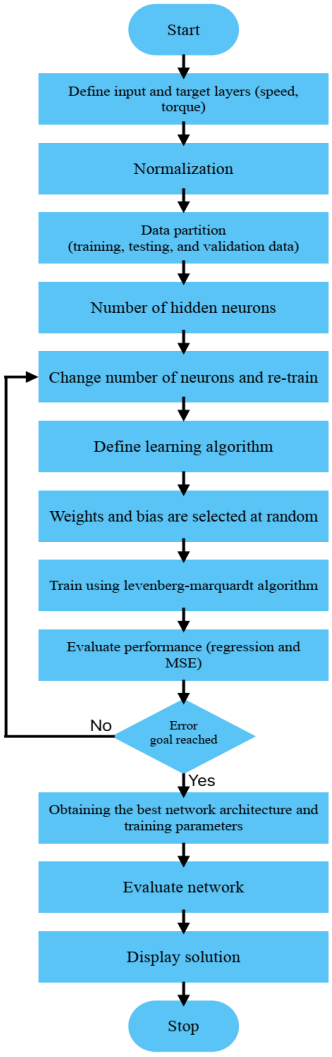

Figure 2. A typical ANN modeling workflow.

An MLP model has three layers: an input layer, hidden layers, and an output layer. Each layer can have one or more neurons, and all neurons in one layer can influence all neurons in the next layer. This relationship can be expressed as:

$$y_k = f (\sum_i w_{ki} \cdot x_i)$$

Where:

y_k is the output value of neuron k

w_{ki} is the weight value between input x_i and output neuron k

x_i is the input value i

In ANN, various transfer functions such as linear, log-sigmoid, and tan-sigmoid can be used to convert input values to output values. In our study, we selected the ReLU (Rectified Linear Unit) and logistic transfer functions to transform the weighted input values into output values.

ReLU activation function

$$f(x) = \max(0,x)$$

This function returns 0 if x is less than 0, otherwise, it returns x.

Logistic activation function

$$f(x) = 1 / 1 + e^{-x}$$

Here, e is the base of the natural logarithm, and x is the input value. This function converts the input value into an output between 0 and 1, allowing the output to be interpreted as a probability.

In this implementation, the behaviors of the parameters listed below have been examined, and based on the performance of the proposed model, suitable values for these parameters have been selected. Some of these parameters were considered together, leading to the execution of 61 different experiments. The findings of these experiments are detailed in the Results section (in Table 3).

The parameters studied

Batch size and iteration count (max_iter), Learning rate (learning_rate), Weight initialization (random_state), Neuron activation function and solver, Neuron count in the hidden layer.

When constructing the artificial neural networks model, there is no fixed formula for determining the number of hidden layers, neuron count, or activation function. Instead, a trial-and-error approach using test data is employed to determine the values that yield the best results for the model.

Test models have been created using the Relu and Logistic activation functions. Additionally, models have been built using the Adam and lbfgs optimizers. It is important to avoid setting a high learning rate, as it may lead to overfitting in the model. Therefore, through necessary testing, a learning rate of 0.001 has been selected for the model.

Generally, lbfgs tends to give optimal results on smaller datasets, while Adam is more suitable for larger datasets [30].

ReLU is a commonly used activation function, especially in deep neural networks. When used with lbfgs, ReLU often performs well because its derivative is non-zero, which can speed up training and reduce the vanishing gradient problem [31].

The logistic activation function is commonly used for binary classification problems. Using lbfgs with the logistic activation function is a good choice for small or medium-sized datasets. However, for deep neural networks or large datasets, activation functions like ReLU might be preferred due to their better performance [32, 33].

One of the important factors affecting the success of artificial neural networks is the selection of the number of neurons in each layer. There is no precise mathematical formula to determine this number, and it is often found through trial and error. Various factors should be considered when determining the number of neurons in a layer. Increasing or decreasing the number of neurons in an artificial neural network can affect the model's performance, error rate, and generalization ability. For instance, using too many neurons can lead to overfitting, where the model fits the training data too closely and fails to generalize well to real-world data [34]. Conversely, using too few neurons may cause the model to underfit, failing to capture the complexity of the dataset, which can decrease performance and increase computational costs [35]. The number of hidden layers and the number of neurons per layer have been determined through trial and error.

The Long Short Term Memory (LSTM) method, a type of deep learning technique, has been employed for real-time defect detection and texture classification on fabrics. This method aims to identify defects on fabrics using digital images.

Through a comprehensive review of literature and expert opinions, a set of 13 candidate features has been identified. The "IsTamir" outcome data serves as the target variable. Among the candidate attributes, a feature selection process has been conducted.

The Forward Selection method was employed for feature selection. This approach was chosen due to its remarkable success in classification tasks. The model commences with the most crucial variable concerning the dependent variable. Initially, the model incorporates solely one variable. Subsequent variables are incrementally added to the model. If the inclusion of a variable enhances the model's performance, it is retained within the model. Employing this process, all variables are examined, ultimately shaping the final model. List of the features are presented in Table 1.

Normalization is a technique commonly applied as part of data preprocessing in machine learning. Its purpose is to transform the values of numerical columns in a dataset onto a common scale without distorting the inherent differences in value ranges. Not every dataset requires normalization for machine learning purposes. The normalization process can be used to reduce data dimensionality, perform operations at appropriately scaled intervals with normalized values, and attain more meaningful and interpretable results.

In literature, various forms of data normalization exist. These include but are not limited to methods like minimum-maximum (min-max) scaling, decimal scaling, z-score normalization, and sigmoid normalization [36].

Table 2. Identified features.

Rework (Response variable)
Fabric quality name (Categorical)
Raw fabric pattern name (Categorical)
Color code (Categorical)
Finished fabric formation type code (Categorical)
Planned length (Numerical)
Raw fabric code (Categorical)
Master recipe name (Categorical)
Process name (Categorical)
Machine name (Categorical)
Operation flow code (Categorical)

Scaling is used to address the magnitude differences between various features in a dataset. When features have different scales, some may exert a stronger influence than others. This imbalance in feature scales can distort their equal contribution and potentially hinder algorithm performance.

The scaling process is implemented to compress feature values within a specific range. In the case of Logistic Regression and Artificial Neural Networks models, scaling has been deemed necessary. The StandardScaler scaling method from the Python sklearn.preprocessing library has been utilized to scale the features.

There are two crucial steps in prediction: first is the preparation of the data for prediction, and second is the comparison of different predictive models. The criteria for comparing models include; accuracy, speed, robustness, scalability, and interpretability.

Fundamental performance indicators employed in assessing the performance of Artificial Neural Networks and machine learning methods include R2, MSE, RMSE, and MAE [37].

To compare the models, Accuracy and F1 Score metrics were used.

Accuracy: The ratio of correctly predicted instances to the total instances.

$$\text{Accuracy} = (TP + TN) / (TP + TN + FP + FN)$$

TP:True Positive

TN:True Negative

FP:False Positive

FN:False Negative

Precision: Indicates how many of the instances predicted as positive are actually positive. It measures the model's ability to correctly classify the positive class.

$$\text{Precision} = TP / (TP + FP)$$

Recall: Indicates how many of the actual positive instances are correctly classified. It is the ratio of correctly predicted positive instances to all actual positive instances.

Recall = $TP / (TP + FN)$

F1 Score: The harmonic mean of Precision and Recall.

F1 Score=$2*((Precision*Recall) / (Precision+ Recall))$

Visualization of Class Distribution: To visualize the distribution of the dataset (Figure 3) in Python, the code block was used:

```
import matplotlib.pyplot as plt
import numpy as np

# Visualize class distribution
plt.bar(np.unique(y), np.bincount(y))
plt.xlabel('Class')
plt.ylabel('Number of Samples')
plt.title('Class Distribution')
plt.show()
```

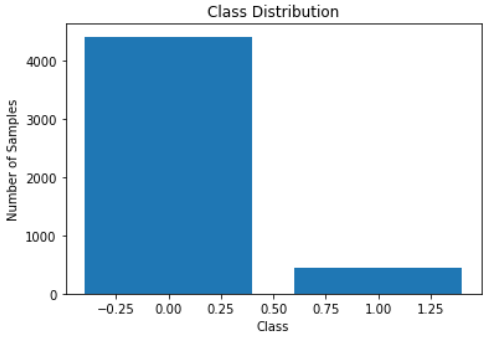

Figure 3. Class distribution.

Given that the dataset is imbalanced, the F1 Score is a more appropriate performance metric to use, as it provides a better measure of the model's performance on datasets with uneven class distributions.

5. Results

In this study, Python programming is used to implement machine learning algorithms. The process begins with handling missing and outlier data. Confirmation from the LEO MRP program ensures that no missing data is anticipated for the relevant attributes.

The get_dummy function is used for categorical data transformation. The get_dummies function takes each category of a categorical variable as a separate column and assigns a value of 1 to the rows corresponding to that category, and 0 to other rows. This way, each category becomes a distinct feature that can be utilized by machine learning models.

Logistic regression implementation

The logistic regression model was implemented using Python, with the following steps:

Data Loading and Preprocessing: The data was loaded from a CSV file and categorical variables were converted into dummy variables. The dependent variable (IsTamir) was encoded using label encoding.

Feature Engineering: The independent variables were selected and the dataset was split into training and testing sets. The data was scaled using StandardScaler.

Model Training and Prediction: A logistic regression model was trained and used to make predictions on the test set.

Evaluation Metrics: Accuracy and F1 Score metrics were used to evaluate the model's performance. Additionally, a confusion matrix and root mean squared error (RMSE) were computed.

The complete Python code is given in Appendix.

The accuracy of the logistic regression model is 92.64%. This means that the model correctly predicts the class of 92.64% of the samples in the test dataset. While accuracy is a commonly used metric, it may not be sufficient for evaluating the performance of models on imbalanced datasets.

The F1 score, which considers both precision and recall, is 0.4779. The F1 score is a better metric for imbalanced datasets because it takes into account false positives and false negatives. A higher F1 score indicates better overall performance of the model.

The confusion matrix provides insight into the model's performance across different classes. It reveals that the model correctly predicted 1431 samples as true positives and 54 samples as true negatives. However, it misclassified 36 samples as false positives and 82 samples as false negatives.

The RMSE value is 0.2713, reflecting the average difference between the actual and predicted values. Lower RMSE values indicate better model performance. However, in the context of classification, RMSE might not be the most informative metric.

While the accuracy of the logistic regression model is relatively high, the F1 score and confusion matrix reveal that the model's performance may be impacted by the imbalanced nature of the dataset. It's important to consider these results in the context of the dataset characteristics and the specific goals of the classification task.

ANNs implementation

The Python code (in Appendix) implements an Artificial Neural Network (ANN) model, starting from data preprocessing steps and extending to training the model and evaluating its performance.

The values obtained from the results based on the parameters selected during the creation of the Artificial Neural Networks models are presented in Table 3 (in appendix).

Table 3 presents the evaluation results of 61 different models based on various configurations, including the activation function, solver, number of hidden layers, and number of neurons.

Upon examining the results, it's evident that there is a wide range of performance across different configurations. Some models achieve high accuracy and F1 Score values, indicating robust classification

performance, while others exhibit lower values, suggesting potential areas for improvement.

These results provide valuable insights into the effectiveness of different configurations in training artificial neural network models for the given task. Further analysis and experimentation may help identify optimal configurations for maximizing classification performance.

Comparison of Logistic Regression and Artificial Neural Network Results

Logistic Regression and Artificial Neural Networks methods were analyzed in Python. The results were compared using the Accuracy metric. Accuracy values for multiple Artificial Neural Network models were calculated and are presented in Table 3. The accuracy and F1 Score obtained from the Logistic Regression model were compared with those from the Artificial Neural Network model with the highest accuracy.

For Logistic Regression, the Accuracy is calculated at 0.90, while for the Artificial Neural Networks (Model 1), the Accuracy is found to be 0.92. This indicates that the Artificial Neural Networks model achieves higher accuracy compared to the Logistic Regression model.

Similarly, the F1 Score performance metric was evaluated. For Logistic Regression, the F1 Score is 0.48, while for the Artificial Neural Networks (Model 1), it is 0.47. Although there is only a slight difference in the F1 Score between the two methods, the Artificial Neural Networks model has a small advantage over Logistic Regression.

In conclusion, the Artificial Neural Networks (Model 1) method achieves higher accuracy compared to Logistic Regression, while the F1 Score values are similar. These results suggest that, for this specific classification problem, the Artificial Neural Networks method may be more effective.

6. Managerial insights and practical implications

The findings of this study can inform strategic planning initiatives aimed at reducing rework costs in fabric dyeing processes. By leveraging predictive analytics, textile companies can proactively identify potential rework needs at the planning stage, enabling preemptive measures to optimize production processes and reduce rework expenses.

Implementing AI-based algorithms for predicting rework needs allows textile companies to optimize operational efficiency by streamlining production processes and minimizing downtime associated with rework activities. This, in turn, enhances overall productivity and cost-effectiveness.

Insights derived from predictive models can facilitate informed resource allocation and risk management strategies. By identifying high-risk production batches or processes prone to rework, companies can allocate resources more efficiently and implement targeted interventions to mitigate risks and minimize rework occurrences.

Table 3. Different configurations and obtained results.

Model No	Activation function	Solver	Number of hidden layers	Number of neurons	Accuracy	F1 Score
1	Relu	Adam	2	3,2	0.92	0.39
2	Relu	Adam	2	3,3	0.92	0.24
3	Relu	Adam	2	3,6	0.91	0.45
4	Relu	Adam	2	6,6	0.91	0.38
5	Relu	Adam	1	6	0.91	0.36
6	Relu	Adam	2	6,5	0.91	0.33
7	Relu	Adam	2	3,4	0.91	0.26
8	Relu	Adam	2	4,7	0.91	0.25
9	Relu	Adam	2	3,5	0.91	0.07
10	Logistic	Adam	2	7,7	0.9	0.47
11	Logistic	Adam	2	6,6	0.9	0.45
12	Logistic	Adam	2	8,8	0.9	0.45
13	Logistic	lbfgs	1	2	0.9	0.45
14	Logistic	Adam	2	5,5	0.9	0.44
15	Logistic	lbfgs	1	7	0.9	0.44
16	Relu	lbfgs	1	4	0.9	0.44
17	Relu	Adam	1	3	0.9	0.43
18	Logistic	Adam	2	7,8	0.9	0.43
19	Relu	lbfgs	2	4,4	0.9	0.42
20	Relu	Adam	4	4	0.9	0.4
21	Relu	Adam	2	4,6	0.9	0.37
22	Logistic	lbfgs	2	5,6	0.9	0.37
23	Relu	Adam	2	4,4	0.9	0.34

Table 3. Different configurations and obtained results. (continued)

Model No	Activation function	Solver	Number of hidden layers	Number of neurons	Accuracy	F1 Score
24	Relu	Adam	2	5,6	0.9	0.34
25	Logistic	lbfgs	2	4,5	0.9	0.33
26	Relu	Adam	2	4,3	0.9	0.3
27	Logistic	lbfgs	1	5	0.89	0.47
28	Logistic	Adam	3	8,8,8	0.89	0.46
29	Logistic	Adam	2	4,4	0.89	0.44
30	Logistic	Adam	1	6	0.89	0.44
31	Logistic	Adam	1	9	0.89	0.44
32	Relu	Adam	2	2,3	0.89	0.43
33	Logistic	Adam	2	8,9	0.89	0.43
34	Logistic	Adam	2	7,5	0.89	0.42
35	Logistic	lbfgs	2	5,5	0.89	0.39
36	Logistic	Adam	2	8,7	0.88	0.44
37	Logistic	lbfgs	1	9	0.87	0.42
38	Logistic	Adam	2	6,7	0.87	0.41
39	Logistic	lbfgs	2	7,7	0.87	0.37
40	Logistic	lbfgs	2	6,7	0.86	0.34
41	Logistic	lbfgs	2	2,2	0.86	0.31
42	Logistic	lbfgs	1	6	0.85	0.43
43	Logistic	lbfgs	1	11	0.85	0.42
44	Logistic	lbfgs	1	10	0.85	0.41
45	Logistic	lbfgs	1	8	0.85	0.4
46	Relu	Adam	2	6,7	0.85	0.39
47	Relu	Adam	2	6,7	0.85	0.39
48	Relu	lbfgs	2	4,3	0.85	0.39
49	Logistic	Adam	1	1	0.85	0.37
50	Relu	Adam	2	3,7	0.85	0.25
51	Logistic	lbfgs	1	12	0.84	0.4
52	Relu	Adam	2	5,5	0.84	0.38
53	Logistic	lbfgs	1	4	0.84	0.36
54	Relu	lbfgs	1	3	0.83	0.4
55	Relu	lbfgs	2	3,3	0.82	0.37
56	Relu	lbfgs	1	5	0.82	0.37
57	Relu	lbfgs	2	3,3	0.82	0.37
58	Relu	lbfgs	1	2	0.8	0.38
59	Logistic	lbfgs	1	3	0.79	0.33
60	Relu	lbfgs	2	6,6	0.77	0.35
61	Relu	Adam	2	7,7	0.73	0.25

7. Conclusions and outlook

The primary objective of this study was to develop an algorithm capable of predicting rework needs at the planning stage for textile companies with fabric dyeing processes, with the goal of minimizing rework costs due to faulty productions. Machine Learning methods, including Logistic Regression and Artificial Neural Networks, were employed to tackle this issue as a classification problem. Both Logistic Regression and Artificial Neural Networks achieved successful outcomes.

Future studies could develop a more effective method for textile companies to predict rework needs. For instance, an algorithm could be designed to create alternative production routes before engaging in re-dyeing or repair processes. Additionally, developing a model that suggests the use of different chemical compositions could significantly reduce rework costs.

In conclusion, this study takes a pivotal step towards solving a significant issue in the textile sector by providing a potential solution for predicting the need for reprocessing in fabric dyeing processes using AI-based algorithms. Future efforts could further enhance this algorithm, ultimately optimizing production processes for textile companies in terms of efficiency and cost-effectiveness.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments. This paper is an outcome derived from the master's thesis of the first author.

References

[1] Saadallah A, Abdulaaty O, Büscher J, Panusch T, Morik K, Deuse J. (2022). Early quality prediction using deep learning on time series sensor data. *Procedia CIRP*, 107, 611-616.

[2] Caiazzo B, Di Nardo M, Murino T, Petrillo A, Piccirillo G, Santini S. (2022). Towards zero defect manufacturing paradigm: A review of the state-of-the-art methods and open challenges. *Computers in Industry*, 134, 103548.

[3] Zhu C, Chang Q, Arinez J. (2020). Data-enabled modelling and analysis of multistage manufacturing systems with quality reqork loops. *Journal of Manufacturing Systems*, 56, 573-584.

[4] Colledani M, Angius A. (2020). Production quality performance of manufacturing systems with in-line product traceability and rework. *CIRP Annals*, 69, 365-368.

[5] Cavdur F, Kaymaz E, Sebatli A. (2018). A mixed-integer programming model for optimizing rework station position in assembly line balancing. *Uludag University Journal of The Faculty of Engineering*, 23 (3), 273-287.

[6] Schuh G, Gützlaff A, Thomas K, Welsing M. (2021). Machine learning based defect detection in a low automated assembly environment. *Procedia CIRP*, 104, 265-270.

[7] Corekcioglu M, Ercan E, Aras Elibüyük S. (2021). Usage applications of artificial neural network methods in textile industry. *Journal of Technical Sciences*, 11(2), 14-20.

[8] Ozdemir H. (2013). Artificial neural networks and their usage in weaving technology. *Electronic Journal of Vehicle Technologies*, 7 (1), 51-68.

[9] Turker E. (2017). A research on estimation of the weave fabric properties with the artificial neural networks. *Textile and Apparel*, 27(1), 10-21.

[10] Arikan Kargi S. (2014). A comparison of artificial neural networks and multiple linear regression models as in predictors of fabric weft defects. *Textile and Apparel*, 24 (3), 309-316.

[11] Balci O, Ogulata RT. (2009). Prediction of CIELab values and color changing occurred after chemical finishing applications by artificial neural networks on dyed fabrics. *Textile and Apparel*, 19(1), 61-69.

[12] Lotfi, R., Kheiri, K., Sadeghi, A., & Babaee Tirkolaee, E. (2022). An extended robust mathematical model to project the course of COVID-19 epidemic in Iran. *Annals of Operations Research*, 1-25.

[13] Lotfi, R., Gholamrezaei, A., Kadlubek, M., Afshar, M., Ali, S. S., & Kheiri, K. (2022). A robust and resilience machine learning for forecasting agri-food production. *Scientific Reports*, 12 (1), 21787.

[14] Sadeghi, A., Kheiri, K., Lotfi, R., & Babaee Tirkolaee, E. (2023). Machine Learning and Deep Learning Techniques for Early Disease Detection. *Journal of Medical Systems*, 47(1), 25-36.

[15] Kumar, P., Mehta, M., & Singh, R. (2019). Comparative Analysis of Support Vector Machines and Neural Networks for Manufacturing Defect Prediction. *International Journal of Advanced Manufacturing Technology*, 104, 1589-1599.

[16] Wang, L., Li, J., & Zhou, Y. (2020). Application of Deep Learning in Predicting Defects in Electronics Manufacturing. *IEEE Transactions on Industrial Informatics*, 16(5), 3178-3185.

[17] Patel, S., & Gupta, R. (2018). Predictive Models for Reducing Rework in Automotive Manufacturing Using Logistic Regression and Decision Trees. *Journal of Manufacturing Science and Engineering*, 140(8), 081015.

[18] Lu, F., Zhou, G., Liu, Y., & Zhang, C. (2022). Ensemble transfer learning for cutting energy consumption prediction of aviation parts towards green manufacturing. *Journal of Cleaner Production*, *331*, 129920.

[19] Deisingh, A. K., Stone, D. C., & Thompson, M. (2004). Applications of electronic noses and tongues in food analysis. *International journal of food science & technology*, *39*(6), 587-604.

[20] Serna-Carrizales, J. C., Zárate-Guzmán, A. I., Flores-Ramírez, R., de León-Martínez, L. D., Aguilar-Aguilar, A., Warren-Vega, W. M., ... & Ocampo-Pérez, R. (2024). Application of artificial intelligence for the optimization of advanced oxidation processes to improve the water quality polluted with pharmaceutical compounds. *Chemosphere*, *351*, 141216.

[21] Gola, A., Świć, A., & Kozłowski, E. (2020). Intelligent quality control system for surface roughness. *Applied Sciences*, 10(12), 4255.

[22] Lam, S. Y., & Ip, W. H. (2011). An application of artificial intelligence techniques for the control of rework in the production cycle. *Expert Systems with Applications*, 38(7), 8314-8322.

[23] Lee, S. W., & Park, J. S. (2001). Intelligent quality prediction system using machine learning for injection moulding process. *International Journal of Advanced Manufacturing Technology*, 18(5), 329-334.

[24] Chen, D., & Hu, S. J. (2001). Real-time identification and prediction of quality for serial-parallel manufacturing processes. *International Journal of Production Research*, 39(18), 4147-4166.

[25] Liu, S., Zhang, L., Ma, J., Sun, X., Li, H., & Li, Y. (2016). Integration of multisource information for quality prediction in manufacturing. *IEEE Transactions on Industrial Informatics*, 13(4), 1925-1935.

[26] Sujatha, C., & Kumanan, S. (2016). An intelligent approach for quality prediction in machining process using soft computing techniques. *Procedia*

Technology, 25, 546-553.

[27] He, P., Gao, L., Zhao, Y., Liu, Y., & Li, W. (2019). A new method of supply chain quality prediction based on optimized BP neural network algorithm. *Computers & Industrial Engineering*, 135, 1067-1080.

[28] Kleinbaum, D. G., & Klein, M. (2020). *Logistic regression: A self-learning text (Statistics for biology and health)*. Springer.

[29] Kumar, A., Kumar, M., & Goswami, P. (2024). Numerical Solution of Coupled System of Emden-Fowler Equations using artificial neural network technique, *An International Journal of Optimization and Control: Theories & Applications*, 14(1), 62-73.

[30] Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[31] Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)* (pp. 807-814).

[32] Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics* (pp. 315-323). JMLR Workshop and Conference Proceedings.

[33] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436-444.

[34] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(1), 1929-1958.

[35] Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer.

[36] Jayalakshmi T, Santhakumaran A. (2011). Statistical normalization and back propagation for classification. *International Journal of Computer Theory and Engineering*, 3 (1), 1793-8201.

[37] Karasu S, Altan A, Sarac Z, Hacioglu R. (2018). Prediction of bitcoin prices with machine learning methods using time series data. *The 26th Signal Processing and Communications Applications Conference (SIU)*, Izmir, Turkey.

Sema Aydın *holds a BSc degree in Industrial Engineering from Hacettepe University, and a MSc degree in Industrial Engineering from TU Bursa. She has a keen interest in AI applications in industry.*

http://orcid.org/0009-0001-3948-0529

Koray Altun *is an Assistant Professor of Industrial Engineering at TU Bursa and a Research Collaborator at GLORAD. He is also the founder and manager of a startup focused on digital innovation and software consultancy. He holds a Ph.D. and a BSc degree in Industrial Engineering from Gaziantep University and Erciyes University, respectively. He has published papers in well-respected journals and has led consulting engagements in reputable companies with a focus on R&D, technology, and innovation management. His recent research interests include the Innovation Excellence Model, Systematic Innovation, Technology and Innovation Management, and R&D Management.*

http://orcid.org/0000-0003-0357-9495

Appendices

1. Logistic Regression in Python Code

```
import pandas as pd
import numpy as np

# Data loading
veriler = pd.read_csv('RW_Data_1.csv')
print(veriler)

# Converting categorical variables to dummy variables
df_KumasKaliteAdi = pd.get_dummies(veriler["KumasKaliteAdi"], prefix="Kalite",
drop_first=True)
df_KumasDesen = pd.get_dummies(veriler["HamKumasDesenAdi"], prefix="KumasDesen",
drop_first=True)
df_UretimPartiNo = pd.get_dummies(veriler["UretimPartiNo"], prefix="PartiNo",
drop_first=True)
df_RenkKodu = pd.get_dummies(veriler["RenkKodu"], prefix="RenkKodu",
drop_first=True)
df_LotNo = pd.get_dummies(veriler["LotNo"], prefix="LotNo", drop_first=True)
df_MamulOlusumTipKodu = pd.get_dummies(veriler["MamulOlusumTipKodu"],
prefix="MamulOlusumTipKodu", drop_first=True)
df_HamUrunKodu = pd.get_dummies(veriler["HamUrunKodu"], prefix="HamUrunKodu",
drop_first=True)
```

```
df_MasterReceteAdi = pd.get_dummies(veriler["MasterReceteAdi"],
prefix="MasterReceteAdi", drop_first=True)
df_ProsesAdi = pd.get_dummies(veriler["ProsesAdi"], prefix="ProsesAdi",
drop_first=True)
df_IslemAkisKodu = pd.get_dummies(veriler["IslemAkisKodu"], prefix="IslemAkisKodu",
drop_first=True)
df_MakinaAdi = pd.get_dummies(veriler["MakinaAdi"], prefix="MakinaAdi",
drop_first=True)

# Creating a DataFrame for PlanMt
PlanMt = veriler.iloc[:, 8:9].values
mt = pd.DataFrame(data=PlanMt, index=range(4855), columns=['PlanMt'])

# Encoding the dependent variable
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
IsTamir = le.fit_transform(veriler.iloc[:, 0])
tamir = pd.DataFrame(data=IsTamir, index=range(4855), columns=['IsTamir'])

# Merging DataFrames
sonuc = pd.concat([df_KumasKaliteAdi, df_KumasDesen, df_UretimPartiNo, df_RenkKodu,
df_LotNo, df_MamulOlusumTipKodu, df_HamUrunKodu, df_MasterReceteAdi, df_ProsesAdi,
df_IslemAkisKodu, df_MakinaAdi, mt, tamir], axis=1)
print(sonuc)

# Splitting into independent and dependent variables
x = sonuc.iloc[:, 0:-1].values  # independent variables
y = sonuc.iloc[:, -1]           # dependent variable

# Splitting the data into training and testing sets
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.33,
random_state=0)

# Scaling the data
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(x_train)
X_test = sc.transform(x_test)

# Logistic regression
from sklearn.linear_model import LogisticRegression
logr = LogisticRegression(random_state=0, max_iter=1000)
logr.fit(X_train, y_train)  # training the model

# Predictions
y_pred = logr.predict(X_test)
print(y_pred)
print(y_test)

# Evaluation
from sklearn.metrics import confusion_matrix, mean_squared_error, accuracy_score,
f1_score

# Confusion matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)

# Root mean squared error
RMSE = np.sqrt(mean_squared_error(y_test, y_pred))
print("RMSE:", RMSE)

# Accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

# F1 Score
f1 = f1_score(y_test, y_pred)
```

```
print("F1 Score:", f1)

# Visualization of class distribution
import matplotlib.pyplot as plt
plt.bar(np.unique(y), np.bincount(y))
plt.xlabel('Class')
plt.ylabel('Number of Samples')
plt.title('Class Distribution')
plt.show()
```

2. ANN in Python Code

```
# -*- coding: utf-8 -*-
"""
import pandas as pd
import numpy as np
#The CSV file should be located in the same directory as the Python code.
veriler=pd.read_csv('RW_Data_1.csv')
print(veriler)

df_KumasKaliteAdi=pd.get_dummies(veriler["KumasKaliteAdi"],prefix="Kalite",drop_fir
st=True)
#print(df_KumasKaliteAdi)

df_KumasDesen=pd.get_dummies(veriler["HamKumasDesenAdi"],prefix="KumasDesen",drop_f
irst=True)
#print(df_KumasDesen)

df_UretimPartiNo=pd.get_dummies(veriler["UretimPartiNo"],prefix="PartiNo",drop_firs
t=True)
#print(df_UretimPartiNo)

df_RenkKodu=pd.get_dummies(veriler["RenkKodu"],prefix="RenkKodu",drop_first=True)
#print(df_RenkKodu)

df_LotNo=pd.get_dummies(veriler["LotNo"],prefix="LotNo",drop_first=True)
#print(df_LotNo)

df_MamulOlusumTipKodu=pd.get_dummies(veriler["MamulOlusumTipKodu"],prefix="MamulOlu
sumTipKodu",drop_first=True)
#print(df_MamulOlusumTipKodu)

df_HamUrunKodu=pd.get_dummies(veriler["HamUrunKodu"],prefix="HamUrunKodu",drop_firs
t=True)
#print(df_HamUrunKodu)

df_MasterReceteAdi=pd.get_dummies(veriler["MasterReceteAdi"],prefix="MasterReceteAd
i",drop_first=True)
#print(df_MasterReceteAdi)

df_ProsesAdi=pd.get_dummies(veriler["ProsesAdi"],prefix="ProsesAdi",drop_first=True
)
#print(df_ProsesAdi)

df_IslemAkisKodu=pd.get_dummies(veriler["IslemAkisKodu"],prefix="IslemAkisKodu",dro
p_first=True)
#print(df_IslemAkisKodu)

df_MakinaAdi=pd.get_dummies(veriler["MakinaAdi"],prefix="MakinaAdi",drop_first=True
)
#print(df_MakinaAdi)

PlanMt=veriler.iloc[:,8:9].values
mt=pd.DataFrame(data=PlanMt,index=range(4855),columns=['PlanMt'])

IsTamir=veriler.iloc[:,0:1].values
tamir=pd.DataFrame(data=IsTamir,index=range(4855),columns=['IsTamir'])
```

```
from sklearn import preprocessing

# DataFrame merging is used to create a new DataFrame by combining multiple
DataFrames.
sonuc=pd.concat([df_KumasKaliteAdi,df_KumasDesen,df_UretimPartiNo,df_RenkKodu,df_Lo
tNo, df_MamulOlusumTipKodu,df_HamUrunKodu,df_MasterReceteAdi,df_ProsesAdi,
df_IslemAkisKodu, df_MakinaAdi,mt,tamir],axis=1)
#print(sonuc)
x=sonuc.iloc[:,0:-1].values #independent variables
y=sonuc.iloc[:,-1] #dependent variables
#print(x)
#print(y)

# splitting the data into training and testing sets
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.33,random_state=0)

# scaling the data
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
X_train=sc.fit_transform(x_train) #eğitim uyguluyor
X_test=sc.transform(x_test)

#Let's create our artificial neural network model and configure our hidden layer.
from sklearn.neural_network import MLPClassifier
#iki katman ve her katman 6 nörondan oluşacak şekilde model kurulmuştur
mlpcl = MLPClassifier(hidden_layer_sizes=(3,3),activation='relu', solver='lbfgs',
max_iter=1000,random_state=42,learning_rate='constant', learning_rate_init=0.001)

#model = MLPClassifier(hidden_layer_sizes=(64, 64), activation='relu',
solver='adam', max_iter=1000, random_state=42)
mlpcl.fit(X_train, y_train.values.ravel())

# Let's make predictions on our test data.
predictions = mlpcl.predict(X_test)
print(predictions)

# Let's evaluate the performance of our predictions - our algorithm.
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test,predictions))
print(classification_report(y_test,predictions))

from sklearn.metrics import accuracy_score
# Calculate accuracy using your predictions and the actual labels.
accuracy = accuracy_score(y_test, predictions)

# print accuracy
print("Accuracy:", accuracy)

from sklearn.metrics import f1_score
# calculate accuracy
f1 = f1_score(y_test, predictions, pos_label='T')
print("F1 Score:", f1)
```


This work is licensed under a Creative Commons Attribution 4.0 International License. The authors retain ownership of the copyright for their article, but they allow anyone to download, reuse, reprint, modify, distribute, and/or copy articles in IJOCTA, so long as the original authors and source are credited. To see the complete license contents, please visit http://creativecommons.org/licenses/by/4.0/.