Design of a Machine Learning-based Decision Support System for Product Scheduling on Non Identical Parallel Machines

Khalid Ait Ben Hamou
Computer Sciences Engineering Laboratory, Faculty of Sciences, Cadi Ayyad University, Marrakech, Morocco
khalid.aitbenhamou@ced.uca.ma (corresponding author)

Zahi Jarir
Computer Sciences Engineering Laboratory, Faculty of Sciences, Cadi Ayyad University, Marrakech, Morocco
jarir@uca.ac.ma

Selwa Elfirdoussi
Emines - University Mohammed VI Polytechnic, Benguerir, Morocco
selwa.elfirdoussi@emines.um6p.ma

Received: 25 May 2024 | Revised: 1 July 2024 and 7 July 2024 | Accepted: 8 July 2024

 DOI: https://doi.org/10.48084/etasr.7934

ABSTRACT

Production planning in supply chain management faces considerable challenges due to the dynamics and unpredictability of the production environment. Decision support systems based on the evolution of artificial intelligence can provide innovative solutions. In this paper, an approach based on machine learning techniques to solve the problem of scheduling the production of N products on M non-identical parallel machines is proposed. Using regression and classification models, our approach aims to predict overall production costs and assign products to the right machines. Some experiments carried out on simulated data sets demonstrate the relevance of the proposed approach. In particular, the XGBoost model stands out for its superior performance compared with the other tested ML algorithms. The proposed approach makes a significant contribution to the optimization of production scheduling, offering significant potential for improvement in Supply Chain Management.

Keywords-decision support system; machine learning; scheduling problem; supply chain management

I. INTRODUCTION

Decision-making is a major challenge, especially in complex or conflictual situations. The use of computers to assist decision-makers has become indispensable when the amount of data and the number of constraints to be managed increase significantly. The concept of Decision Support Systems (DSS) was developed in the 1970s [1]. The work presented in [2] defined as computer-based technological solutions supporting complex decision-making and problem-solving. Since then, DSS applications have evolved considerably. Once based on limited user interfaces and restricted databases, they were primarily aimed at individual decision-makers. With technological progress and increasing digitization, modern DSS offer advanced functionalities for groups of decision-makers, and even for virtual teams, enabling inter-organizational collaboration [2]. With the rise of Artificial Intelligence (AI), which is now ubiquitous, DSSs have benefited from various areas of AI, becoming smarter and able to process unstructured data and model complex relationships. In Supply Chain Management (SCM), the adoption of AI-based methods, particularly Machine Learning (ML), is providing innovative solutions to complex decision-making problems, particularly in the face of uncertainty and multiple constraints [1]. However, information systems dealing with production scheduling face considerable difficulties [2] due to the dynamics and unpredictability of the production environment, and the complexity of combinatorial problems. To meet these challenges, it is essential to support production planning with intelligent algorithms capable of analyzing and exploiting various types of data.

Recent literature, circumscribed between 2018 and 2023, highlights the growing use of ML to refine scheduling processes at the heart of SCM. It shows that most research

works are characterized by their heterogeneity both in terms of methods and targeted industrial applications.

Authors in [3] conceptualized a genetic programming-based heuristic for optimizing project scheduling with limited resources. Their distinctive approach relies on the creation of priority rules through genetic mutation and recombination. Following this innovative impulse, authors in [4] explored the use of the Deep Q Network algorithm for automated scheduling of production tasks, eliminating the need for human intervention. Authors in [5] took a turn towards real-time scheduling in a smart factory, using Reinforcement Learning to overcome the constraints of the MDRs method, demonstrating increased flexibility in the face of variations in the production environment. In 2019 authors in [6] presented a predictive-reactive framework that combines ML and simulation-based optimization to refine production planning. Authors in [7] focused on a multi-objective flow shop scheduling problem, developing a mathematical model to minimize various performance indicators using opposing learning strategies and cluster analysis. Reinforcement learning has been successfully applied in the context of chemical production by [8] and in the integration of genetic algorithms and ant colonies by the authors in [9] who sought to accelerate the convergence of scheduling solutions. Authors in [10] presented a reordering framework, considering balancing schedule adjustments with the accumulation of delays. Authors in [11] developed a hybrid control solution combining Big Data techniques and ML for reactive management of predictive production planning systems. Authors in [12] took a turn towards automating the solution of job-shop scheduling problems, with the use of a graphical neural network to improve the generalization of solutions. Authors in [13] dealt with customer order scheduling using a two-stage optimization method, incorporating ML to refine production schedules. Authors in [14] recommended the use of decision trees to select priority rules, associating performance indicators with specific rules. The application of Neuro Evolution of Augmenting Topologies (NEAT) [15] in complex scheduling problems has demonstrated its superiority over traditional methods, offering a new perspective for hybrid two-stage workshops. Authors in [16] considered the use of ML for estimating processing times, thus optimizing the scheduling of parallel machines. Authors in [17] integrated predictive maintenance into production scheduling, employing deep learning models to predict equipment service life. Significant advances were made by the authors in [17, 18] who used the graphical neural network for applications in job-shop scheduling and flexible workshop planning. In addition, authors in [20] applied Genetic Programming to flexible shop scheduling, while authors in [19] combined ML with reasoning about domain problems to develop efficient dispatching rules. More recently, authors in [21] adopted a deep Q-learning approach to solve job-shop scheduling problems, and authors in [22] presented a Markov Decision Process model for reinforcement learning in scheduling. Authors in [23] improved the solution of job-shop scheduling problems by exploiting ML techniques to predict an initial solution. Finally, authors in [24, 25] have pushed the boundaries of Deep Reinforcement Learning, combining it with graphical neural networks to address flexible job shop scheduling.

This review highlights the digital transformation driven by ML, enabling more responsive, efficient, and flexible SCM, testifying to a burgeoning field of research and promising prospects for the future. Although previous research has significantly contributed to the integration of ML techniques into scheduling within supply chain management, certain gaps remain and serve as the basis for our study. Firstly, except for [22], the reviewed works focus on specific production environments and do not discuss the ability to generalize scheduling solutions to different industries or diversified production systems. Secondly, although powerful, the models used can present significant complexity in their configuration and optimization, requiring advanced technical expertise. Thirdly, the integration of predictive maintenance into scheduling is little addressed, with only [10, 16] having explored this aspect. These studies do not address how scheduling and predictive maintenance can be jointly optimized to improve both equipment durability and efficiency.

Our research aims to fill these gaps by developing a scheduling method that dynamically adapts to variations in production conditions and integrates predictive maintenance, thereby increasing the accuracy and practical applicability of scheduling solutions.

II. MATERIALS AND METHODS

A. Scheduling Poblem Analysis

Production scheduling is defined as the allocation of available production resources over time to meet some set of performance criteria [26]. As mentioned in [27], production scheduling is concerned with the efficient allocation of resources over time for the manufacture of goods. In general, it aims to satisfy cost minimization or production maximization objectives and to improve the efficiency of a supply chain in terms of delivery times and production costs.

This paper deals with a scheduling problem where the resources are non-identical parallel machines, and the tasks are an order book containing several items to be produced. The problem is complex in view of the number of constraints to be satisfied. The constraints considered in this work are:

1. Delivery time constraint: An order book has a delivery start and end date. This duration is divided into a unit of time called a period (an hour in our case).
2. Constraint of heterogeneous machines: Not all machines are identical; there are special machines that produce only a few products.
3. Production cost constraint: An item can be produced on several machines, with different costs, and in a different number of periods. The quantity produced per hour differs from one machine to another. The choice of machine for a product must therefore take this constraint into account.
4. Maintenance schedule constraints: The various machines are serviced at preventive maintenance periods known in advance. This schedule is set by the maintenance managers prior to the planning of product scheduling on the

machines. No production will take place on these lines during these periods.

5. Delivery window constraint for an order: The end of production of a product must respect the delivery window. In fact, the end of production must be before the final date set by the customer, and must not be before the delivery start date, to avoid stockpiling the produced items. This storage can be costly.

The complexity of this problem lies in the heterogeneity of resources and the non-authorization of preemption, as well as the delivery windows imposed by customers. This classifies it as an NP-Hard problem. The vast majority of scheduling problems are NP-Hard and therefore cannot be formulated as linear programs [28]. Solving this scheduling problem requires accurate, representative data. In the context of this study, we have opted for the use of simulation data, generated using Python programs specifically developed for this purpose. The following section details these data sets, covering both structured data, data from the scheduling problems studied and the optimal solutions generated.

B. Dataset

In the absence of real data suited to our research objectives, we proceeded in three phases to create a dataset representative of scheduling problems for N products on M non-identical machines.

The first phase is devoted to generating the structural data essential for modeling production instances. This process includes determining whether it is possible to produce item i on machine j, and quantifying the operational parameters if production is feasible. For each item-machine pair where production is possible, the program establishes the production rate, expressed in quantity of finished products per hour, and the hourly cost associated with this production for each machine concerned.

In the second phase, we generated scheduling problem instances. Each instance is defined by a specific list of products, identified by Pi, which must be produced. For each product, the program determines not only the quantity required for production, but also the delivery period. In addition, the program assigns maintenance time slots to each machine, delimited by a start and end date, during which the machines are not available for production.

Finally, the third phase consists of calculating the set of possible solutions and selecting the optimum solution in terms of minimum production cost.

Algorithm 1 illustrates the method used to calculate the optimal solution to each instance of the scheduling problem. It starts by establishing a scheduling table, considering machine maintenance periods (line 2). For each possible permutation of products (line 4), the algorithm evaluates the cost and allocation of products to machines, rejecting unfeasible permutations (line 10) and selecting the machine with the lowest production cost for each product (line 12). The permutation that results in the lowest total cost is recorded as the optimum solution (line 19) and stored in the database (line 22).

Although the algorithm guarantees the discovery of the optimal solution, it is mainly applicable to small and medium-sized instances due to the increasing computational complexity of large datasets.

```
Algorithm 1:
1 FUNCTION generate solutions(id_problem)
2  INIT solution with maintenance periods
3  RETRIEVE problem details for id_problem
4  FOR EACH permutation of products
5     INITIALIZE cost for permutation
6     FOR EACH product in permutation
7       FIND manufacturable machines
8       GET list of possible machines
9       IF no possible machines
10          REJECT permutation
11      ELSE
12          SELECT least cost machine
13          ASSIGN product to machine
14          UPDATE solution
15          UPDATE cost for permutation
16    END FOR
17    IF current cost < optimal cost
18      SET optimal cost to current cost
19      RECORD optimal solution &
20      RECORD permutation
21    END FOR
22    STORE optimal solution in database
23 END FUNCTION
24
25 FOR EACH problem instance
26   CALL generate_solutions (instance id)
24 END FOR
```

Our dataset comprises a total of 25710 instances of the problem of scheduling 6 products on 3 machines over a 12-period horizon, with their optimal solutions. This dataset is stored and managed using MongoDB, a NoSQL database management system renowned for its flexibility and performance with large amounts of data.

C. Data Pre-Processing and Standardization

Data pre-processing is an important step in our methodology, not least because of the heterogeneous nature of our feature types, which come in the form of vectors, matrices and dictionaries. To make these complex data suitable for ML models, specific pre-processing was necessary, as summarized in TABLE I for a scheduling problem involving p products on m machines. The total number of features is: 4p+2m+3pm+1 in our case, where p = 6 and m = 3, the total value is 85. To standardize continuous characteristics, such as product quantities and production rates, we used StandardAero from the scikit-learn library. This tool adjusts each feature so that the mean is zero and the standard deviation is one, making predictive models less sensitive to the varying scales of different features and contributing to faster convergence during learning.

TABLE I. DATA PRE-PROCESSING

Features	Description	Count
quantity_p{i}	Quantity of product requested P{i}	p
delivery_start_hour_p{i}	Delivery start time of P{i}	p
delivery_end_hour_p{i}	P{i} delivery end time	p
P{i}_producible_m{j}	Is P{i} productive by M{j}	p * m
P{i}_production_cost_m{j}	Cost of one period of production of P{i} by machine M{j}	p * m
P{i}_production_qty_m{j}	Quantity of P{i} that can be produced by M{j} per period	p * m
maint_start_hour_m{k}	Machine maintenance start period M{k}	m
maint_end_hour_m{k}	Machine maintenance end time M{k}	m
P{i} = k	The product P{i} will be assigned to the machine M{k}.	p
optimal_cost	Optimum production cost	1

D. Proposed Methodology

Our approach to solving the production scheduling problem in a supply chain that integrates ML into a structured DSS. Figure 1 shows an overview of the proposed DSS. Upstream, structural, and problem data feed the ML model. The latter then performs classification and regression predictions, which are then integrated into the ordination program to generate the final scheduling solution. The diagram of our methodology details the following components:

1) Composition of the Dataset

The dataset is composed of two categories of data:

- Structural data: This data is fixed and does not vary between different scheduling problems. It includes information such as production rates and costs associated with producing a product on a given machine.
- Problem data: Specific to each problem instance, this data includes quantities of each product to be produced, time windows for product delivery, and scheduled maintenance periods for each machine.

2) Output Characteristics

There are two types of output characteristics:

- Categorical characteristics (P{i}): These represent the assignment of a product to a machine, where each product takes as its value the number of the machine assigned to it.
- Continuous characteristic (Optimal cost): A single characteristic reflecting the optimal production cost for the entire scheduling problem.

3) ML Models

- Regression for optimal cost: We use regression models to predict the continuous characteristic of optimal cost, thus optimizing the selection of production parameters to minimize expenditure.
- Classification for product assignment: Classification models are used to determine the assignment of products to machines, with the creation of a separate model for each product to predict the specific machine to which it will be assigned.
- Post-prediction ordering: Having predicted the assignment of products to machines, a third Python program is used to order the products assigned to each machine, respecting the delivery time constraints associated with each product.

This methodology provides a comprehensive framework for integrating ML into production scheduling problems, addressing both production assignment forecasting and production cost estimation. The aim is to provide optimal scheduling that respects all the operational and economic constraints inherent in SCM.

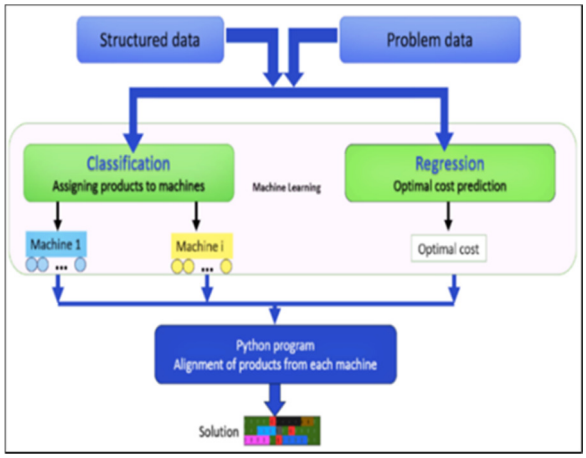

Fig. 1. The proposed methodology.

4) ML Algorithms Used

To address the two aspects of our production scheduling problem, optimal cost prediction and product allocation to machines, we have selected a suite of ML algorithms [29] based on their performance in regression and classification. ML algorithms used for regression (optimal cost prediction):

- Linear regression: The foundation of predictive methods for its interpretability and speed of calculation, used here as a basic point of comparison.
- KNN Regression (k=11): A decisive hyperparameter, the number of nearest neighbors, was chosen after an exhaustive search. k=11 was determined as offering the best performance from a range of values from 1 to 50. Figure 2 shows the score obtained as a function of the value of k.
- Regression Decision Tree (max_depth = 9): In a similar way, the maximum depth of the tree has been optimized to avoid overlearning while capturing the complexity of the data as explained in Figure 3.

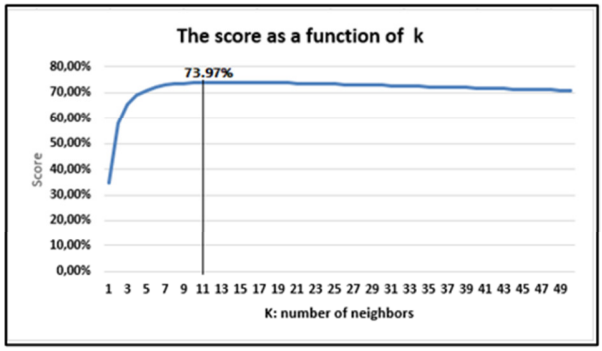

Fig. 2. Performance of the KNN model as a function of the number of neighbors.

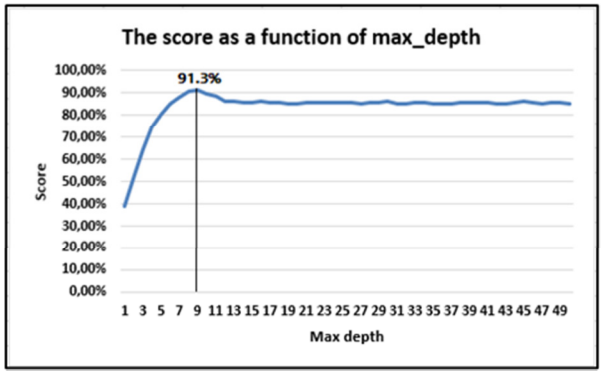

Fig. 3. Decision Tree regression performance as a function of maximum tree depth.

- Extreme Gradient Boosting (XGBoost) Regression: the model proposed by [30] renowned for its efficiency and accuracy, is an advanced implementation of the gradient boosting algorithm.

5) ML Algorithms used for Classification (Product Assignment):

- Logistic regression: Despite its simplicity, this robust model is the benchmark for binary and multiclass classification.
- KNN Classification (k=11): The same k selection principle was applied here, ensuring consistency between our regression and classification models.
- Decision Tree Classification.
- XGBoost Classification: Employs the same powerful algorithm as for regression but adapted for categorical results.

Each categorical output characteristic (P{i}) was addressed with a separate model, enabling more accurate prediction and fine-grained analysis of individual machine performance.

E. Metrics Used

To evaluate the effectiveness of the ML models, we chose recognized metrics that reflect the objectives of our study.

For regression models, performance was measured by RMSE (Root Mean Square Error), R2 score, MAE (Mean Absolute Error) and Explained Variance Score:

- RMSE: This metric is particularly informative as it amplifies and penalizes larger errors, providing a strict assessment of performance.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \quad (1)$$

- R^2 score (coefficient of determination): R^2 is a statistical measure that tells us how much of the variance in the data is explained by the model.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2} \quad (2)$$

- MAE: Gives an idea of the mean error without squaring it, thus offering a more direct perspective of the mean absolute error.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \quad (3)$$

- Explained Variance Score: To see how well our model explains the variation in the data. This metric measures the proportion of response variance that can be predicted from the explanatory variables.

$$\text{Explained Variance} = 1 - \frac{Var(y-\hat{y})}{var(y)} \quad (4)$$

where y_i is the true values of y, $\hat{y}_i$ is the predicted values of y and $\hat{y}$ is the average value of y.

For classification models, we evaluated performance using metrics that capture how effectively our models correctly identify target classes. In the following TP represents True Positives, FP represents False Positives, TN represents True Negatives, and FN represents False Negatives.

- Accuracy: This metric calculates the percentage of correct predictions relative to the total number of predictions made, providing an overall measure of model performance.

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{Total number of predictions}} \quad (5)$$

- Precision: Reflects the proportion of actual positive identifications among the items labeled as positive by the model. It is an indicator of the reliability of the model's positive predictions [31].

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} \quad (6)$$

- Recall (or Sensitivity): This metric measures the model's ability to identify all real instances of a particular class.

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}} \quad (7)$$

- F1-Score: The harmonic mean of Precision and Recall, provides a single score that balances these two metrics, particularly useful when classes are unbalanced.

$$\text{F1} - \text{Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision+Recall}} \quad (8)$$

III. EXPERIMENTAL RESULTS

A. Regression Results

Table II summarizes the performance of each model according to RMSE, MAE, Explained Variance, and R^2 Score, allowing a direct comparison of their effectiveness in predicting production costs. The results indicate that XGBoost showed the best performance with an RMSE of 1090.16 and an R^2 Score of 0.96, suggesting a strong fit with the test data, which implies a superior ability of this model to model the complexity of production costs compared to the other models tested.

TABLE II. PERFORMANCE OF ML MODELS ACCORDING TO RMSE, MAE, EXPLAINED VARIANCE, AND R2 SCORE

Model ML	RMSE	MAE	Explained Variance	R2 Score
KNN regression (K=11)	2900.80	2242.57	0.74	0.74
Decision Tree	1676.67	1099.49	0.91	0.91
Linear regression	1415.74	880.53	0.94	0.94
XGBoost regression	1090.16	707.51	0.96	0.96

Figure 4 shows a comparative RMSE plot for each model, visually illustrating the relative effectiveness of each algorithm. XGBoost's clear superiority on this graph reflects its robustness under the varied conditions of our dataset.

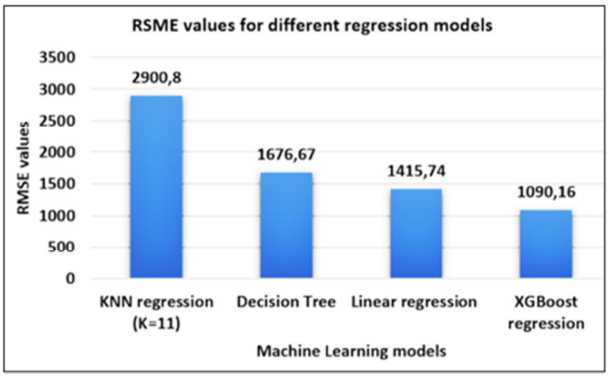

Fig. 4. Graphical comparison of RMSE values.

The poor performance of the KNN model is perhaps explained by the sensitivity to scale of the input features. Some features have much wider value ranges than others, and will dominate the distance between points, which can lead to poor performance. Before the standardization of input features, the score was much lower. Our data have non-linear relationships that KNN has difficulty capturing with its proximity-based prediction method, so the XGBoost model, which can capture non-linear relationships and interactions between features, was able to perform better. To visualize how close the predictions are to the actual values, Figure 5 shows scatter plots for the ML regression models used. Scatter plots show the relationship between model-predicted values (y-axis) and actual values (x-axis). Ideally, predictions should be as close as possible to the black diagonal line, which represents a perfect prediction where predicted values are equal to actual values.

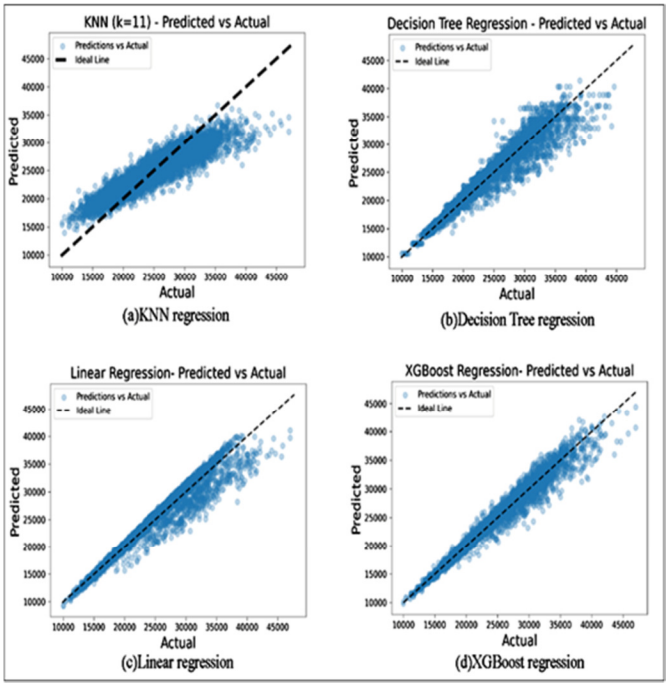

Fig. 5. Comparison of predicted vs. actual values for different regression models.

For the Decision Tree algorithm, the points appear to be close to the ideal line, indicating that the model's predictions are relatively accurate. For the KNN, the predictions are a little more scattered in relation to the ideal line, which could indicate less precision in the predictions compared to the Decision Tree. The Linear Regression plot shows that predictions are tightly aligned around the ideal line, suggesting that this model has performed well and that predictions are consistent with actual values. Finally, the XGBoost Regression shows a distribution of points very close to the ideal line, suggesting that this model has superior predictive ability compared to other models, with predictions very much aligned with actual values.

B. Classification Results

The metrics used to evaluate these models were Accuracy, Precision, Recall, and F1-Score.Table III presents these metrics for each product feature (P1-P6), highlighting the models' performance in correctly assigning products to machines. For example, for P1, XGBoost outperformed the other models with an Accuracy of 0.96 and an F1-Score of 0.84, demonstrating its ability to correctly classify products with great accuracy and efficiency. For P3, all models achieve perfection with a score of 1.00 on all metrics, suggesting that P3 has highly distinctive features that are easily learned by all models because the structured data shows that P3 can only be produced by the m3 machine, and the value of P3 is the same for all dataset instances. Figure 6 shows the ROC curves for feature P1 generated by different ML algorithms. These curves are a visual indicator of the discriminatory performance of the models, where an Area Under the Curve (AUC) close to 1.0 testifies to excellent classification ability. For P1, the XGBoost model clearly stands out with an AUC of 0.97, denoting near-perfect accuracy in class distinction. Next are the logistic

regression model and the decision tree, with respective AUCs of 0.90 and 0.77, indicating significant classification competence, albeit slightly inferior to XGBoost. The KNN model, despite an AUC of 0.69, shows a more moderate performance, suggesting a less clear distinction between classes for this specific model. These ROC curves confirm the robustness of the XGBoost approach in our classification context, reaffirming its potential as a predominant tool for the accurate allocation of products to machines in DSSs.

TABLE III. RESULTS OF ML CLASSIFICATION MODEL

	ML Model	Accuracy	Precision	Recall	F1-Score
P1	KNN	0.92	0.96	0.50	0.48
	Decision Tree	0.91	0.72	0.77	0.74
	Logistic Regression	0.92	0.77	0.60	0.63
	XGBoost	0.96	0.89	0.80	0.84
P2	KNN	0.88	0.63	0.33	0.31
	Decision Tree	0.87	0.57	0.55	0.56
	Logistic Regression	0.89	0.64	0.45	0.49
	XGBoost	0.92	0.75	0.61	0.67
P3	KNN	1.00	1.00	1.00	1.00
	Decision Tree	1.00	1.00	1.00	1.00
	Logistic Regression	1.00	1.00	1.00	1.00
	XGBoost	1.00	1.00	1.00	1.00
P4	KNN	0.87	0.29	0.33	0.31
	Decision Tree	0.76	0.44	0.49	0.46
	Logistic Regression	0.88	0.64	0.46	0.49
	XGBoost	0.92	0.76	0.62	0.67
P5	KNN	0.96	0.48	0.50	0.49
	Decision Tree	0.97	0.80	0.79	0.79
	Logistic Regression	0.96	0.75	0.52	0.53
	XGBoost	0.98	0.94	0.81	0.86
P6	KNN	0.97	0.48	0.50	0.49
	Decision Tree	0.98	0.80	0.85	0.83
	Logistic Regression	0.97	0.68	0.52	0.53
	XGBoost	0.99	0.98	0.79	0.86

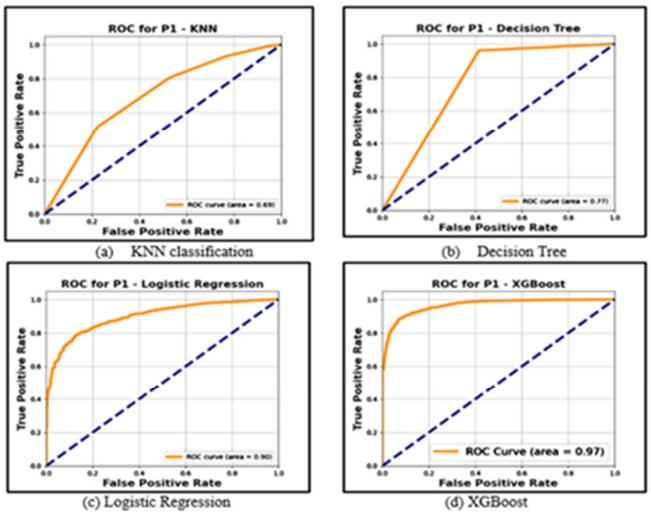

Fig. 6. Receiver Operating Characteristic (ROC) curves for P1.

The following section will explore the implications of these results, analyzing the performance of the different models and the potential reasons behind the observed trends. We will also discuss the models' limitations and prospects for future research.

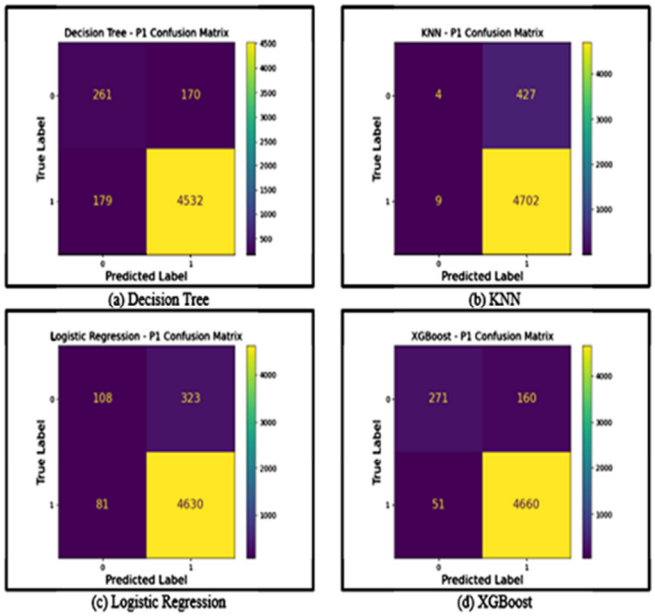

Fig. 7. Confusion Matrices for P1.

IV. DISCUSSION

The experimental results presented highlight the robustness of the XGBoost approach, which consistently outperformed other classification models in our study. With near-perfect ROC curve AUCs for features P1, P5 and P6, XGBoost proved its ability to operate with high accuracy, even when classification conditions were less distinct and more complex.

The previous section illustrated XGBoost's superiority with metrics such as Accuracy, Precision and F1-Score, which, in correlation with the ROC curves, strengthen the case for its adoption for similar classification tasks in production contexts. Notably, for features where other models faltered, such as P5 and P6, XGBoost demonstrated exceptional performance, suggesting its ability to handle the variability and complexity inherent in production data. These results are consistent with the findings of other research in the field of ML on other problems [31-36]. However, it's important to recognize that, despite these successes, there are inherent limitations to the application of ML models in real-world environments. The models, while accurate, are subject to the quality of the input data and can be sensitive to use cases that deviate from training conditions.

Furthermore, the evaluation of regression models showed that the XGBoost model, with the lowest RMSE and the highest R^2 Score, illustrated an exceptional ability to anticipate costs accurately and consistently. Scatter plots of predictions versus actual values confirmed XGBoost's superiority, reflecting a dense concentration of points along the ideal line, symbolizing near-perfect predictions. The linear regression and Decision Tree models also performed well, but with greater variance, indicating slightly lower accuracy in cost modeling. The KNN model, despite its simplicity and speed of execution, showed fewer convincing results in terms of RMSE and R^2,

which can be attributed to its difficulty in capturing the complexity and non-linearity of the data.

Although our results highlight the power of the XGBoost approach for classifying production data, a notable constraint of our study is the scalability of exhaustive solution computation. This method, while optimal for small instances, becomes impractical for large datasets due to its exponential computational cost. This reality points to the urgency of adopting approximation strategies or heuristics that could deliver near-optimal solutions faster, thus better aligning with the dynamic requirements of modern production environments.

In addition, this research paves the way for future investigations, where the exploration of more advanced techniques, such as deep learning or hyperparameter optimization, could be employed to further improve model performance. Future research could also examine the impact of integrating real-time feedback and dynamic adjustments into models, aiming to make production systems even more responsive and economically efficient.

V. CONCLUSION

This research has tackled the complex challenge of production scheduling by combining the capabilities of ML with a structured DSS for SCM. Although previous research has significantly contributed to integrating ML techniques into scheduling within SCM, several issues remain open. Specifically, (a) many contributions deal with specific production environments and do not discuss the ability to generalize scheduling solutions to different industries or diversified production systems, (b) the proposed models present significant complexity in their configuration and optimization, requiring advanced technical expertise, and (c) the integration of predictive maintenance into scheduling is rarely addressed.

To address this issue, we propose a methodology that offers a comprehensive approach integrating structural and problem-specific data to feed an effective predictive model, having the advantage of optimizing costs and product allocation to machines. The strength of our approach lies in its ability to dynamically adapt scheduling solutions to variations in production data, while taking operational constraints into account. Through the elaboration of a rich and well-structured dataset, we were able to demonstrate the relevance of regression models for estimating optimal costs and classification models for predicting product assignment. Our approach has demonstrated accurate prediction of production assignments while providing cost estimates in line with current economic requirements. The proposed methodology incorporates a post-prediction ordering stage, which enables the products assigned to each machine to be logically organized, guaranteeing on-time delivery and maximizing the efficiency of the production process. This refinement is crucial for supply chain practitioners who are looking for solutions that are not only cost-optimized, but also achievable on time. However, we recognize that exhaustive computation of optimal solutions for all permutations has limitations in terms of scalability, particularly in the case of large data sets. This limitation opens up prospects for the integration of heuristic and optimization methods that could speed up the search for solutions while maintaining acceptable quality.

To summarize, our contribution provides a solid foundation for the future integration of intelligent DSSs into production scheduling. Future research could focus on improving computational efficiency and exploring even more sophisticated predictive models, propelling the production industry into an era of digitization and automation.

REFERENCES

[1] D. Ni, Z. Xiao, and M. K. Lim, "A systematic review of the research trends of machine learning in supply chain management," *International Journal of Machine Learning and Cybernetics*, vol. 11, no. 7, pp. 1463–1482, Jul. 2020, https://doi.org/10.1007/s13042-019-01050-0.

[2] G. Schmidt, "A Decision Support System for Production Scheduling," *Journal of Decision Systems*, vol. 1, no. 2–3, pp. 243–260, Jan. 1992, https://doi.org/10.1080/12460125.1992.10511527.

[3] M. Đumić, D. Šišejković, R. Čorić, and D. Jakobović, "Evolving priority rules for resource constrained project scheduling problem with genetic programming," *Future Generation Computer Systems*, vol. 86, pp. 211–221, Sep. 2018, https://doi.org/10.1016/j.future.2018.04.029.

[4] B. Waschneck *et al.*, "Optimization of global production scheduling with deep reinforcement learning," *Procedia CIRP*, vol. 72, pp. 1264–1269, Jan. 2018, https://doi.org/10.1016/j.procir.2018.03.212.

[5] Y.-R. Shiue, K.-C. Lee, and C.-T. Su, "Real-time scheduling for a smart factory using a reinforcement learning approach," *Computers & Industrial Engineering*, vol. 125, pp. 604–614, Nov. 2018, https://doi.org/10.1016/j.cie.2018.03.039.

[6] S. L. Takeda Berger, R. M. Zanella, and E. M. Frazzon, "Towards a data-driven predictive-reactive production scheduling approach based on inventory availability," *IFAC-PapersOnLine*, vol. 52, no. 13, pp. 1343–1348, Jan. 2019, https://doi.org/10.1016/j.ifacol.2019.11.385.

[7] L. He, W. Li, Y. Zhang, and Y. Cao, "A discrete multi-objective fireworks algorithm for flowshop scheduling with sequence-dependent setup times," *Swarm and Evolutionary Computation*, vol. 51, pp. 100575, Dec. 2019, https://doi.org/10.1016/j.swevo.2019.100575.

[8] C. D. Hubbs, C. Li, N. V. Sahinidis, I. E. Grossmann, and J. M. Wassick, "A deep reinforcement learning approach for chemical production scheduling," *Computers & Chemical Engineering*, vol. 141, pp. 106982, Oct. 2020, https://doi.org/10.1016/j.compchemeng.2020.106982.

[9] K. Guo, M. Yang, and H. Zhu, "Application research of improved genetic algorithm based on machine learning in production scheduling," *Neural Computing and Applications*, vol. 32, no. 7, pp. 1857–1868, Apr. 2020, https://doi.org/10.1007/s00521-019-04571-5.

[10] Y. Li, S. Carabelli, E. Fadda, D. Manerba, R. Tadei, and O. Terzo, "Machine learning and optimization for production rescheduling in Industry 4.0," *The International Journal of Advanced Manufacturing Technology*, vol. 110, no. 9, pp. 2445–2463, Oct. 2020, https://doi.org/10.1007/s00170-020-05850-5.

[11] C. Morariu, O. Morariu, S. Răileanu, and T. Borangiu, "Machine learning for predictive scheduling and resource allocation in large scale manufacturing systems," *Computers in Industry*, vol. 120, pp. 103244, Sep. 2020, https://doi.org/10.1016/j.compind.2020.103244.

[12] C. Zhang, W. Song, Z. Cao, J. Zhang, P. S. Tan, and X. Chi, "Learning to Dispatch for Job Shop Scheduling via Deep Reinforcement Learning," in *Advances in Neural Information Processing Systems*, 2020, vol. 33, pp. 1621–1632.

[13] Z. Shi, H. Ma, M. Ren, T. Wu, and A. J. Yu, "A learning-based two-stage optimization method for customer order scheduling," *Computers & Operations Research*, vol. 136, pp. 105488, Dec. 2021, https://doi.org/10.1016/j.cor.2021.105488.

[14] W. Guo, M. Vanhoucke, J. Coelho, and J. Luo, "Automatic detection of the best performing priority rule for the resource-constrained project scheduling problem," *Expert Systems with Applications*, vol. 167, pp. 114116, Apr. 2021, https://doi.org/10.1016/j.eswa.2020.114116.

[15] S. Lang, T. Reggelin, J. Schmidt, M. Müller, and A. Nahhas, "NeuroEvolution of augmenting topologies for solving a two-stage hybrid flow shop scheduling problem: A comparison of different solution strategies," *Expert Systems with Applications*, vol. 172, pp. 114666, Jun. 2021, https://doi.org/10.1016/j.eswa.2021.114666.

[16] H. Yamashiro and H. Nonaka, "Estimation of processing time using machine learning and real factory data for optimization of parallel machine scheduling problem," *Operations Research Perspectives*, vol. 8,pp. 100196, Jan. 2021, https://doi.org/10.1016/j.orp.2021.100196.

[17] S. Zhai, B. Gehring, and G. Reinhart, "Enabling predictive maintenance integrated production scheduling by operation-specific health prognostics with generative deep learning," *Journal of Manufacturing Systems*, vol. 61, pp. 830–855, Oct. 2021, https://doi.org/10.1016/j.jmsy.2021.02.006.

[18] J. Juros, M. Brcic, M. Koncic, and M. Kovac, "Exact solving scheduling problems accelerated by graph neural networks," in *2022 45th Jubilee International Convention on Information, Communication and Electronic Technology (MIPRO)*, Feb. 2022, pp. 865–870, https://doi.org/10.23919/MIPRO55190.2022.9803345.

[19] K. Lei *et al.*, "A multi-action deep reinforcement learning framework for flexible Job-shop scheduling problem," *Expert Systems with Applications*, vol. 205, pp. 117796, Nov. 2022, https://doi.org/10.1016/j.eswa.2022.117796.

[20] R. Braune, F. Benda, K. F. Doerner, and R. F. Hartl, "A genetic programming learning approach to generate dispatching rules for flexible shop scheduling problems," *International Journal of Production Economics*, vol. 243, pp. 108342, Jan. 2022, https://doi.org/10.1016/j.ijpe.2021.108342.

[21] S. Jungbluth, N. Gafur, J. Popper, V. Yfantis, and M. Ruskowski, "Reinforcement Learning-based Scheduling of a Job-Shop Process with Distributedly Controlled Robotic Manipulators for Transport Operations," *IFAC-PapersOnLine*, vol. 55, no. 2, pp. 156–162, Jan. 2022, https://doi.org/10.1016/j.ifacol.2022.04.186.

[22] A. Rinciog and A. Meyer, "Towards Standardising Reinforcement Learning Approaches for Production Scheduling Problems," *Procedia CIRP*, vol. 107, pp. 1112–1119, Jan. 2022, https://doi.org/10.1016/j.procir.2022.05.117.

[23] E. Morinaga, X. Tang, K. Iwamura, and N. Hirabayashi, "An improved method of job shop scheduling using machine learning and mathematical optimization," *Procedia Computer Science*, vol. 217, pp. 1479–1486, Jan. 2023, https://doi.org/10.1016/j.procs.2022.12.347.

[24] C.-L. Liu, C.-J. Tseng, T.-H. Huang, and J.-W. Wang, "Dynamic Parallel Machine Scheduling With Deep Q-Network," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 53, no. 11, pp. 6792–6804, Aug. 2023, https://doi.org/10.1109/TSMC.2023.3289322.

[25] W. Song, X. Chen, Q. Li, and Z. Cao, "Flexible Job-Shop Scheduling via Graph Neural Network and Deep Reinforcement Learning," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 2, pp. 1600–1610, Oct. 2023, https://doi.org/10.1109/TII.2022.3189725.

[26] S. C. Graves, "A Review of Production Scheduling," *Operations Research*, vol. 29, no. 4, pp. 646–675, May 1981, https://doi.org/10.1287/opre.29.4.646.

[27] F. A. Rodammer and K. P. White, "A recent survey of production scheduling," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 18, no. 6, pp. 841–851, Aug. 1988, https://doi.org/10.1109/21.23085.

[28] M. Pinedo and K. Hadavi, "Scheduling: Theory, Algorithms and Systems Development," in *Operations Research Proceedings 1991*, Berlin, Heidelberg, 1992, pp. 35–42, https://doi.org/10.1007/978-3-642-46773-8_5.

[29] S. Angra and S. Ahuja, "Machine learning and its applications: A review," in *2017 International Conference on Big Data Analytics and Computational Intelligence (ICBDAC)*, Mar. 2017, pp. 57–60, https://doi.org/10.1109/ICBDACI.2017.8070809.

[30] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA, May 2016, pp. 785–794, https://doi.org/10.1145/2939672.2939785.

[31] A. Saini, K. Guleria, and S. Sharma, "An Efficient Deep Learning Model for Eye Disease Classification," Jun. 2023, pp. 1–6, https://doi.org/10.1109/SCSE59836.2023.10215000.

[32] P. Hajek, M. Z. Abedin, and U. Sivarajah, "Fraud Detection in Mobile Payment Systems using an XGBoost-based Framework," *Information Systems Frontiers*, vol. 25, no. 5, pp. 1985–2003, Oct. 2023, https://doi.org/10.1007/s10796-022-10346-6.

[33] S. Ramaneswaran, K. Srinivasan, P. M. D. R. Vincent, and C.-Y. Chang, "Hybrid Inception v3 XGBoost Model for Acute Lymphoblastic Leukemia Classification," *Computational and Mathematical Methods in Medicine*, vol. 2021, no. 1, pp. 2577375, 2021, https://doi.org/10.1155/2021/2577375.

[34] R. Szczepanek, "Daily Streamflow Forecasting in Mountainous Catchment Using XGBoost, LightGBM and CatBoost," *Hydrology*, vol. 9, no. 12, pp. 226, Dec. 2022, https://doi.org/10.3390/hydrology9120226.

[35] W. Ismaiel, A. Alhalangy, A. O. Y. Mohamed, and A. I. A. Musa, "Deep Learning, Ensemble and Supervised Machine Learning for Arabic Speech Emotion Recognition," *Engineering, Technology & Applied Science Research*, vol. 14, no. 2, pp. 13757–13764, Apr. 2024, https://doi.org/10.48084/etasr.7134.

[36] C. Matara, S. Osano, A. O. Yusuf, and E. O. Aketch, "Prediction of Vehicle-induced Air Pollution based on Advanced Machine Learning Models," *Engineering, Technology & Applied Science Research*, vol. 14, no. 1, pp. 12837–12843, Feb. 2024, https://doi.org/10.48084/etasr.6678.

[37] H. Al-Dossari, F. A. Nughaymish, Z. Al-Qahtani, M. Alkahlifah, and A. Alqahtani, "A Machine Learning Approach to Career Path Choice for Information Technology Graduates," *Engineering, Technology & Applied Science Research*, vol. 10, no. 6, pp. 6589–6596, Dec. 2020, https://doi.org/10.48084/etasr.3821.