Flexible Services and Manufacturing Journal
https://doi.org/10.1007/s10696-025-09640-7

Fast risk estimation for job shop scheduling solutions under interval uncertainty via machine learning

Alice Calamita[1] · **Carlo Meloni**[1] · **Marco Pranzo**[2] · **Marcella Samà**[3]

Received: 5 November 2024 / Accepted: 8 October 2025

Abstract
This paper addresses the evaluation of quantile-based risk measures of the makespan in complex job shop scheduling problems represented as activity networks. We consider the case where activity durations are uncertain and only the range of possible duration values is known in advance for each activity. Risk assessment is a crucial step in activity scheduling as it enables decision-makers proactively address delays or worsening of the makespan, avoiding incurring extra costs due to missed deadlines or worsening of the service quality. In particular, we focus on the value-at-risk and the conditional-value-at-risk of the makespan associated with given a feasible schedule affected by uncertainty. Calculating these risk measures with an exact approach is computationally expensive and becomes progressively less efficient as problem complexity grows. We therefore propose a machine learning-assisted approach that provides a rapid and accurate estimate of these risk indicators based on specific features of the given activity network. Machine learning models can provide fast risk assessment by leveraging collected data of risk evaluations, and are suitable for integration into real-time risk management and optimization-based decision support systems. Computational experiments conducted on a wide set of benchmark instances demonstrate the effectiveness of the proposed approach, achieving very high accuracy in risk estimation while maintaining consistently fast computational times, regardless of the problem's scale or complexity.

Keywords Risk evaluation · Scheduling · Makespan · Machine learning · Activity networks · Job shop

Extended author information available on the last page of the article

Published online: 03 November 2025

1 Introduction

Risk-aware decision planning is essential in modern organizational processes: whether in project management, transportation, or manufacturing, delays in completion times can significantly affect costs, service levels, and customer satisfaction. The ability to efficiently evaluate risk of delay or worsening of the makespan has a profound impact on operations, enabling organizations to quote customer delivery lead times more reliably, manage resources effectively, and reduce the losses associated with last-minute adjustments and missed deadlines (Tao et al. 2018; Li et al. 2020).

In this paper, we investigate the use of a fast method addressing real-time risk assessment in *Job Shop Scheduling problems* (JSSs), a class of problems extensively studied for optimizing planning and scheduling of activities across multiple domains. In particular, we focus on complex JSS modeled as temporal activity networks (Elmaghraby 1977) with activities having incomplete and uncertain temporal attributes including releases, processing times, due dates, and sequence dependent setups. The presence of uncertainties in job durations make the exact solution of JSS especially challenging and enable the need of risk assessment. Following the seminal paper (Meloni and Pranzo 2020), we consider the case of *Integer Interval-valued activity Networks* (IINs) where durations can assume, as often arise in different practical applications, only equally possible integer values belonging to a known interval. Given an IIN representing a feasible solution for a scheduling problem, an advanced iterative counting procedure has been recently proposed (Meloni and Pranzo 2023) to evaluate quantile-based measures of risk for the makespan. This methodology gives as results theoretically proven lower and upper bounds for the quantile and the superquantile of the makespan at a prescribed level of probability. These quantile-based indices are commonly used risk measure also known as *Value-at-Risk* (VaR) and *Conditional-Value-at-Risk* (CVaR), respectively. Even if this procedure yields extremely good results on benchmarks from different contexts (Meloni et al. 2021; Meloni and Pranzo 2023), including the case of blocking job shop (Meloni et al. 2022), its complexity is pseudo-polynomial and grows quadratically with the network size and the amount of uncertainty affecting it Meloni and Pranzo (2020, 2023). Therefore, the existing pseudo-polynomial counting procedures may become a computational bottleneck in large and complex scenarios where real-time decision-making is required, preventing their application in real-time decision-making contexts. Accurate and rapid risk estimation is essential, as it can pave the way for the development of risk-aware algorithms that support real-time decision-making and enable the integration of risk measures into advanced optimization techniques, without becoming a computational bottleneck.

Following this need, the present study introduces an approach based on *Machine Learning* (ML) to estimate VaR and CVaR of the makespan in JSSs affected by uncertain activity durations. By training ML models on historical data, we aim to overcame the limitations of previous approaches, offering a methodology capable of providing instant risk estimates. Indeed, once properly trained and validated, ML models can generate predictions in negligible computing time, regardless of the problem's complexity, making them an ideal choice for real-time informed decision planning. These models can identify patterns and trends in makespan delays learning from collected

data, and are able to deliver robust estimates of risk measures also considering possible updates (Rossit et al. 2019). Moreover, job shop activity networks often have complex structures (Brucker and Knust 2013; Cheng et al. 2021; Xiong et al. 2022), and the relationship between activity durations and the makespan is generally nonlinear. This makes ML models particularly promising for accurately estimating the impact of uncertainty on the potential worsening of the makespan (Li et al. 2020; Raaymakers and Weijters 2003; Tremblet et al. 2023; Yamashiro and Nonaka 2021).

Specifically, the main contribution of the proposed paper can be summarized as follows:

- Identifying features that could effectively describe the activity network and its associated uncertainty;
- Developing and testing ensemble-based architectures to derive regression models capable of predicting quantile-based risk measures of the makespan;
- Comparing the performance of these learning architectures against a linear regressor to determine whether more complex models also capturing nonlinear relationship truly offer superior accuracy in predicting risk;
- Feature importance analysis to understand which characteristics of the activity network and uncertainty have the most significant impact on the risk predictions across the different models.

The remainder of the paper is organized as follows. Section 2 provides a review of the relevant related literature. Starting from an introduction of the activity networks and the adopted risk indices, Sect. 3 briefly describes the baseline exact procedure that is used to calculate risk measures of delay on collected instances in order to create an extensive benchmark set to train and test our ML models. Section 4 presents the learning-assisted method proposed in this paper, describing each ML model tested and how it can be obtained. Section 5 is devoted to the computational performance of these models (linear regressor included) on complex job shop instances and feature importance analysis. Conclusions are drawn in Sect. 6, along with the discussion of possible future research directions.

2 Literature review

JSS is one of the most studied problems in combinatorial optimization (Baker and Trietsch 2009; Brucker and Knust 2013; Framinan et al. 2014). This interest is keeping momentum due to the technological developments of recent years (Zhang et al. 2019), as the diffusion of cyber-physical systems and Industry 4.0 frameworks (Li et al. 2020; Zhang et al. 2021). In this sense, incorporating shop-floor data and forecasting models in scheduling methodologies is seen as a very promising approach for dealing with real JSSs and their complexity (Romero-Silva and Hernández-López 2020; Rossit et al. 2019).

Deterministic models are usually adopted to represent scheduling problems using performance measures expressed as a scalar function of completion times, with the makespan being the most common measure (Baker and Trietsch 2009; Sterna 2021).

Nevertheless these models do not accurately reflect the uncertainties that can arise in real-world scenarios. Therefore, while this approach may be appropriate for situations with limited uncertainties, it is not acceptable for all contexts (Baker and Trietsch 2009; Larsen and Pranzo 2019). One well know consequence of ignoring uncertainties in activity duration is the underestimation of the makespan, which has been extensively studied in both the services and manufacturing fields (Sarin et al. 2010; Liu and Urgo 2023).

Stochastic models can be used to address concerns related to operation times variability caused by changes in the environment or information updates. These models assume that operation times are random variables with known probability distribution functions (Baker and Trietsch 2009). However, such models can be computationally demanding which may not always be compatible with operational or real-time requirements (Sarin et al. 2010). In such cases, risk assessment is useful when decision makers are risk-averse and prefer solutions that perform satisfactorily in most cases (Elmaghraby 2005). The literature does not have a universally accepted risk measure, as each index has its own advantages and disadvantages (Bertsimas et al. 2004; Rockafeller and Royset 2013). The variance of the makespan is a commonly used risk measure, but it may not be appropriate for optimization problems in temporal networks (Sarin et al. 2010; Wiesemann 2012). One-sided quantile-based measures such as VaR and CVaR at a prescribed probability level may be preferred when the goal is to minimize the makespan downward deviations (Bertsimas et al. 2004; Elmaghraby 2005; Tao et al. 2018; Liu and Urgo 2023).

Although the use of quantile-based measures to take into account uncertainty in scheduling problems seems a promising topic, only few relevant papers appeared in the literature. They are mainly related to some specific application (Catalão et al. 2012; García-González et al. 2007; Kalinchenko et al. 2011; Sang et al. 2021), or limited to some specific scheduling environment (Atakan et al. 2017; Sarin et al. 2014; Urgo and Váncza 2019). Compared to other sectors, where this kind of risk measures are more used, their restricted diffusion in scheduling is perhaps due to the computational difficulties connected to the adopted approaches that are often based on scenarios and sampling methods (Larsen and Pranzo 2019; Sarin et al. 2010, 2014), which lead to limiting the analysis to scheduling cases with simple layout configurations, offering as output statistical estimates of the indicators of interest (Bruni et al. 2009; Liao et al. 2012; Sarin et al. 2014).

The computation of the expected shortfall of the makespan in scheduling problems represented by activity networks where a reliable prior knowledge of random processing times is not available has been recently addressed in Meloni and Pranzo (2020) and Meloni and Pranzo (2023) considering cases in which the processing times are integers and their equally possible values belong to known intervals. These cases can be considered as scheduling problems with incomplete or imperfect information (Zieliński 2005; Liu and Urgo 2023), assuming that the realized duration of an activity is only known when the execution of the activity is completed, and the uncertainty on these parameters implies that the objective value is also uncertain. These quantile-based risk assessment issues can arise in different real contexts including project management (Meloni and Pranzo 2020), industrial scheduling

(Meloni et al. 2022) and transportation (Meloni et al. 2021) often requiring fast and accurate computation.

Nevertheless, real-time, massive and repetitive evaluation tasks, especially involving large-size and highly uncertain instances may require procedures that can scale up offering good performances with low computational time variability. ML models could handle large datasets while exhibiting a better scaling behavior than pseudo-polynomial approaches, enabling for efficient evaluation of risk indices for the makespan even in more extensive and uncertain networks (Li et al. 2020; Rossit et al. 2019).

Machine learning tools have gained significant interest in recent years, and opened avenues for utilizing predictive analysis and extensive historical data for decision-making. The scheduling literature has witnessed diverse applications of machine learning. The earliest work in this area (Nakasuka and Yoshida 1992) focused on predicting the optimal dispatching rule for a given instance. Successively, Fransoo et al. (1994) pioneered the estimation of makespan, leading to the development of various approaches in subsequent research. Today, researchers continue to actively explore this field, tackling more complex environments such as job shop scheduling problems and learning abstract dispatching rules from simulation data (Jun et al. 2019; Li and Olafsson 2005; Shao et al. 2024), leading toward their use in presence of uncertainty (Hammami et al. 2024). These ML approaches can be viewed as pre-processing phases that enhance the performance of heuristic methods. Machine learning can be also used as a post-processing phase, where models predict whether rescheduling can improve a feasible schedule after it has been determined (Li et al. 2020; Rossit et al. 2019). Despite this thriving literature, there is a lack of specific work using ML models to predict the makespan value of a scheduling problem or the feasibility of a schedule before given deadlines. In the context of batch processing, several studies have focused on ML-based approaches to estimate the makespan. For instance, Raaymakers and Weijters (2003) investigated the use of regression and neural networks to predict makespan in a job shop environment. Similarly, Schneckenreither et al. (2020) explored this approach using neural networks and compared them to other methods through simulation. Both studies emphasized the robustness of such models and their potential in predicting lead time. Makespan prediction in batch processing is often employed to provide quick answers for order acceptance. The work (Tremblet et al. 2023) contributes to this field by further examining machine learning models for makespan estimation in a more complex production environment, utilizing a ML model based on random forest.

In this paper we aim to use ML methods to bring together the risk evaluation procedures and the complex job shop scheduling problems. Specifically we train ML methods to accurately estimate the risk associated to a schedule without running expensive risk assessment algorithm. To do so, we use one of the expensive risk assessment algorithm (Meloni and Pranzo 2020, 2023) to produce good and reliable data to train the ML predictors. Furthermore, we want to evaluate the accuracy of the prediction obtained by the ML models, identify which inputs are useful for the prediction, and thus which characteristics significantly impact the risk prediction across different models.

3 Job shop schedule as activity network with interval temporal values

Let us consider a schedule of a JSS under uncertainty obtained solving a deterministic version of the problem, in this paper we aim to quantify the risk on the makespan related to the uncertainty neglected during the solution process, using ML models. A crucial step to obtain a well performing ML model is to use correct and reliable data for its training phase. To obtain such data, we produce risk assessment using the approach detailed in Meloni and Pranzo (2020, 2023). Among the overall evaluations computed through this approach, we select only the ones in which we are able to estimate the risk with a precision of at least 99.95%. This straightforward approach is implemented to ensure that a correct training uses only reliable data.

For sake of completion, in the rest of this section, we proceed to give further details on how data for the training set has been produced. We start with the description of how a schedule with uncertain duration can be modeled using an activity network (Sect. 3.1), we continue with the definition of the risk measures adopted (Sect. 3.2) and conclude with the exact method used as a baseline algorithm to calculate them (Sect. 3.3).

3.1 Activity networks

To represent a solution under makespan performance measure, we adopt the IIN model, as introduced in Meloni and Pranzo (2020); Meloni et al. (2022); Meloni and Pranzo (2023). An IIN is a discrete network defined by the pair (G, T), where $G = (N, A)$ is a directed graph consisting of a set of nodes N and a set of arcs A, while $\mathbf{T}$ contains time intervals information. Note that in this paper, we use bold font to indicate uncertain quantities. Each node $i \in N$ corresponds to an event (such as the start of an operation), while each arc $(i, j) \in A$ defines a precedence relationship between events i and j with its associated interval of duration $[\underline{t_{ij}}, \overline{t_{ij}}]$. For arcs with deterministic duration we have that $\underline{t_{ij}} = \overline{t_{ij}}$. Finally, we define $\mathrm{A} \subseteq A$ as the subset of arcs of G with uncertain durations.

Given a deterministic solution, the corresponding IIN model can be easily constructed assuming the availability of prediction intervals for all time attributes involved. The resulting directed acyclic network is connected and contains single source and sink nodes.

The *configuration*$\mathcal{T}$ of a given IIN, denoted as Q, refers to the realization of a specific integer value t_{ij} from the time interval $\mathrm{T}_{ij} = [\underline{t_{ij}}, \overline{t_{ij}}]$ for each arc (i, j). For each IIN's configuration, $C_{max}(\mathcal{T})$ is the corresponding value of the makespan C_{max} that equals the length of the *critical path* in the network, i.e., the longest path between the source node in the graph, representing the beginning of the overall schedule, and the sink node, representing its end (Baker and Trietsch 2009; Brucker and Knust 2013). The total number of possible configurations of network Q is denoted as $|\mathrm{U}_{\mathcal{T}}(Q)|$ and can be calculated as

$$|\mathrm{U}_{\mathcal{T}}(Q)| = \prod_{\forall (i,j) \in A} (\overline{t_{ij}} - \underline{t_{ij}} + 1). \tag{1}$$

Since the activities with deterministic duration contribute with a unit value factor to the overall product, the IIN Q of a scheduling solution without uncertainties has only one configuration, resulting in $|\mathrm{U}_{\mathcal{T}}(Q)| = 1$.

3.2 Quantile-based risk indices for the makespan

Let us consider the objective function C_{max} in a specific IIN Q, which belongs to the range $[\underline{C_{max}}, \overline{C_{max}}]$. The value $\underline{C_{max}}$ represents the makespan when all durations in the network take their optimistic minimum values, while $\overline{C_{max}}$ corresponds to the pessimistic case where all durations assume their maximum values. The expected makespan, denoted $E(C_{max})$, is obtained when all durations are set to their expected values and can be calculated as $E(C_{max}) = \lfloor \frac{\overline{C_{max}} - \underline{C_{max}}}{2} \rfloor$.

To assess the risk of delay associated with C_{max}, we adopt two quantile-based risk indices formally described as follows:

- the *Value-at-Risk* of C_{max} at probability level α, denoted as $\mathrm{VaR}_{\alpha}(\mathrm{C}_{max})$, is the threshold value that is exceeded in the worst $(1-\alpha)100\%$ of cases:

$$\mathrm{VaR}_{\alpha}(\mathrm{C}_{max}) = inf\{t : prob(\mathrm{C}_{max} \leq t) \geq \alpha\} \tag{2}$$

- the *Conditional-Value-at-Risk* of C_{max} at probability level α, denoted as $\mathrm{CVaR}_{\alpha}(\mathrm{C}_{max})$, is the expected value of the worst $(1-\alpha)100\%$ of cases:

$$\mathrm{CVaR}_{\alpha}(\mathrm{C}_{max}) = E(\mathrm{C}_{max} | \mathrm{C}_{max} \geq \mathrm{VaR}_{\alpha}(\mathrm{C}_{max})). \tag{3}$$

Higher values of α can be chosen by decision makers who are more risk-averse. As α tends to its upper limit 1, $\mathrm{VaR}_{\alpha}(\mathrm{C}_{max})$, and $\mathrm{CVaR}_{\alpha}(\mathrm{C}_{max})$ tend to the worst (or pessimistic) case value $\overline{C_{max}}$. Conversely, when α tends to 0, $\mathrm{VaR}_{\alpha}(\mathrm{C}_{max})$ tends to the best (or optimistic) value $\underline{C_{max}}$, whereas $\mathrm{CVaR}_{\alpha}(\mathrm{C}_{max})$ tends to the expected value $E(\mathrm{C}_{max})$.

3.3 Baseline algorithmic method for risk assessment

The counting method introduced in Meloni and Pranzo (2020), then improved and extended in Meloni and Pranzo (2023), is designed to evaluate both VaR_{α} and CVaR_{α} for the makespan of an IIN. Given an IIN, the method aims at identifying the *counting target* CT_{α} at a given probability level α. Specifically, CT_{α} corresponds to the number of worst-case (i.e., the highest) configurations in $\mathrm{U}_{\mathcal{T}}(Q)$, defined as

$$CT_{\alpha} = \lceil (1-\alpha)|\mathrm{U}_{\mathcal{T}}(Q)| \rceil. \tag{4}$$

The method operates in three distinct phases, starting from the pessimistic makespan value $\overline{C_{max}}$ and iterating backwards to count all configurations that lead to each relevant successively smaller C_{max} value. Given a scheduling solution and a specified probability level α, in phase *(i)* the algorithm constructs the corresponding IIN Q. It then performs a preliminary analysis to calculate $|\mathrm{U}\mathcal{T}(Q)|$, determine the counting target $CT\alpha$, and set the pessimistic makespan level $\overline{L} = \overline{C_{max}}$ as the starting point for the evaluation.

In phase *(ii)*, the core counting process takes place, iterating over each specific level L corresponding to a possible value of makespan C_{max} until CT_α configurations have been identified and there is enough data to compute the desired quantile based risk measure (i.e., CVaR_α or VaR_α). At each level L, the counting is obtained exploiting all possible *exact reductions* applicable on the *critical subgraph* (i.e., the graph composed of all the arcs of Q having an associated *slack value* at most $(\overline{L} - L) - slack$ with respect to the critical path in Q) (Meloni and Pranzo 2023). When all the critical subgraphs used in the counting procedure completely reduce to a single arc through exact simplifications, the proposed method guarantees the exact computation of VaR_α and CVaR_α. Otherwise, the method provides heuristic evaluations yielding lower and upper bounds of the risk measures, denoted as $LB_{\mathrm{VaR}_\alpha}(\mathrm{C}_{max})$, $UB_{\mathrm{VaR}_\alpha}(\mathrm{C}_{max})$ and $LB_{\mathrm{CVaR}_\alpha}(\mathrm{C}_{max})$, $UB_{\mathrm{CVaR}_\alpha}(\mathrm{C}_{max})$, respectively.

The final phase *(iii)* computes the required results for both risk measures certifying their exactness or averaging the corresponding upper and lower bounds and providing a bound on the maximum relative error.

In this study, we use the algorithmic configuration denoted as Sl/Adv/mS, which has proven to be effective for risk assessment in scheduling solutions (Meloni and Pranzo 2023). This procedure has a *pseudo-polynomial* complexity $O(\Gamma^2|A|^2)$, where $|A|$ is the number of arcs in the IIN, and $\Gamma = \overline{C_{max}} - \underline{C_{max}} + 1$ represents the amount of uncertainty in the network. This procedure is quite efficient on several real-world instances but, given its pseudo-polynomial complexity, its computation times can grow fast with both the size of the network and the amount of uncertainty affecting it Meloni et al. (2022). However, the availability of this baseline algorithm – capable of giving high quality results even if not able to scale satisfactorily – enables the training of ML models to perform the desired risk assessment with consistently low computation times, while preserving the quality of the results.

4 Learning-assisted method

To bridge the gap between the efficiency required for practical applications and the unreliable computational times of the baseline algorithm due to scalability issues, we introduce a machine learning assisted approach. The idea is to leverage the power of ML to learn the relationship between the activity network representing a feasible schedule and the risk indices VaR_α and CVaR_α associated with the schedule.

More precisely, we make use of supervised learning, i.e., an ML approach in which a model tries to learn a function that maps input features describing the activ-

ity network to their corresponding known target outputs, i.e., the risk measures we aim to predict.

In Sects. 4.1 and 4.2, we introduce the tested models and specify the input features and the targets outputs, respectively. Then, in Sect. 4.3, we outline the whole ML process by which the models' structure and weights are obtained, including data preprocessing, model training, and evaluation. Indeed, the use of any ML model requires careful consideration of appropriate training data, model selection, feature engineering, and validation procedures to ensure reliable and accurate results (Hastie et al. 2009; James et al. 2013).

4.1 Machine learning models

In this study, we consider four distinct regression models:

- *The Multiple Linear Regressor* (MLR) (Angrist and Pischke 2009) corresponds to the best-fitting hyperplane minimizing the sum of squared errors between the observed outputs and the predicted outputs. Specifically, it models the linear relationship between a dependent variable and multiple independent variables. MLR is easy to calculate and interpret, and suitable for scenarios where the relationships between dependent and independent variables are primarily linear.
- The *Random Forest*-based regressor (RF) (Breiman 2001) is an ensemble learning method that constructs a multitude of decision trees during training and outputs the mean prediction of the individual trees. By averaging multiple trees trained on a random subset of the data and features, it reduces overfitting enhancing the model's generalization capability.
- *Extreme Gradient Boosting* (XGB) (Chen and Guestrin 2016) and *Light Gradient Boosting Machine* (LGBM) (Ke et al. 2017) are two advanced implementations of the gradient boosting algorithm designed to improve upon its efficiency and performance. Gradient boosting algorithm (Friedman 2001, 2002) works by iteratively training models where each new model attempts to correct the errors made by previous ones. It is highly flexible and can handle complex nonlinear relationships. XGB is an optimized implementation of gradient boosting that can offer superior performance in terms of accuracy and scalability, thanks to regularization techniques controlling overfitting and efficient parallel processing capabilities. Whereas, LGBM is a gradient boosting framework designed for speed and memory efficiency, which gives a great advantage when dealing with large-scale data. It also provides better accuracy thanks to its leaf-splitting rule, resulting in more complex trees that can capture more intricate patterns in the data.

Except for MLR, which is a statistical technique, all the other are ensemble learning methods that combine multiple weak models (in our case, regression trees), to create a strong predictive model.

These models were selected to span a spectrum of modeling complexities: from the interpretable linear model to state-of-the-art ensemble methods, which are known to outperform other nonlinear approaches on tabular data (see e.g., Grinsztajn et al.

(2022)), ensuring a balanced trade-off between model complexity, predictive accuracy and computational efficiency.

4.2 Input features and targets

In order to train a supervised ML model, we need to define the input features providing the necessary information for the model to make accurate predictions. The features we employ, which attempt to describe the activity network size and complexity, are: (*i*) the probability level α, (*ii*) the expected makespan $E(C_{max})^{mapped}$ expressed as a [0,1] displacement between $\underline{C_{max}}$ and $\overline{C_{max}}$, (*iii*) the total number of configurations $|\mathrm{U}_{\mathcal{T}}(Q)|$ (1), *(iv)* the number of network arcs $|A|$, (*v*) the number of network nodes $|N|$, (*vi*) the number of uncertain network arcs $|\mathrm{A}|$, (*vii*) the difference between pessimistic and optimistic makespan denoted as Γ, (*viii*) the number of arcs having a slack time value greater than Γ denoted as Γ-*slack* (i.e., the arcs that will never belong to the critical subgraph under any circumstance), (*ix*) the arcs always belonging to the critical subgraph denoted as 0-*slack*, and (*x*) the uncertain ratio distribution $\xi(0.1)$ corresponding to the number of arcs belonging to the first decile of the slack distribution (i.e., the arcs having slack smaller than 0.1 times Γ). In other words, $\xi(0.1)$ is the number of highly critical arcs. To get $E(C_{max})^{mapped}$ we do the following operation

$$E(C_{max})^{mapped} = \frac{E(C_{max}) - \underline{C_{max}}}{\overline{C_{max}} - \underline{C_{max}}} \tag{5}$$

that let us eliminate $\underline{C_{max}}$ and $\overline{C_{max}}$ from the input features, as they resulted to be highly correlated to $E(C_{max})$.

The targets are VaR^{mapped} and CVaR^{mapped} expressed again as a displacement between their feasible ranges. In particular, VaR can be mapped between $\underline{C_{max}}$ and $\overline{C_{max}}$ as follows

$$\mathrm{VaR}^{mapped} = \frac{\mathrm{VaR} - \underline{C_{max}}}{\overline{C_{max}} - \underline{C_{max}}} \tag{6}$$

whereas CVaR can be mapped between the lowest value it can assume ($E(C_{max})$) and the highest ($\overline{C_{max}}$) as follows

$$\mathrm{CVaR}^{mapped} = \frac{\mathrm{CVaR} - E(C_{max})}{\overline{C_{max}} - E(C_{max})}. \tag{7}$$

4.3 Machine learning pipeline

In this section, we describe the pipeline that can be followed to obtain each predictive model: (*i*) dataset generation, (*ii*) shuffling and division of the data into training and test sets, (*iii*) scaling of the data (if required by the regressor), *(iv)* selection of the hyperparameters (if required by the regressor), (*v*) model training and testing,

(*vi*) calculation of the prediction interval. We analyze each step individually in the following subsections.

4.3.1 Dataset generation

In the initial phase of our pipeline, we create the data needed to train and then test the models, starting from well-established deterministic blocking JSS instances enriched by the additional realistic constraints (Meloni et al. 2022) and injected with different types and entities of uncertainty. From these initial instances, we generate feasible deterministic schedules adopting optimistic duration value t_{ij} for all JSS operations. We then represent each JSS solution as an IIN by injecting back uncertainty information to the deterministic schedules. Note that each solution in the dataset is unique. Finally, we evaluate each IIN with the baseline algorithm described in Sect. 3.3 to calculate VaR and CVaR so as to obtain inputs and target values for ML usage.

4.3.2 Shuffling and division of the data

The subsequent step involves the shuffling and partitioning of the dataset into two distinct sets: the training set and the test set.

Shuffling the data is a crucial step as it helps mitigate the risk of any potential bias in the order of the data. By shuffling the data, we ensure that our training and test sets are representative of the overall distribution and do not reflect any particular sequence from the original dataset. Then, we partition the data into training and test sets. This is a commonly used strategy in machine learning to train a model on known data (i.e., the data belonging to the training set) and then evaluate the predicted outcomes on unseen data (i.e., the data belonging to the test set). By evaluating the model's predictions against the observed outcomes in the test set, we can measure the model's ability to generalize beyond the data it is trained on.

4.3.3 Scaling of the data

Data scaling is a preprocessing step where all the features are transformed to a common scale to prevent features with larger scales from dominating those with smaller scales (see e.g., Aggarwal (2015)) and to ensure that each feature is given equal importance. This also happens in our dataset, where, for instance, features such as α and $E(C_{max})^{mapped}$ 5 take values that are less than or equal to 1, whereas other input features such as $|A|$ and $|N|$ have an order of magnitude of at least 10^2.

However, it is important to note that not all machine learning algorithms require feature scaling. Algorithms like decision trees and, by consequence, tree-based ensemble methods (such as RF, XGB, and LGBM) are scale-invariant. This means they partition the data based on the structure of the data rather than the values of the data and do not necessarily require feature scaling. On the other hand, for linear regression models, where the scale of variables directly influences the model's coefficients, scaling is crucial.

4.3.4 Selection of the hyperparameters

The ML models we use, except for MLR, require the tuning of the hyperparameters. The hyperparameters include aspects that define both the model architecture and the training process, hence, their selection can significantly influence the performance of the model, resulting in improved predictive performance and generalization.

The strategy we use to select the best hyperparameters is a *k-fold cross-validation*, which is a widely-used resampling procedure aimed at assessing the predictive performance of machine learning models (Arlot and Celisse 2010). The idea is to split the training set into *k* equally sized folds. We then iteratively train the model *k* times, each time using *k*-1 folds as the training set and the remaining fold as the validation set. To determine the best hyperparameter configuration for our model, we use the random search technique in conjunction with the *k*-fold cross-validation. Instead of exhaustively evaluating all possible combinations of hyperparameters, random search randomly samples combinations from predefined ranges. This enables a more efficient exploration of the hyperparameter space, especially in high-dimensional scenarios (Bergstra and Bengio 2012).

In each of the *k* iterations, we train the model using one random combination of hyperparameters and evaluate its performance on the validation set using a chosen evaluation metric, which in our case is the root mean square error as it penalizes models making larger errors. This cross-validation process is repeated for various hyperparameter configurations, and the configuration that yields the minimum average root mean square error across all folds is chosen.

4.3.5 Model training and testing

After selecting the optimal hyperparameters, the model is trained on the entire training dataset and its performance is evaluated on training and test data. The evaluation of the accuracy of the predictions in the training data is used to assess if there have been some underfitting or overfitting phenomena. The evaluation of the accuracy of the predictions on test data instead is used to assess the capability of the model to generalize.

Taken the *i*-th sample of a dataset containing *n* samples, we denote by y_i^j and $\hat{y}_i^j$ the observed and the predicted value of the *j*-th output at sample *i*-th, where $i = \{1, \ldots, n\}$ and $j \in \{1, 2\}$ ($j = 1$ corresponds to VaR, $j = 2$ corresponds to CVaR). The metrics we employ to evaluate the prediction error on each output *j* are:

- *Mean Absolute Error* (MAE): the average of the absolute differences between the predicted and observed values

$$\frac{\sum_{i=1}^{n} |y_i^j - \hat{y}_i^j|}{n} \tag{8}$$

- *Mean Absolute Percentage Error* (MAPE): the average percentage error between the predicted and observed values

$$100 \frac{\sum_{i=1}^{n} \frac{|y_i^j - \hat{y}_i^j|}{y_i^j}}{n} \tag{9}$$

- *Root Mean Square Error* (RMSE): the square root of the average of squared differences between the predicted and observed values. It assigns a heavier weight to larger errors

$$\sqrt{\frac{\sum_{i=1}^{n} (y_i^j - \hat{y}_i^j)^2}{n}} \tag{10}$$

- *Maximum Relative Error* (MaxRE): the largest relative error made by the model. It is calculated as the maximum value of the absolute difference between the predicted and observed value, divided by the observed value. It provides a measure of the worst-case error between the predicted and observed values

$$\max_{i=1,\ldots,n} \frac{|y_i^j - \hat{y}_i^j|}{y_i^j}. \tag{11}$$

Each error metric provides a different insight into the model performance to the decision maker and can be collectively used for a comprehensive evaluation.

In line with our goal of finding a procedure that can be reliable on timing, other interesting metrics to evaluate are related to the computational time needed to get a prediction. We consider:

- *Training Time* (TrainT): this metric computes the total computational time, expressed in seconds, needed by the model for the training phase on the testing set of data
- *Testing Time* (TestT): this metric computes the total computational time, expressed in seconds, needed by the model to predict the output of a testing set of data.

4.3.6 Prediction interval

In order to have a more comprehensive picture of the predictive capability of each model, we use a *bootstrapping approach* to calculate the prediction intervals. Prediction intervals provide an estimate of the interval in which the prediction falls with a certain given probability. Having a prediction interval gives a more robust and reliable view of each model and its uncertainty in the prediction, and more information to the decision-maker than the solely predicted value.

To calculate these intervals we adopt a bootstrapping approach (Angelopoulos and Bates 2023; Vovk et al. 2005), which is a statistical method for estimating the variability of a sample statistic by resampling with replacement from the original dataset. The procedure consists of a generation of a large number of bootstrap samples, that are used to generate a distribution of predictions. Then, to identify the prediction interval, the lower and upper percentile of this distribution are used.

5 Results

We now discuss the results obtained with the ML models introduced in this paper to perform risk assessment. In particular, Sect. 5.1 describes the experimental settings we used to get the dataset needed to train the ML models. In Sect. 5.2, we report the results on the feature importance of each regression model, while in Sect. 5.3 we provide their forecasting performance.

5.1 Experimental Settings

As mentioned in Sect. 4.3.1, the ML pipeline starts from the generation of the dataset needed to train (and then test) the regression models. To generate this dataset, we considered 2 145 JSS different solutions obtained with the following procedure. Given 58 well established deterministic blocking JSS instances (Adams et al. 1988; Applegate and Cook 1991; Fisher and Thompson 1963; Lawrence 1984), we enriched them by the additional realistic constraints as described in Meloni et al. (2022). Then, starting from these instances, we generated 2 145 feasible deterministic schedules using the algorithm described in D'Ariano et al. (2007) and adopting optimistic duration value $\underline{t_{ij}}$ for all JSS operations. The respective IINs are built following the procedure presented in Meloni et al. (2022) by injecting uncertainty over a subset of precedence relations of the activity network. Uncertainty has been added to the job precedence subset, the machine precedence subset, or both. Only a given percentage η of arcs in a subset is uncertain, with $\eta \in \{10\%, 15\%, 20\%, 25\%\}$. Uncertain arcs have different uncertainty magnitudes ranging from 10% to 25% of arc duration. In particular, the interval duration of uncertain arcs $\mathrm{T}_{ij} = [\underline{t_{ij}}, \overline{t_{ij}}]$ is build as $\overline{t_{ij}} = \delta \underline{t_{ij}}$, with $\delta \in \{1.10, 1.15, 1.20, 1.25\}$. Overall, 48 different IINs spawn from each JSS solution.

Regarding the probability level α of the risk measures CVaR_α and VaR_α, three different values were used, namely $\alpha \in \{0.99, 0.95, 0.90\}$, which represent different sensitivities of the decision maker regarding the risk of worsening of the makespan. We thus performed 308 880 risk assessments using the baseline algorithm briefly described in Sect. 3.3.

To get the final dataset, we selected among the 308 880 instances only the ones where the risk assessment was either solved to optimality (i.e., $UB_{\mathrm{VaR}_\alpha}(\mathrm{C}_{max}) = LB_{\mathrm{VaR}_\alpha}(\mathrm{C}_{max})$ and $UB_{\mathrm{CVaR}_\alpha}(\mathrm{C}_{max}) = LB_{\mathrm{CVaR}_\alpha}(\mathrm{C}_{max})$) or near-optimality (with a maximum gap for both $\mathrm{VaR}_\alpha(\mathrm{C}_{max})$ (2) and $\mathrm{CVaR}_\alpha(\mathrm{C}_{max})$ (3) of less than 0.5‰), resulting in a final dataset of 214 941 elements. This selection is essential to ensure that the ML models learn to predict the exact risk measures, rather than their lower and upper bounds mimicking the results of the risk procedure.

The ML models have been implemented using the Python library *scikit-learn* (Pedregosa et al. 2011), with the exception of XGB, which has been implemented using the Python library *xgboost* (Chen and Guestrin 2016). As mentioned in Sect. 4.3.2, we split our dataset into a training and a test set using a 70/30 proportion. Permutation importance was done using the Python library *scikit-learn* (Pedregosa et al. 2011), whereas SHAP values are calculated using the Python library *shap* (Lun-

dberg and Lee 2017). To calculate the prediction intervals, the Python library *MAPIE* (Taquet et al. 2022) has been used and the bootstrapping technique selected among the ones offered by this library is CV+, which is computationally more efficient than the classic leave-one-out approach.

Each model architecture (except MLR, which does not contain hyperparameters) was chosen after a k-fold cross-validation with $k = 5$ and 500 random combinations of hyperparameters. Prior to this, preliminary manual tests were conducted to define a meaningful range for each hyperparameter to test. The ranges manually selected for the cross-validation process and the best values of each hyperparameter identified after the cross-validation process are reported in Table 1. For each model, the best configuration of hyperparameters is the one yielding the minimum RMSE (10) among all combinations tested.

The hyperparameters listed in Table 1 include all key hyperparameters known to significantly affect model performance. We report in the table the parameter names used in the open-source libraries employed in this study. We would like to point out that different ML models require different sets of hyperparameters. These hyperparameters can be grouped into three main categories:

1. The first group includes structure-related hyperparameters that define the architecture and complexity of the models:
   - `n_estimators` specifies the number of trees in the ensemble;
   - `max_depth` sets the maximum depth allowed for each tree and controls model complexity;
   - `num_leaves` defines the maximum number of leaves per tree and has a major impact on the model expressiveness;
   - `min_samples_leaf` and `min_samples_split` indicate the minimum number of samples required to form a leaf and to split an internal node respectively;
   - `min_child_samples` specifies the minimum number of data points required in a leaf and helps preventing overfitting;
   - `subsample` controls the fraction of training data used to build each tree improving generalization.
2. The second group regards the regularization hyperparameters and includes `reg_alpha` and `reg_lambda`, used to reduce overfitting by penalizing model complexity.
3. In the third group we include only the `learning_rate`, which determines the weight assigned to each new tree added during the training process, controlling how quickly the model adapts to the residual errors of previous iterations.

All computations have been run on an Intel Xeon CPU E5-2699 v4 2.2 GHz processor with 1.5 TB RAM.

Table 1 Hyperparameter ranges explored and best values selected for each model architecture (RF, XGB or LGBM) and feature to predict (VaR or CVaR)

	Hyperparameter	Range explored	Best value selected
RF	`n_estimators`	random integer in [100, 3000]	VaR: 937 - CVaR: 937
	`max_depth`	random integer in [5, 40]	VaR: 35 - CVaR: 35
	`min_samples_split`	random integer in [5, 40]	VaR: 6 - CVaR: 6
	`min_samples_leaf`	random integer in [3, 40]	VaR: 3 - CVaR: 3
XGB	`learning_rate`	random in [0.001, 1]	VaR: 0.07 - CVaR: 0.05
	`n_estimators`	random integer in [100, 3000]	VaR: 2111 - CVaR: 2740
	`subsample`	random in [0.7, 0.95]	VaR: 0.95 - CVaR: 0.85
	`max_depth`	random integer in [5, 40]	VaR: 8 - CVaR: 9
	`reg_alpha`	random in [0.0, 0.3]	VaR: 0.0 - CVaR: 0.0
	`reg_lambda`	random in [0.0, 0.3]	VaR: 0.3 - CVaR: 0.05
LGBM	`learning_rate`	random in [0.001, 1]	VaR: 0.05 - CVaR: 0.1
	`n_estimators`	random integer in [100, 3000]	VaR: 2949 - CVaR: 2841
	`subsample`	random in [0.7, 0.95]	VaR: 0.9 - CVaR: 0.9
	`max_depth`	random integer in [5, 40]	VaR: 27 - CVaR: 38
	`num_leaves`	random integer in [10, 500]	VaR: 355 - CVaR: 244
	`min_child_samples`	random integer in Applegate and Cook (1991); Meloni and Pranzo (2020)	VaR: 10 - CVaR: 5
	`reg_alpha`	random in [0.0, 0.3]	VaR: 0.0 - CVaR: 0.0
	`reg_lambda`	random in [0.0, 0.3]	VaR: 0.1 - CVaR: 0.0

5.2 Feature importance

To evaluate the effects and the relevance of each input feature on the regression models, we compute both Permutation Importance (PI) and SHapley Additive exPlanations (SHAP values) on the test set.

PI measures the decrease in model accuracy on the test set when the values of a particular input feature are randomly shuffled. In the PI plot, features are listed on the y-axis, while the x-axis shows the corresponding decrease in accuracy, indicating the feature's impact on model performance. A larger decrease in accuracy suggests that the feature is more important, as the model is more sensitive to alterations in its values. Conversely, a feature with minimal impact on accuracy is considered less important. The order of features in the PI plot reflects their importance.

SHAP values provide a detailed overview of the contribution of each input feature to the model predictions. In the SHAP summary plot, features are displayed on the y-axis, ranked by their overall importance, while the x-axis represents the SHAP value. This value quantifies the feature's effect on the prediction compared to a baseline (i.e., the prediction made by a model without that feature). Positive SHAP values indicate that the feature increases the predicted value, while negative SHAP values suggest a decrease. Each point in the plot corresponds to a SHAP value for a feature-instance pair. The color of the points reflects the feature's value, with red indicating higher values and blue indicating lower values. The shape and spread of the SHAP values across the x-axis provide insight into the magnitude of the feature's effect, with wider spreads indicating a greater influence on the model's predictions. Additionally, the color helps reveal the relationship between feature values and their impact on the output. For example, a distinct color pattern (red or blue) reveals how high or low values of the feature influence the prediction.

We now analyze PI and SHAP summary plots of each regressor separately and then make an overall discussion of these results in Sect. 5.2.5.

5.2.1 Multi linear regressor

The analysis of PI on the test set (Fig. 1) shows that removing $|A|$ and Γ-slack produces a high decrease in the accuracy of the prediction, while the other features have, in general, a more limited impact. We observe how on the x-axis PI for VaR exhibits higher values than that for CVaR, showing how the removal of these critical features in the VaR model causes a greater decrease in the forecast accuracy compared to the CVaR model.

The SHAP analysis reported in Fig. 2 identifies $|A|$ and Γ-slack as the most relevant input features for both VaR and CVaR models, while the remaining input features have a residual effect, confirming the results obtained with the PI analysis. We can also note that high values of $|A|$ positively correlate with higher outputs showing that larger instances have predicted VaR and CVaR values closer to $\overline{C_{max}}$ than $\underline{C_{max}}$ (for VaR) or $E(C_{max})$ (for CVaR). Additionally, the initial JSS instances, for their own nature, can be clustered given the number of jobs and machines considered, which is directly reflected in the value $|A|$ of our IIN and shown in the segmented distribution of the SHAP. This behaviour is visible not just in $|A|$, but also very strongly in α and,

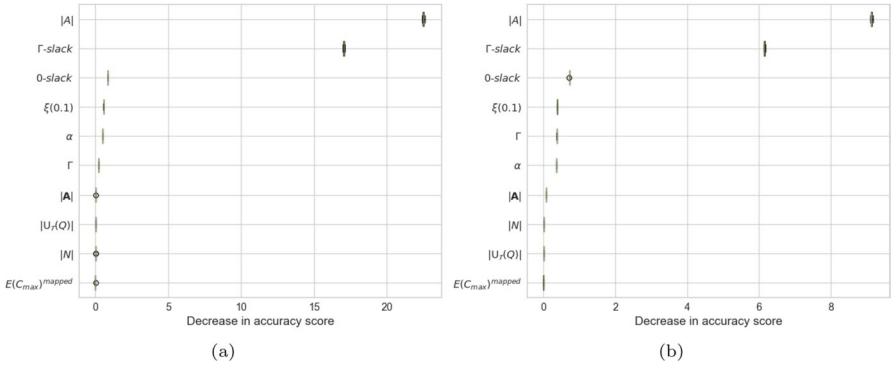

Fig. 1 PI analysis on MLR for **a** VaR and **b** CVaR

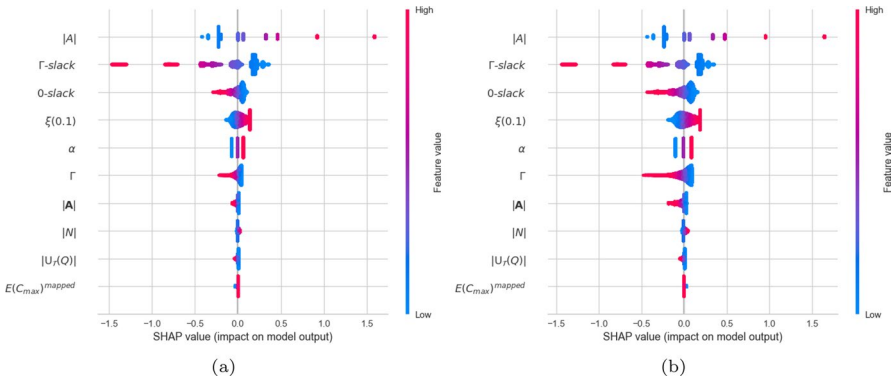

Fig. 2 SHAP values on MLR for **a** VaR and **b** CVaR

less strongly, in Γ-slack. In this case, high values of the Γ-slack correlate with lower values of the output showing that the presence of a large number of critical arcs (i.e., a smaller Γ-slack value) tends to drive the VaR towards $\overline{C_{max}}$ and the CVaR towards $E(C_{max})$.

5.2.2 Random forest

In Fig. 3, we show the results of PI on the test set for the RF models predicting VaR and CVaR. We observe similar findings for the two risk measures: the removal of Γ has the highest impact on the model accuracy, followed by the removal of α and $|\mathrm{A}|$, whereas the other features show marginal impact, with the size of the activity network ($|N|$ and $|A|$) being the least relevant. We also observe how the decrease in accuracy score is an order of magnitude smaller than the MLR predictor, showing how tree-based models have in general a better accuracy than MLR (indeed this happens also for XGB and LGBM).

The SHAP values confirm the results of PI since they identify Γ as the most important input feature, followed by α and $|\mathrm{A}|$ both for VaR and CVaR. They also reveal

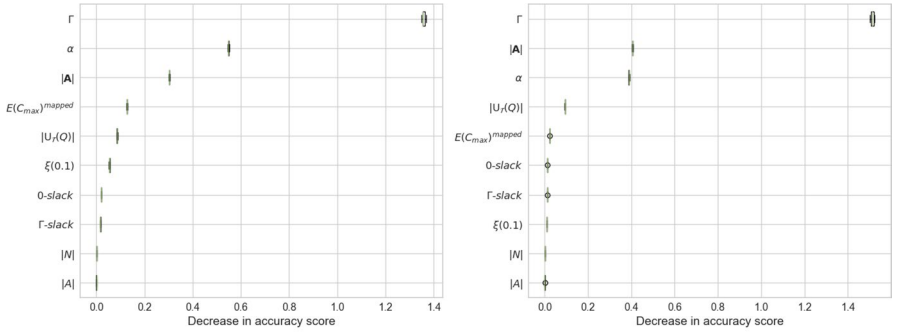

Fig. 3 PI analysis on RF for **a** VaR and **b** CVaR

how high values of Γ and $|\mathrm{A}|$ correlate with lower values of the predicted output. We can observe how the α values are grouped into 3 regions mapping the 0.9, 0.95 and 0.99 α values. However, their impact on the model output is less strictly correlated to the specific value as it is for the MLR (which instead show three very distinct lines).

5.2.3 Extreme gradient boosting

In Fig. 5, we show the results of PI on the XGB predictor for both VaR and CVaR. The removal of Γ and $|\mathrm{A}|$ have the highest impact on the model accuracy, followed by the removal of $|\mathrm{U}_{\mathcal{T}}(Q)|$ and α. The other features show marginal impact.

According to the SHAP values (Fig. 6), the most important features are Γ, $|\mathrm{A}|$, $|\mathrm{U}_{\mathcal{T}}(Q)|$ and α for both VaR and CVaR, confirming the results of the PI analysis. The other features are less and less relevant till $|N|$ and $|A|$ which have only marginal effects on the XGB algorithm, shown by their position in the ranking and a shape very close to 0. When comparing the SHAP values for VaR and CVaR (Fig. 6a and b) we observe that the effects of Γ and $|\mathrm{A}|$ are remarkably more marked for the CVaR than for the VaR risk measure.

5.2.4 Light gradient boosting machine

The PI on LGBM predictor is shown in Fig. 7 for both VaR and CVaR. For LGBM, Γ is the only feature having a strong impact, followed by $|\mathrm{A}|$, α and $|\mathrm{U}_{\mathcal{T}}(Q)|$ with smaller effects. For the CVaR, we observe how α has the same importance of $|\mathrm{U}_{\mathcal{T}}(Q)|$, while for VaR it is more important. The other features show marginal impact.

From Fig. 8, we can observe very similar results to XGB Fig. 6. Also in this case, the most important features according to the SHAP analysis are Γ and $|\mathrm{A}|$, followed by α and $|\mathrm{U}_{\mathcal{T}}(Q)|$, for both VaR and CVaR. The main difference with XGB is the swap in importance between $|\mathrm{U}_{\mathcal{T}}(Q)|$ and α.

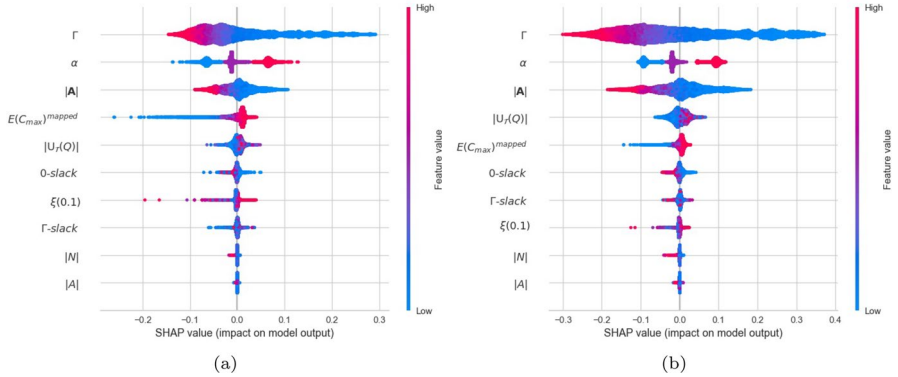

Fig. 4 SHAP values on RF for **a** VaR and **b** CVaR

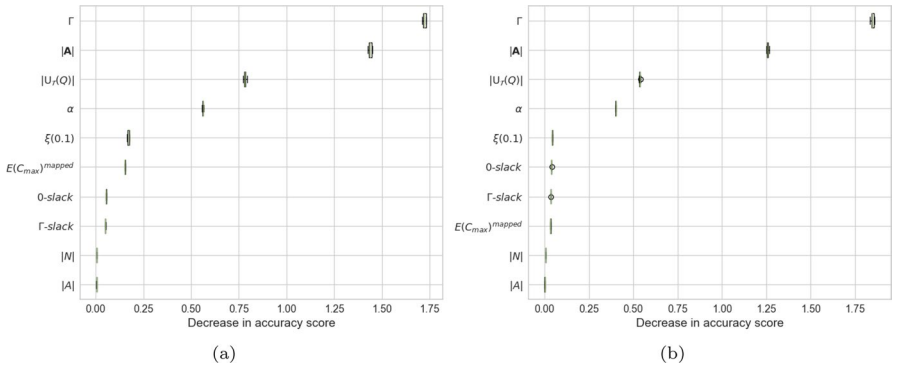

Fig. 5 PI analysis on XGB for **a** VaR and **b** CVaR

5.2.5 Discussion

From the previous discussions on feature importance for each forecasting model, several observations can be done. First of all, we categorize the input features into three groups: (*i*) features based on deterministic instance data ($|A|$ and $|N|$), (*ii*) features based on uncertain instance data ($|\mathrm{A}|$, $|\mathrm{U}_{\mathcal{T}}(Q)|$, α), and (*iii*) features related to the distribution of the uncertainties (Γ, Γ-slack, $0-$slack, $\xi(0.1)$, $E(Cmax)^{mapped}$). Among these, MLR is the only predictor that relies on a deterministic feature from the first group ($|A|$), whereas, $|N|$ appears to be not relevant for any of the models. When comparing feature importance across different predictors, it is evident that MLR selects input features that differ significantly from those chosen by tree-based models. This discrepancy could depend on MLR's limitations in capturing nonlinear relationships between inputs and outputs.

The SHAP plots (Figs. 2, 4, 6 and 8) provide further insights:

- Features with high values on the right end of the spectrum show a direct correlation with higher prediction values, while those with high values on the left

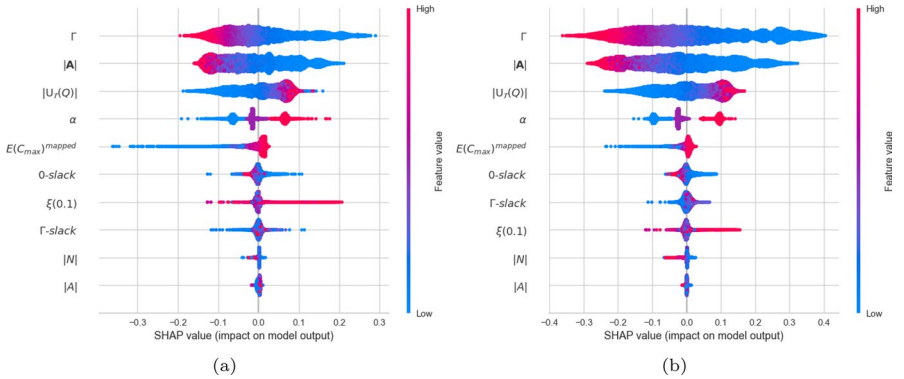

Fig. 6 SHAP values on XGB for **a** VaR and **b** CVaR

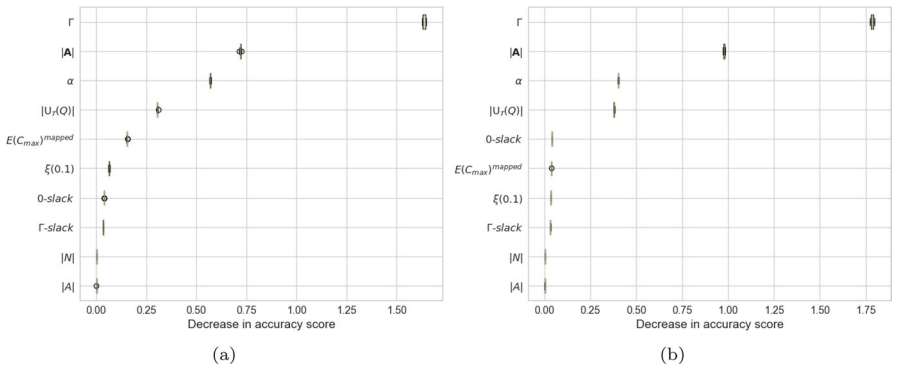

Fig. 7 PI analysis on LGBM for **a** VaR and **b** CVaR

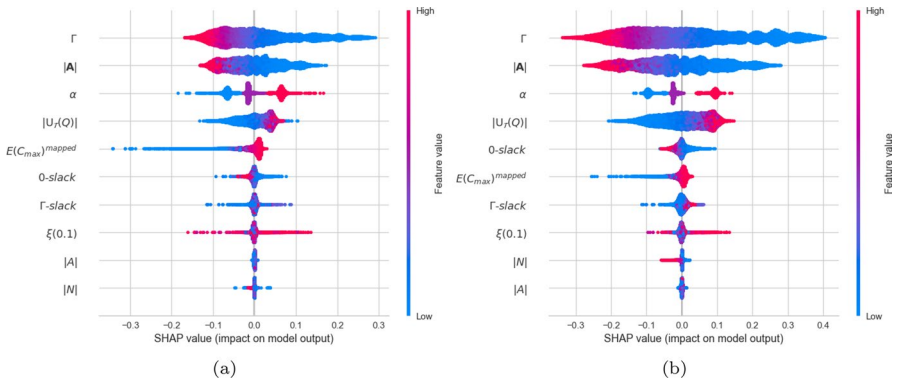

Fig. 8 SHAP values on LGBM for **a** VaR and **b** CVaR

end indicate an inverse correlation. It is important to recall that the counting approach used in the risk assessment to produce our dataset, starts from the pessimistic $\overline{C_{max}}$ value, mapped to an output of 1, and decreases until the counting target is reached. Consequently, low output values are closer to the lower end of the mapped prediction interval (i.e., 0), corresponding to $\underline{C_{max}}$ for VaR and $E(C_{max})$ for the CVaR. Assuming the learned inferences are correct, a direct correlation implies that instances yielding outputs near 1 are easier to solve using the counting approach, as they do not require to go deep with the counting algorithm. Conversely, an inverse correlation and lower output values suggests instances that are more challenging for the counting approach.

- A feature with a fragmented or non-continuous shape in the SHAP plots indicates the presence of data clusters in the values of that feature. For example, α exhibits three distinct regions in all the SHAP figures, corresponding to the 0.9, 0.95, and 0.99 values. Among our regression models, MLR is the one that most strongly reflects this clustering effect, as the most important features for this model also exhibit this data structure (see e.g., Γ-slack and $|A|$).
- Finally, when the lower and upper extremes of the SHAP value range are close to each other, it implies that varying the feature value has little effect on the predicted output. This suggests that the feature is not particularly relevant, as is often the case with $|N|$.

5.3 Model performance

The accuracy of the prediction obtained by the different ML models developed and the computation time needed to train each model and get the predictions are key factors to consider when selecting the most suitable prediction method.

In Tables 2 and 3 we report the accuracy results on the models predicting VaR and CVaR respectively. The tables show the same structure with each column referring to a different predictor (MLR, RF, XGB, and LGBM). Rows 1 to 4 report different error measures introduced in Sect. 4.3.5 to evaluate the quality of the prediction, i.e., Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), Root Mean Square Error (RMSE) and Maximum Relative Error over the test sets (MaxRE). Rows 5 and 6 report, respectively, the percentage of predictions falling outside the

Table 2 Results on VaR

	MLR	RF	XGB	LGBM
Results *before* feasible range adjustements				
MAE	1.182	0.462	0.341	0.314
MAPE (%)	0.062	0.026	0.019	0.017
RMSE	1.922	0.738	0.492	0.461
MaxRE	0.012	0.019	0.012	0.012
Out Int (%)	1.371	0.000	3.609	3.162
Max Out	0.577	0.000	0.938	0.732
Results *after* feasible range adjustements				
MAE	1.180	0.462	0.338	0.312
MAPE (%)	0.062	0.026	0.018	0.017
RMSE	1.922	0.738	0.492	0.461
MaxRE	0.012	0.019	0.012	0.012

Table 3 Results on CVaR

	MLR	RF	XGB	LGBM
Results *before* feasible range adjustments				
MAE	1.062	0.405	0.184	0.178
MAPE (%)	0.054	0.022	0.010	0.010
RMSE	1.799	0.682	0.337	0.334
MaxRE	0.014	0.012	0.006	0.006
Out Int (%)	2.965	0	2.905	2.371
Max Out	0.532	0	0.830	0.673
Results *after* feasible range adjustments				
MAE	1.059	0.405	0.183	0.177
MAPE (%)	0.054	0.022	0.010	0.010
RMSE	1.799	0.682	0.337	0.334
MaxRE	0.014	0.012	0.006	0.006

prediction interval ($[\underline{C_{max}}, \overline{C_{max}}]$) (Out Int (%)) and the maximum value outside the interval (Max Out). In the second part of the tables we show the errors when the feasible range adjustment procedure has been applied to force all the predictions to fall within the $[\underline{C_{max}}, \overline{C_{max}}]$ interval.

Table 2 shows that MLR produces the least accurate results - even if the quality of the predictions is already quite good - with a MAE above 1, but in relative terms the MAPE is only 0.06%. RF drastically improves over the MLR, and even better results are obtained with the most advanced predictors XGB and LGBM. XGB and LGBM also exhibit a higher percentage of predictions falling outside the confidence interval, around 3%. However, their prediction is the most accurate, with a MAPE below 0.02%. After applying the range adjustment, we observe a further marginal improvement in the MAE measure, especially for XGB and LGBM.

In Table 3, MLR again shows the poorest performance, while the best results are achieved by the two gradient boosting algorithms. The error metrics for CVaR are slightly better than those for VaR, indicating more accurate predictions overall.

The poor performance of MLR compared to the other ML models highlights the presence of non-linear relationships between the features and the estimated risk measures. Tree-based methods achieve better prediction accuracy as they are instead able to capture also nonlinearities. We can also observe that RF consistently predicts within the feasible range due to its ensemble structure, which averages outputs inherently bounded by the training data. Gradient boosting methods, on the other hand, may produce predictions outside this range as a result of their sequential error-correcting process, which can lead to overshooting. Despite this, they still exhibit superior overall performance (even before applying any feasible range adjustment), benefiting from their iterative refinement of predictions, in contrast to the parallel nature of RF.

For what concerns computation times, Table 4 reports the seconds needed to train (TrainT) and test (TestT) the different algorithms. As for VaR, we can observe that MLR is the fastest model, requiring only a fraction of a second to train and run the full suite of tests. More CPU time is needed for the training of RF (59s) and LGBM (35s), while the training of XGB is faster (14s). As for the testing, MLR is by far the fastest predictor, XGB requires around 2 seconds, followed by RF (4s) and LGBM

Table 4 CPU times

		MLR	RF	XGB	LGBM
VaR	*TrainT (s)*	0.109	59.370	14.474	34.990
	TestT (s)	0.104	4.090	2.235	5.765
CVaR	*TrainT (s)*	0.108	59.015	21.536	25.815
	TestT (s)	0.112	4.054	4.287	3.828

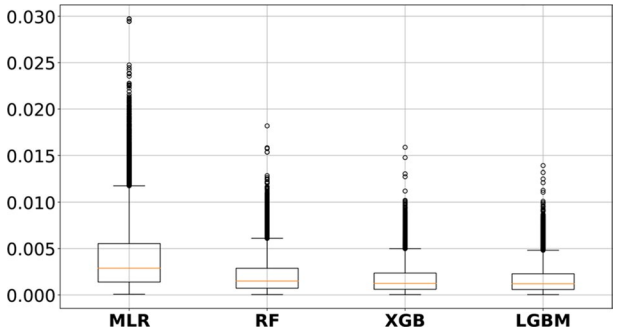

Fig. 9 Width of prediction intervals on test set for VaR estimation

(6s). In absolute terms these differences in test times could be considered relevant; however, since one is often interested in the prediction of a single instance, also the use of the slowest predictor would probably make little difference since the time required for a single prediction in all cases is barely measurable (note that TestT is the overall time needed to predict all test instances, that are more than 64 000 in our study). Concerning CVaR computation times, we observe a similar behavior to VaR, with the exception of LGBM, which, for the CVaR has times comparable to XGB.

Overall, we can observe that XGB and LGBM are comparable in terms of accuracy and computational times, and both represent a very fast and precise (and thus viable) practical alternative to the counting method (Meloni and Pranzo 2023).

5.4 Variability of the prediction intervals

To evaluate the reliability of the predictions, we used a bootstrapping approach (Angelopoulos and Bates 2023; Vovk et al. 2005), which creates multiple models based on the same ML algorithm but trained on different samples obtained through random changes in the training/test composition. This approach enables the estimation of an interval of possible predicted values for each instance, where the original prediction (without bootstrapping) falls with a desired confidence level, which we set to 95%.

Figures 9 and 10 show box plots for each forecasting algorithm (i.e., MLR, RF, XGB, and LGBM), illustrating the variability of the prediction intervals for VaR and CVaR estimation respectively. These box plots display key statistics on the relative width of the prediction intervals across test instances, including minimum value, 25th percentiles, median value, 75th percentile, maximum value, and outliers. The relative width of a prediction interval can be computed as $(ub^p - lb^p)/y^p$, where ub^p and lb^p are the upper and lower limits of the prediction interval obtained with the bootstrap-

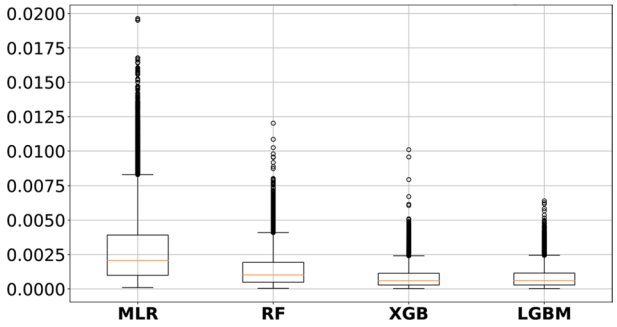

Fig. 10 Length of prediction intervals on test set for CVaR estimation

ping approach at 95% of confidence, and y^p is the predicted value of VaR or CVaR obtained by the ML models. Given a forecasting algorithm, a relative width equal to zero on an instance indicates a highly confident prediction, meaning that all models generated with the bootstrapping sampling provide the same predicted value for that instance.

In Fig. 9, MLR exhibits the widest prediction intervals, indicating the poorest performance, followed by RF. XGB and LGBM show no significant statistical differences in their behaviors and result to be the ones providing the smallest prediction intervals, suggesting greater stability and accuracy. Moreover, the tree-based methods show fewer worst-case outliers compared to MLR, further highlighting their robustness.

Figure 10 for CVaR shows similar trends to VaR, with XGB and LGBM producing the narrowest forecast intervals. Additionally, forecast intervals for CVaR are generally smaller than those for VaR, indicating more accurate forecasts.

6 Conclusions and future research

In this paper, we provided different machine learning models estimating two quantile-based risk measures for the makespan, values-at-risk and conditional-value-at-risk. Our models leverage features that describe activity networks associated with feasible schedules in complex job shops with uncertain durations, where only the range of possible duration values is known in advance. In particular, the input features we used to train the ML models fall into three categories: those describing the deterministic instance, those describing the uncertain instance, and those characterizing the distribution of uncertainty within the uncertain instance. Feature importance analysis revealed that all features, except for the number of nodes, significantly impact the prediction of risk measures in at least one of the tested models. To our knowledge, this is the first study to explore ML models for estimating these specific risk measures, and no prior work has identified valid input features for this context. We validated our approach through extensive computational experiments on a comprehensive set of benchmark instances. Among the tested models, Extreme Gradient Boosting and Light Gradient Boosting Machine emerged as the most effec-

tive, achieving extremely accurate predictions with a mean absolute percentage error smaller than 0.2%. Moreover, a single estimate can be obtained in a few microseconds on average, offering both accuracy and efficiency.

The results obtained show that ML models, and in particular XGB and LGBM, can effectively be used for risk assessment. Thanks to their high accuracy and minimal real-time computational requirements, these models enable the development of risk-aware optimization algorithms, where the risk of the incumbent solution is provided by the forecasting model at each iteration. The more costly exact evaluation, instead, could be used for assessing the quality of the proposed optimal solutions. Another promising direction for future research involves the development of predictive ML models tailored to specific applications, such as project scheduling or train rescheduling. In real-world scenarios, instances often exhibit common characteristics, making them ideal candidates for tailored training that could yield even more precise predictors.

Acknowledgements Carlo Meloni was instrumental for the realization of this paper. He was the one that gathered the authors, gave the initial push, started writing with us the first draft of this work and coordinated our efforts until his passing. We acknowledge his contribution and thank him for all he has given us in these years of friendship.

Funding Open access funding provided by Università degli Studi Roma Tre within the CRUI-CARE Agreement.

Declarations

Conflicts of Interest The authors have no relevant financial or non-financial interests to disclose. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Adams J, Balas E, Zawack D (1988) The shifting bottleneck procedure for jobshop scheduling. Manage Sci 34(3):391–401
2. Aggarwal CC (2015) Data mining: the textbook, 1. Springer
3. Angelopoulos AN, Bates S (2023) Conformal prediction: a gentle introduction. Foundations and Trends® in Machine Learning 16(4):494–591
4. Angrist JD, Pischke JS, (2009) Mostly harmless econometrics: an empiricist's companion. Princeton University Press
5. Applegate D, Cook W (1991) A computational study of job-shop scheduling. ORSA Journal on Computer 3(2):149–156

6. Arlot S, Celisse A (2010) A survey of cross-validation procedures for model selection. Statistics Surveys 4:40–79
7. Atakan S, Bülbül K, Noyan N (2017) Minimizing value-at-risk in single-machine scheduling. Ann Oper Res 248(1–2):25–73
8. Baker K, Trietsch D (2009) Principles of sequencing and scheduling, 2nd edn. John Wiley & Sons, New York
9. Bergstra J, Bengio Y (2012) Random search for hyper-parameter optimization. J Mach Learn Res 13:2
10. Bertsimas D, Lauprete GJ, Samarov A (2004) Shortfall as a risk measure: properties, optimization and applications. Journal of Economic Dynamics & Control 28(7):1353–1382
11. Breiman L (2001) Random forests. Mach Learn 45:5–32
12. Brucker P, Knust S (2013) Complex Scheduling, 2nd edition. Springer
13. Bruni ME, Guerriero F, Pinto E (2009) Evaluating project completion time in project networks with discrete random activity durations. Computers & Operations Research 36(10):2716–2722
14. Catalão JPS, Pousinho HMI, Contreras J (2012) Optimal hydro scheduling and offering strategies considering price uncertainty and risk management. Energy 37:237–244
15. Chen T, Guestrin C (2016) XGBoost: a scalable tree boosting system, proceedings of the 22nd ACM SIGKDD international conference on knowledge Discovery and Data Mining, 785–794
16. Cheng L, Tang Q, Zhang Z, Wu S (2021) Data mining for fast and accurate makespan estimation in machining workshops. J Intell Manuf 32:483–500
17. D'Ariano A, Pacciarelli D, Pranzo M (2007) A branch and bound algorithm for scheduling trains in a railway network. Eur J Oper Res 183(2):643–657
18. Elmaghraby SE (1977) Activity networks: project planning and control by network models. Wiley, New York
19. Elmaghraby SE (2005) On the fallacy of averages in project risk management. Eur J Oper Res 165(2):307–313
20. Fisher H, Thompson G (1963) Probabilistic learning combinations of local job-shop scheduling rules. In: Muth JF, Thompson GL, Industrial scheduling. Prentice-Hall, 225–251
21. Framinan JM, Leisten R, Ruiz García R (2014) Manufacturing scheduling systems. Methods and tools. Springer-Verlag, London, An Integrated View on Models
22. Fransoo JC, de Kok TG, Paulli J (1994) Makespan estimations in flexible manufacturing systems, TU Eindhoven. Fac. TBDK, Vakgroep LBS : working paper series, 9414, Eindhoven University of Technology
23. Friedman JH (2001) Greedy function approximation: a gradient boosting machine. Ann Stat 2001:1189–1232
24. Friedman JH (2002) Stochastic gradient boosting. Computational statistics & data analysis 38(4):367–378
25. García-González J, Parrilla E, Mateo A (2007) Risk-averse profit-based optimal scheduling of a hydro-chain in the day-ahead electricity market. Eur J Oper Res 181:1354–1369
26. Grinsztajn L, Oyallon E, Varoquaux G (2022) Why do tree-based models still outperform deep learning on typical tabular data? Adv Neural Inf Process Syst 35:507–520
27. Hammami NEH, Lardeux BB, Hadj-Alouane A, Jridi M (2024) Design and calibration of a DRL algorithm for solving the job shop scheduling problem under unexpected job arrivals. Flexible Services and Manufacturing Journal, 1–32
28. Hastie T, Tibshirani R, Friedman J (2009) The elements of statistical learning, 2nd edition. Springer
29. James G, Witten D, Hastie T, Tibshirani R (2013) An introduction to statistical learning. Springer, New York
30. Jun S, Lee S, Chun H (2019) Learning dispatching rules using random forest in flexible jobshop scheduling problems. Int J Prod Res 57(10):3290–3310
31. Kalinchenko K, Veremyev A, Boginski V, Jeffcoat DE, Uryasev S (2011) Robust connectivity issues in dynamic sensor networks for area surveillance under uncertainty. Pacific Journal of Optimization 7:235–248
32. Ke G, Meng Q, Finley T, Wang T, Chen W, Ma W, Ye Q, Liu TY (2017) LightGBM: a highly efficient gradient-boosting decision tree. Adv Neural Inf Process Syst 2017:30
33. Larsen R, Pranzo M (2019) A framework for dynamic rescheduling problems. Int J Prod Res 57(1):16–33

34. Lawrence S (1984) Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques. Carnegie-Mellon University, Supplement), Graduate School of Industrial Administration
35. Liao L, Sarin SC, Sherali HD (2012) A scenario generation-based lower bounding approach for stochastic scheduling problems. Journal of the Operational Research Society 63:1410–1420
36. Li Y, Carabelli S, Fadda E, Manerba D, Tadei R, Terzo O (2020) Machine learning and optimization for production rescheduling in industry 4.0, The International Journal of Advanced Manufacturing Technology, 110, 9-10, 2445–2463
37. Li X, Olafsson S (2005) Discovering dispatching rules using data mining. J Sched 8(6):515–527
38. Liu L, Urgo M (2023) A branch-and-bound approach to minimise the value-at-risk of the makespan in a stochastic two-machine flow shop. Int J Prod Res 62(6):2107–2123
39. Lundberg SM, Lee SI (2017) A unified approach to interpreting model predictions. Adv Neural Inf Process Syst 30:4765–4774
40. Meloni C, Pranzo M (2020) Expected shortfall for the makespan in activity networks under imperfect information. Flex Serv Manuf J 32:668–692
41. Meloni C, Pranzo M (2023) Evaluation of the quantiles and superquantiles of the makespan in interval valued activity networks. Computers & Operations Research 151:106098
42. Meloni C, Pranzo M, Samà M (2021) Risk of delay evaluation in real-time train scheduling with uncertain dwell times. Transportation Research Part E Logistics and Transportation Review 152:102366
43. Meloni C, Pranzo M, Samà M (2022) Evaluation of VaR and CVaR for the makespan in interval valued blocking job shops. Int J Prod Econ 247:108455
44. Nakasuka S, Yoshida T (1992) Dynamic scheduling system utilizing machine learning as a knowledge acquisition tool. Int J Prod Res 30(2):411–431
45. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011) Scikit-learn: machine learning in Python. J Mach Learn Res 12:2825–2830
46. Raaymakers W, Weijters A (2003) Makespan estimation in batch process industries: a comparison between regression analysis and neural networks. Eur J Oper Res 145(1):14–30
47. Rockafeller RT, Royset JO (2013) Superquantiles and Their Applications to Risk, Random Variables, and Regression, INFORMS Tutorials in Operations Research, 151–167
48. Romero-Silva R, Hernández-López G (2020) Shop-floor scheduling as a competitive advantage: a study on the relevance of cyber-physical systems in different manufacturing contexts. Int J Prod Econ 224:107555
49. Rossit DA, Tohmé F, Frutos M (2019) Industry 4.0: Smart scheduling. Int J Product Res 57(12):3802–3813
50. Sang P, Begen MA, Cao J (2021) Appointment scheduling with a quantile objective problems. Comput Operat Res 132:105295
51. Sarin SC, Nagarajan B, Liao L (2010) Stochastic scheduling: expectation-variance analysis of a schedule. Cambridge University Press, New York
52. Sarin SC, Sherali HD, Liao L (2014) Minimizing conditional-value-at-risk for stochastic scheduling problems. J Sched 17:5–15
53. Schneckenreither M, Haeussler S, Gerhold C (2020) Order release planning with predictive lead times: a machine learning approach. Int J Prod Res 59(11):3285–3303
54. Shao C, Yu Z, Ding H, Cao G, Ding K, Duan J (2024) A dynamic flexible job shop scheduling method based on collaborative agent reinforcement learning. Flexible Services and Manufacturing Journal, 1-33
55. Sterna M (2021) Late and early work scheduling: A survey. Omega 104:102453
56. Tao L, Wu DD, Liu S, Dolgui A (2018) Optimal due date quoting for a risk-averse decision-maker under CVaR. Int J Prod Res 56(5):1934–1959
57. Taquet V, Blot V, Morzadec T, Lacombe L, Brunel N (2022) MAPIE: an open-source library for distribution-free uncertainty quantification, arXiv preprint, arXiv:2207.12274
58. Tremblet D, Thevenin S, Dolgui A (2023) Makespan estimation in a flexible job-shop scheduling environment using machine learning. Int J Prod Res 62(10):3654–3670
59. Urgo M, Váncza J (2019) A branch-and-bound approach for the single machine maximum lateness stochastic scheduling problem to minimize the value-at-risk. Flex Serv Manuf J 31:472–496
60. Vovk V, Gammerman A, Shafer G (2005) Algorithmic learning in a random world, 29. Springer, New York

61. Wiesemann W (2012) Optimization of temporal networks under uncertainty. Springer, Heidelberg
62. Yamashiro H, Nonaka H (2021) Estimation of processing time using machine learning and real factory data for optimization of parallel machine scheduling problem. Operations Research Perspectives 8:100196
63. Xiong H, Shi S, Ren D, Hu J (2022) A survey of job shop scheduling problem: the types and models. Computers & Operations Research 142:105731
64. Zhang J, Ding G, Zou Y, Qin S, Fu J (2019) Review of job shop scheduling research and its new perspectives under Industry 4.0. J Intell Manufact 30(4):1809–1830
65. Zhang M, Tao F, Nee AYC (2021) Digital twin enhanced dynamic job-shop scheduling. J Manuf Syst 58:146–156
66. Zieliński P (2005) On computing the latest starting times and floats of activities in a network with imprecise durations. Fuzzy Sets Syst 150:53–76

Alice Calamita works at an innovative spin-off of the Department of Computer, Control and Management Engineering (DIAG) at Sapienza University of Rome, where she is responsible for developing analytical solutions to support strategic and operational decision-making. She received a Ph.D. in Operations Research from Sapienza University of Rome in 2024 and an M.Sc. degree in Management Engineering from the same university in 2020. Her research interests included the efficient modeling and solution of emerging combinatorial optimization problems in network design and the integration of machine learning techniques into optimization-based decision support systems.

Carlo Meloni is Associate Professor in Operations Research at Sapienza University of Rome. He is a founding member of the EURO Working Group on Stochastic Optimization (EWGSO). His main research interest referr to theory and practice of Operations Research and the application of optimization, simulation and other OR/MS methodologies both to theoretical and real-world problems. He is the author of more than 70 scientific papers.

Marco Pranzo is Associate Professor in Operations Research at the University of Siena. His research interest focus on the design and development of heuristics and exact algorithms for scheduling and combinatorial optimization problems arising from complex real-world applications in manufacturing, transportation, healthcare and so on. He also investigates the influence of uncertainty and the associated risk on deterministic solution frameworks. He is the author of more than 60 papers on high quality scientific journals.

Marcella Samà is Associate Professor in Operations Research at Roma Tre University. Her research focuses mainly on designing and developing models and algorithms aimed at solving scheduling and combinatorial problems for real-world application in public services problems, with a specific regard to transit and logistics. She works on decision support systems and solver which consider advanced OR/MS methodologies for real-time management problems, e.g., in the railway and air traffic flow sectors. She has authored more than 40 papers on international scientific journals.

Authors and Affiliations

Alice Calamita[1] · Carlo Meloni[1] · Marco Pranzo[2] · Marcella Samà[3]

✉ Marcella Samà
marcella.sama@uniroma3.it

Alice Calamita
alice.calamita@uniroma1.it; alice.calamita.ac@gmail.com

Marco Pranzo
marco.pranzo@unisi.it

[1] Dipartimento di Ingegneria Informatica Automatica e Gestionale, Sapienza University of Rome, Via Ariosto 25, 00185 Roma, Italy

[2] Dipartimento di Ingegneria dell'Informazione e Scienze Matematiche, Università di Siena, Via Roma 56, 10587 Siena, Italy

[3] Dipartimento di Ingegneria Civile, Informatica e delle Tecnologie Aeronautiche, Università degli Studi Roma Tre, Via Vito Volterra, 62, 00146 Roma, Italy