IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received January 20, 2022, accepted February 10, 2022, date of publication February 24, 2022, date of current version March 25, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3154445

Analyzing and Predicting Verification of Data-Aware Process Models–A Case Study With Spectrum Auctions

ELAHEH ORDONI[1], JAKOB BACH[1], AND ANN-KATRIN FLECK[2]
[1]Institute for Program Structures and Data Organization, Karlsruhe Institute of Technology (KIT), 76131 Karlsruhe, Germany
[2]Institute for Economics, Karlsruhe Institute of Technology (KIT), 76131 Karlsruhe, Germany

Corresponding author: Elaheh Ordoni (elaheh.ordoni@kit.edu)

This work was supported by the KIT-Publication Fund of the Karlsruhe Institute of Technology.

ABSTRACT Verification techniques play an essential role in detecting undesirable behaviors in many applications like spectrum auctions. By verifying an auction design, one can detect the least favorable outcomes, e.g., the lowest revenue of an auctioneer. However, verification may be infeasible in practice, given the vast size of the state space on the one hand and the large number of properties to be verified on the other hand. To overcome this challenge, we leverage machine-learning techniques. In particular, we create a dataset by verifying properties of a spectrum auction first. Second, we use this dataset to analyze and predict outcomes of the auction and characteristics of the verification procedure. To evaluate the usefulness of machine learning in the given scenario, we consider prediction quality and feature importance. In our experiments, we observe that prediction models can capture relationships in our dataset well, though one needs to be careful to obtain a representative and sufficiently large training dataset. While the focus of this article is on a specific verification scenario, our analysis approach is general and can be adapted to other domains.

INDEX TERMS Formal verification, machine learning, model checking, spectrum auctions.

I. INTRODUCTION

A. MOTIVATION

Industry takes great interest in verification techniques to detect undesirable behavior of processes before their execution. A very relevant domain for verification is auction design. This current article focuses on spectrum auctions. Despite their importance, embarrassing outputs have happened in the past. A wrong policy to increase bid prices in the US mobile market resulted in a loss of 70 billion dollars [1]. In another case [2], about fifty percent of the products remained unsold at the end of the auction. To alleviate such results, one can apply methods from the fields of auction theory, verification, and machine learning. We will discuss each of these points in the following. Our article is located at the intersection of all three fields.

The associate editor coordinating the review of this manuscript and approving it for publication was Ehab Elsayed Elattar.

1) AUCTION THEORY

Literature features two alternatives to study auctions, namely (a) experimental analyses with human subjects performed in laboratories [3], or (b) theoretical analyses. However, finding undesirable outcomes continues to be a challenge with both categories: Regarding (a), laboratories tend to perform only relatively few experiments to arrive at possible outcomes. To check all possibilities of an experimental design in [4] would require more than 13 million experiments, to give an example. Such a setup is beyond the capacity of any laboratory. Regarding (b), researchers use auction theory to predict equilibrium results, based on assumptions on bidding behavior [5]. A standard assumption is rationality of bidders [6]. However, this assumption does not always hold [7], [8]. Even the development of frameworks for truthful bidding under interference constraints [9] cannot preclude irrational bidders. Overall, catastrophic outcomes of auctions can result from design errors going unnoticed.

2) VERIFICATION

Verification methods have been developed to find undesirable behavior in process models that work with data, so-called *data-aware process models*. With such techniques applied to auctions, one can detect the lowest possible revenue of an auctioneer or the lowest efficiency of an auction design [10], to give examples. However, respective techniques tend to be computationally expensive: First, the state space grows exponentially with the number of data objects. Exploring the state space to verify properties is slow or even infeasible. Second, the number of properties to be verified is vast. To illustrate, think of the 4G German spectrum auction with six products, each one associated with bids ranging from $\approx 10^4$ to $\approx 10^9$ euros [11]. The number of properties to detect the lowest revenue of this auction grows exponentially with the number of products. This amounts to the verification of millions of properties to detect the lowest revenue of the 4G German spectrum auction. As a result, verification techniques also have problems detecting unexpected outcomes of spectrum auctions.

3) MACHINE LEARNING

In the face of the previously described efficiency issues of auction-theoretic approaches and verification methods, machine learning might offer a way out. Predicting the outcomes of auctions or verification results can be significantly faster than running the underlying methods themselves. Machine learning on auction data has focused on a variety of auction designs [12]–[15], which are different from the one used in this article, however. Literature on machine learning in the field of verification is broad as well [16]. However, related work [17]–[19] focuses on other use cases than our study. Thus, differences in the domain, formal languages, and verified properties limit comparability to our work.

B. CONTRIBUTIONS

We leverage machine-learning techniques to predict verification results for a specific data-aware process model, i.e., to detect undesirable outcomes of an auction. To this end, we work with a dataset consisting of roughly 130,000 verification runs on a process model of the German 4G spectrum auction to sell one of the most valuable bandwidths, the 800 MHz band.

Our study consists of two parts. First, to explore the dataset, we analyze the distribution of auction outcomes and characteristics of the verification procedure, e.g., verification time. Second, we make predictions for three prediction targets, i.e., revenue, verification result, and verification time. Revenue is essential for the auctioneer, the verification results help to determine revenue, and analyzing verification time might help to improve the verification procedure. We study whether one can predict these quantities only with features describing the auction design and the properties to be verified, i.e., without knowing the course of actual verification runs themselves. As we still had to run verification when creating the dataset, we can evaluate the performance of predictions against this ground truth. We use random forests as prediction models, which have high predictive power due to ensembling and can capture non-linear dependencies in data. We also analyze the importance of features for the prediction models.

Such predictions would save costly verification runs of a vast number of properties and can give the auctioneer a rough estimate of possible outcomes. In particular, verification yields exactly which outcomes of auctions are possible, but it is slow. In contrast, machine learning makes estimates fast, but predictions might not be perfectly accurate. We also analyze if one can train models with only a subset of products and predict verification results for the other products. Such an approach would allow estimating the output of auctions with more products, even though verification of such auctions is not feasible due to the state-space explosion. Efficiently making such predictions is not possible for arbitrary process models and properties due to the theoretical hardness of model checking. However, it might still be possible in use cases with a limited scope, as our auction scenario. We analyze how well prediction models generalize for different ways to split the data into training and test set.

C. RESULTS

Even for one auction design, the outcomes of the analyzed auction, i.e., winners and prices of products, vary, and the dataset reflects this. In particular, revenue is suboptimal in most cases. Prediction models successfully capture these variations in outcomes. They show close-to-optimal prediction quality if evaluated with standard cross-validation, i.e., randomized splits. This observation applies to all prediction targets, i.e., verification result, verification time, and revenue. Additionally, training and making predictions is fast, taking under a second for our dataset, which results from hours of running verification.

However, prediction models might have problems to generalize to other auction settings, even if they reproduce the given scenario used for training. For example, prediction quality drops if we train models only with some auction products and make predictions for other products. In the worst case, we observe predictions on the unseen test data that are not better than random guessing. Consequently, we see the collection of a training dataset that is representative regarding process models and properties as a crucial factor for any attempt to use machine learning in the context of verification.

We observed a decrease in prediction performance with reduced training set size for randomly sampling verification runs into the training set. However, the decrease in prediction performance was not proportional to dataset size, but lower, which we find encouraging. Thus, one needs to find an acceptable trade-off between cost in dataset collection and prediction performance. This trade-off decision depends on the specific scenario and user, e.g., auctioneer.

Being aware of these caveats, we see a combination of machine learning and verification promising to fine-tune auction designs. Such an approach might allow detecting

and subsequently excluding designs with the least favorable outcomes for the auctioneer.

We publish the dataset, experimental results, and the code for all our analyses, c.f. Section V-D.

D. OUTLINE

Section II reviews related work. Section III describes simultaneous multi-round (SMR) auctions. Section IV explains their verification, in general as well as specifically to create our dataset. Section V describes our dataset and experimental design. Section VI evaluates the experimental results in detail. Section VII summarizes and discusses the experimental results. Section VIII concludes.

II. RELATED WORK

In this section, we review related work from different areas that are relevant for this article: spectrum auctions, verification of process models, and machine learning.

A. SPECTRUM AUCTIONS

Spectrum auctions have been studied for a long time. [20] has a look at the simultaneous ascending auction from an auction-theoretic point of view. [21] shows the limitations of theoretical findings on Simultaneous Multi-Round (SMR) auctions. [22] looks at the 3G auction design in the UK and Germany and compares them experimentally. [23] conducts an auction-theoretic analysis of the 3G auction. [11] analyzes the 4G auction auction-theoretically. They conclude that although the auction ended efficiently, implicit collusion to achieve low prices would have been possible. [24] provides a secure solution for truthful spectrum auctions. [25] designs a spectrum-auction mechanism that creates an efficient outcome in polynomial time. Still, an underlying assumption of the proposed mechanisms and frameworks is rationality.

B. VERIFICATION OF PROCESS MODELS

Many approaches exist for the verification of process models. In the following, we only mention works that are relevant for this article.

Authors in [10] studied the verification of data-aware process models that support modification of data values. However, their approach only allows verifying spectrum auctions with at most three bidders and products. In another recent approach, authors verify data-aware process models using colored Petri Nets [26]. They model the complete domain of data objects: To represent a data object with n distinct values, i.e., colors, they generate n new states. Consequently, the number of states becomes too large, and the verification procedure is computationally expensive.

A range of abstraction techniques has been developed, aiming to reduce the size of the process models and, thus, the state space [27], [28]. Here, the idea is to determine the values of data objects necessary for verification and combine all the unnecessary values into an abstracted one. The problem is that such techniques might yield an incorrect result [10]. This outcome may happen when process elements such as activities modify the value of data objects, e.g., an activity that increases the price of a product. In another approach [29], authors first abstract the process model and then evaluate all data objects of each abstracted process fragment for three sets of rules. Each rule keeps or deletes a data object in a process model. As a downside, the rules in [29] do not preserve the verification result when data values are modified. The approach in [30] uses symbolic abstraction and supports data modifications based on decision tables. A decision table comprises a set of rules that consist of conditions for the inputs and expressions. The abstraction technique proposed in [30] is not effective when an activity modifies the value of a data object with a large domain. Besides this, when they detect an undesirable outcome, they cannot provide a counterexample.

As another approach, reduction techniques based on relevance are proposed [31], [32]. The idea here is to detect the process elements that are necessary to verify a given property, so-called *relevant elements*. The irrelevant elements are candidates for reduction. This technique works quite well, but the number of properties to be verified stays the same or even becomes larger. [33] also reduces data objects and attributes used in a process model by abstraction. Their reduction aims to simplify the visualization of process models, and there is no guarantee that the abstracted data elements preserve a verification result.

C. MACHINE LEARNING WITH VERIFICATION DATA

[16] provides a survey of how machine learning helps in various verification approaches: SAT/SMT solving, theorem proving, model checking, and static analysis. The authors state that the choice of features is critical for such scenarios. In addition, they note that interpretability might become problematic for powerful machine-learning models. Regarding model checking, they review approaches for finding counterexamples and for finding frequent paths, which are different from the aspects of model checking we analyze here.

In the following, we present several studies that use machine learning to predict the outcome of verification or to predict characteristics of the verification process, like verification time. Our study focuses on one specific use case. Thus, predictions performances from other domains, model types, and properties might not be comparable, so we refrain from reporting concrete results here. [17] uses regression models to predict hardware-verification time with a dataset of 10,000 hardware designs. [34] conducts a case study to predict two verification metrics in hardware design. [35] applies classification models to recommend model checkers for hardware-verification instances, using a dataset of 16,500 hardware designs. [36] uses classifiers to recommend model checkers based on characteristics of the models. They study the Model Checking Contest 2017, which featured 10 model checkers and 77 models from different categories. [18] addresses the problem of state-space explosion in verification. They train classifiers to detect

deadlocks on small models and then apply them to large models, using three different scenarios for evaluation. [19] predicts whether Kripke structures satisfy LTL formulas, randomly generating 50 Kripke structures and 200 LTL formulas. [37] predicts whether workflow nets are sound or not, generating 2209 workflows nets with different characteristics.

Finally, while we apply machine learning to a verification scenario, there also is work in the other direction, i.e., verifying machine learning-approaches [38], [39].

D. MACHINE LEARNING WITH AUCTION DATA

Machine learning has been applied to various types of auctions, for which we can only give some examples. Note that the auction design is different from ours in all these cases. [12] uses machine learning to efficiently determine allocations for combinatorial auctions, where bidders can place bids on bundles of products. They evaluate their approach with three different models of spectrum auctions from the *Spectrum Auction Test Suite* [40]. [15] uses regressions models to find a schedule of products in sequential auctions by predicting the revenue. [13] trains a recommender to find suitable bidders in procurement auctions. [14] uses models to determine a revenue-optimal reserve price for second-price auctions.

III. SIMULTANEOUS MULTI-ROUND (SMR) AUCTIONS

Simultaneous Multi-Round (SMR) auctions have been the standard format to allocate spectrum licenses to bidders for more than two decades [41]. This auction type allows selling several products, e.g., spectrum licenses, after several bidding rounds.

At the beginning of an auction, the auctioneer specifies a *reserve price* for each product, i.e., its lowest acceptable price. Each bidder may choose to bid on zero, one, or multiple products simultaneously in each round. In the type of auction we analyze, each bidder has an individual budget for each product. In this case, the budget closely reflects a bidder's valuation for this individual product. Thus, bidders cannot use the leftover budget from one product to acquire a different product. In addition, bidders make separate bids for each product, i.e., they cannot bid on bundles of products, different from combinatorial auctions. Additionally, there is a so-called *capacity rule* [42]: Each bidder has a *capacity*, the maximum number of products they may win. This rule prevents bidders from winning too many items. In spectrum auctions, this guarantees a certain number of awarded bidders and prevents bidders from forming a monopoly. After bidding finishes in a round, the highest bid for each product will be its reserve price in the following round. This bid is announced to all bidders, while other bids are not disclosed. In addition, bidders do not know their competitors' bids from the currently ongoing bidding round. The auction ends when there is no new bid for any product in a bidding round. The bidder with the highest standing bid for each product is its winner.

IV. VERIFICATION OF SMR AUCTIONS –DATASET CREATION

This paper presents an approach to analyze and predict verification results in a data-driven manner. To this end, we create a dataset by verifying properties of an SMR auction model. Fig. 1 provides an overview of our verification approach. From a high-level perspective, the approach is generalizable and can also accommodate other scenarios than SMR auctions. In this section, we will describe the general steps and their customization to our specific dataset.

First, we model the SMR auction in BPMN (Activity *model in BPMN*, Section IV-A). However, verifying the resulting BPMN model is not feasible due to state-space explosion. Thus, we use approaches from related work to reduce the state space on the level of process models (Activity *reduce process*, Section IV-B). Then, we map the reduced BPMN model to Petri Nets [43] (Activity *transform to Petri Nets*, Section IV-C). Given the final Petri Nets and properties in form of CTL formulas [44], we employ an off-the-shelf model checker to verify properties (Activity *model checking*, Section IV-D).

A. PROCESS MODEL IN BPMN

We use BPMN to model the design of the German 4G spectrum auction to sell 800 MHz band [11]. The auction has four bidders and six products. We assign a random budget from the range [1, 100] to each bidder for each product, similarly to the auction experiments in [4]. We also define a reserve price of 3 for all products. Further, each bidder has an individual capacity. Table 1 shows the budgets and capacities of the bidders.

TABLE 1. Design of the SMR auction.

	Bidder 1	Bidder 2	Bidder 3	Bidder 4
Budget for Product 1	90	90	70	60
Budget for Product 2	90	80	80	60
Budget for Product 3	90	80	70	70
Budget for Product 4	60	80	70	60
Budget for Product 5	90	80	70	90
Budget for Product 6	90	80	60	60
Capacity	2	3	2	1

Fig. 2 shows a simplified version of this auction in BPMN notation. This model consists of three subprocesses: *availability of bidders*, *bidding of each bidder*, and *winner determination*. The first subprocess checks whether bidders can afford additional products that they have not won yet (*availability of bidders*). The auction continues if there is at least one qualified bidder who can place a bid in Subprocess *bidding of each bidder*. Activity *place bid* issues a random bid between the current price of the product and the budget of the qualified bidder. In our process model, bidders will always bid if they have both budget and capacity left to acquire a product. Activity *decrease capacity* decreases the capacity of the bidder who just won a product. If a bidder

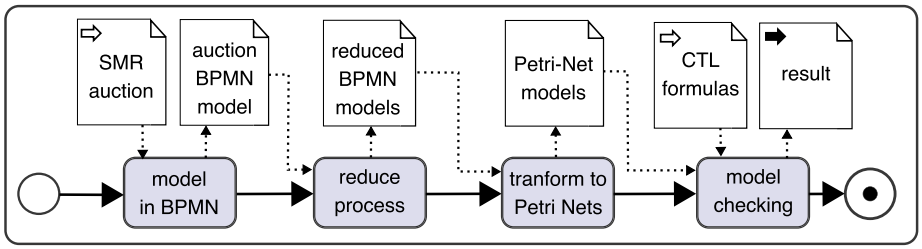


FIGURE 1. Verification overview.

has no capacity left, Activity *remove bid* removes their bids. Subprocess *winner determination* outputs the new reserve prices and the winners based on the existing bids. These three subprocesses are repeated until no more bids are placed. The resulting BPMN model to represent the German 4G spectrum auction consists of 423 control-flow elements and 150 data objects.

B. REDUCTION OF PROCESS MODEL BASED ON PROPERTIES

To verify a given SMR auction, one could map the process model directly to a Petri Net. Then, by employing an off-the-shelf model checker, one could verify properties in the Petri Net. However, to verify our SMR auction design, directly using existing verification techniques is infeasible. This infeasibility results from state-space explosion [45], i.e., the state space increases exponentially with the number of data objects and values. To overcome this problem, we use a reduction algorithm analogous to [32] to reduce the size of the process model. This reduction algorithm makes use of the properties to be verified. In particular, the algorithm detects the elements of the process model that are necessary to verify the property, so-called *relevant elements*. All the irrelevant elements are pruned, resulting in a much smaller process model and thus a smaller state space. To illustrate, the following property checks whether Product 1 can be sold for the price of 2 at the end of the process:

$$\text{Property (I)}: \; EF(product.1.price.2 \wedge e.end).$$

We detect the relevant process elements and reduce the process model accordingly to verify this property. However, a property to verify a price of Product 1 results in a set of relevant elements different from that of Product 2. With n products, this calls for n different reductions of the process model. In our study, we reduce our process model six times, one time for each product.

C. TRANSFORMATION OF PROCESS MODELS TO PETRI NETS

Given the reduced process models, we transform them to Petri Nets with the rules from [10]. We use plain Petri Nets as a target for the mapping because of the availability of efficient analysis techniques [46]. Note that each reduced process model results in a different Petri Net. Thus, we get six Petri Nets, each to verify a specific product. Each Petri Net consists of 562 places, 568 transitions, and more than three million states on average.

D. VERIFICATION OF PROPERTIES

Our overall goal for verification is to detect the auctioneer's revenue, which is the sum of the final prices. In particular, as differences in bidding behavior may result in different outcomes, we are interested in the minimum possible revenue, which the auctioneer wants to maximize. This maximin-principle [47] is used in many applications of auction and decision theory [48]–[50], though calculating solutions is time- and resource-consuming. We do not assume specific bidding behavior, but test all possible behaviors to derive auction results with the lowest possible prices. Therefore, our results are more robust in a real-world application than if we assumed a particular bidding behavior, including rationality of bidders.

Given the reduced state space described before, we verify the lowest final price of each product individually, analogous to Property (I). However, due to the capacity rule, the order of products for verifying the final prices matters. For example, if the price of a product is verified after all products, it might be lower than usual. A cause for this is that the bidders with high budgets for that product might have already won other products and thus have no capacity left, so a bidder with a lower budget wins this product. Thus, we have to check all the possible permutations of products. Note that combining results from these permutations allows simulating the simultaneous bidding behavior from real-world auctions.

Because of the previously discussed points, our verification procedure is iterative. As we have six products, there are $6! = 720$ permutations. For each permutation, we run verification for the products in the permutation order, i.e., we simulate auctioning products off sequentially. We verify the final price first and potential winners second for each product. For the final price, we start with a price one unit lower than the lowest budget for that product and increase the price until verification turns out true. To determine potential winners, we check for all bidders with remaining capacity if they can afford the product to that price. Having found a winner, we decrease their capacity and verify the following product in the permutation. If multiple bidders might win the product, we need to consider all these cases for the subsequent product

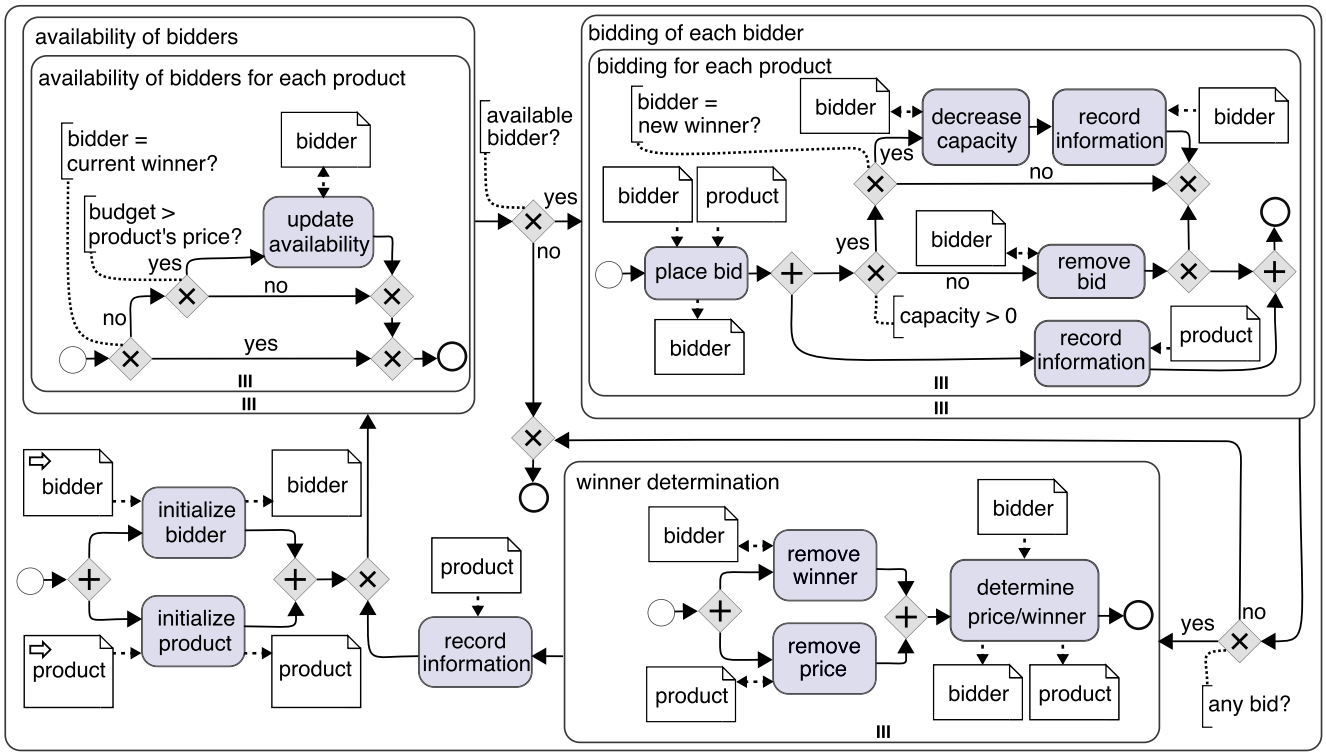


FIGURE 2. Simplified process model of an SMR auction in BPMN notation.

because these cases differ in the remaining capacities of the bidders and might yield different outcomes.

V. EXPERIMENTAL DESIGN

In this section, we start by introducing the dataset in Section V-A and the goals of the experiments in Section V-B. Next, we present the experimental approach in Section V-C. We finish with a few hints on the implementation in Section V-D.

A. DATASET

From the results of the iterative verification procedure, we create a dataset for further analyses. The dataset consists of 130,292 rows and 30 columns (features). Each row corresponds to verifying one property against the underlying Petri Net. The features fall into five different categories:

- *id*: Identifiers for rows, related to the iterative verification procedure.
- *process*: These features are data values in the underlying process model. In our case, budgets and initial capacities of bidders are fixed, as displayed in Table 1. However, due to the iterative verification procedure, capacities decrease once a product has a price and a winner. Thus, we include the bidders' current capacities in the dataset.
- *property*: These features represent the property to be verified, i.e., the CTL formula. In particular, we have properties to verify the price of a product and its winner.
- *verification*: These features characterize the verification procedure, e.g., the verification time and the binary verification result.
- *allocation*: The final allocation consists of the final lowest prices and winners of each product, plus the resulting revenue.

Appendix VIII-D provides a detailed list of all features. Note that we will use subsets of this *full dataset* for predictions, as explained in Section V-C.

B. GOALS

The overall goal of this article is to analyze verification of data-aware process models that represent a simultaneous multi-round auction. In particular, we consider the following research questions to guide our analysis of the dataset described before:

(Q1) Which final allocations, i.e., prices and assignments of products to bidders, are possible?
(Q2) How are revenue, verification result, and verification time related to each other and other features?
(Q3) How well can we predict revenue, verification result, and verification time from other features?
(Q4) Which features of the auction and the verification procedure are most important in the predictions?
(Q5) How well do the prediction models generalize, i.e., how much does their performance depend on the data split?
(Q6) How well do prediction models of different complexity perform?

Our analyses focus on the revenue, verification result, and verification time. If we use the term *revenue* with regard to our experiments, we refer to the lowest possible revenue for a given auction design and verification order of products. Finding the lowest possible revenue is the goal from the domain perspective. The verification result tells us which prices and winners of products are possible, thereby determining the revenue. By predicting the verification result, we want to avoid running the model checker many times to verify all the properties. Verification time is interesting from the perspective of both process modeling and verification. Understanding which factors determine verification time might help to make modeling and verification more efficient.

One can answer (Q1) and (Q2) by exploring the dataset, while the remaining research questions call for the training of prediction models. While (Q3) purely looks at prediction quality, the questions after that target at a better understanding of the factors that influence prediction quality.

C. APPROACH

1) EXPLORATION

For exploratory analysis regarding (Q1) and (Q2), we create plots and compute statistics, focusing on the full dataset with all 130,292 rows. In particular, we analyze the distributions of individual features as well as dependencies between features.

2) PREDICTION

To answer research questions (Q3) to (Q6), we build a prediction pipeline. We define three different prediction scenarios on the dataset (Section V-C2.a). For each of them, we train multiple prediction models (Section V-C2.b). To evaluate prediction quality, we choose an appropriate evaluation metric, depending on the prediction scenario (Section V-C2.c). To analyze generalization performance, we combine this evaluation with different methods to split the data (Section V-C2.d).

a: PREDICTION SCENARIOS

To answer (Q3), we analyze three prediction scenarios, as listed in Table 2. The verification result is a binary variable and, therefore, suitable for classification. Verification time and revenue are continuous variables and, therefore, suitable for regression. As we created the dataset from actual verification runs, we know the ground-truth values of all target variables, so we can use them to evaluate predictions. Note that we only use subsets of the data for predictions, i.e., we create separate *prediction datasets*, as described in the following. We only use those features whose values are known before running verification. Otherwise, the prediction models would be descriptive at best but could not forecast the verification procedure.

For verification result and verification time as targets, predictive features are the capacities of the bidders and the property that is currently verified. Further, we reduce the number of rows in the dataset since the iterative verification procedure described in Section IV-D contains redundancies. In particular, it might repeatedly verify the same property for the same process model. In fact, the full dataset only contains 2043 unique combinations of the feature values we use to predict verification result and verification time. If using the full dataset for predictions, one might overestimate prediction performance due to these duplicate rows, as we found out in preliminary experiments. Thus, we reduce the dataset to these 2043 feature-value combinations for predictions. Each combination is associated with just one verification result in any case, but verification time might vary. Thus, for verification time, we take the mean per feature-value combination as prediction target.

For revenue as prediction target, we need to re-shape the full dataset differently. In particular, we cannot use the same features as we use for predicting verification result and verification time. This is because values for revenue do not result from verifying a single property, but they are a consequence of all verification runs for a product permutation. In addition, verification for each permutation starts with the same capacities, so capacities are not meaningful as features here. Thus, the only features we use to predict revenue are the positions of the six products in the permutation. We get a dataset with one row per product permutation, i.e., 720 rows overall. As multiple revenues might be associated with each permutation, we take the minimum revenue per permutation as the target, which is most interesting for the auctioneer.

b: PREDICTION MODELS

As prediction models for all prediction scenarios, we use random forests [51], which performed well in preliminary experiments. They can solve classification as well as regression problems. In addition, they allow modeling non-linear dependencies between features and target variable, which we expect to exist in our dataset. To vary the model complexity for (Q6), we train models with 1, 10, and 100 trees. To answer (Q4) regarding feature importance, we use a built-in importance measure, which describes how much each feature improved the objective value when training the random forest.

c: EVALUATION METRICS

To evaluate prediction quality, we use Matthews Correlation Coefficient (MCC) [52] for classification and R^2 [53] for regression. Both metrics are normalized such that a perfect prediction has a score of one and naive baselines have a score of zero. For classification, such baselines with a score of zero are randomly guessing class labels and constantly guessing the most common class. For regression, one gets a score of zero by always predicting the mean of the target variable. We focus on test-set performance for both metrics in our evaluation, as random forests can easily overfit the training set, reaching a perfect or close-to-perfect prediction performance there.

TABLE 2. Overview of prediction scenarios.

Target	Task type	Metric	Features	Rows
allocation.revenue	Regression	R^2	*order.p[1-6].pos*	720
verification.result	Classification	MCC	*process.b[1-4].capacity*, *property.product*, *property.price*, *property.winner*	2043
verification.time	Regression	R^2	*process.b[1-4].capacity*, *property.product*, *property.price*, *property.winner*	2043

TABLE 3. Overview of splitting methods.

		Target		
Splitting method	Type	Time	Result	Revenue
10-fold	Random	✓	✓	✓
Reverse 10-fold	Random	✓	✓	✓
Capacity	Feature (*process.p[1-4].capacity*)	✓	✓	
Product	Feature (*property.product*)	✓	✓	
Position	Feature (*order.p1.pos*)			✓

d: EVALUATION SPLITS

To analyze generalization performance for (Q5), we use five different *splitting methods* to evaluate prediction quality. Some methods split the dataset randomly, other methods split along feature values. As the features differ between the prediction scenarios, some splitting methods only apply to some prediction targets. Table 3 provides an overview.

10-fold cross-validation is a standard method in machine learning. It splits the dataset randomly into ten equally sized parts. Models train on nine folds, while the remaining test fold allows assessing generalization performance. This procedure is repeated ten times. In our case study, training on 90% of the data means we could only save 10% of the verification runs. Thus, we also try to train models on 10% of the data and evaluate the models with the remaining 90%. We call this *reverse 10-fold* cross-validation.

The remaining three splitting methods observe feature values. For example, in the *capacity* split, all rows belonging to the same combination of bidders' capacities go either into training or testing. This allows analyzing how well predictions generalize over capacity settings, i.e., different auction designs. The same goes for the other two feature-based splits.

D. IMPLEMENTATION

We implement our analyses in Python and make the code available online.[1] For predictions, we use the package *scikit-learn* [54]. We publish all experimental data, including the full dataset and results.[2] Additionally, we contribute a subset of the data, equal to the prediction scenarios for verification result and verification time, under the name 'Auction Verification' to the UCI Machine Learning Repository.

VI. EVALUATION

First, we address (Q1) and (Q2) by exploring the dataset in Section VI-A. Second, we address (Q3) to (Q6) by training and evaluating prediction models in Section VI-B.

[1] https://github.com/Jakob-Bach/Analyzing-Auction-Verification
[2] https://doi.org/10.5445/IR/1000142949

A. EXPLORING THE DATA

1) ALLOCATIONS

Overall, the full dataset contains 2404 outcomes of the auction. These outcomes include duplicates, i.e., the same final prices and winner assignments can occur in multiple outcomes. In particular, only 36 different combinations of final prices and 31 different combinations of winners occur in the full dataset.

As Table 4a shows, two or three different final prices occur for each product. The distribution of final prices varies strongly between products. E.g., Product 3 and Product 6 both have two possible final prices with similar frequency, while Product 2 and Product 4 show slight variation in their final price. For all products, the final price can be as low as the minimum budget for that product, c.f. Table 1. In contrast, without a capacity rule, one would expect the final price of each product to rise to the second-highest budget for that particular product, as marked in bold in Table 4a. Since this is not always the case in our dataset, we conclude that suboptimal outcomes occur, at least compared to a scenario without a capacity rule. Even considering the capacity rule, most outcomes are not optimal for the auctioneer, as we will see later when analyzing the revenue.

As Table 4b shows, two to four different bidders can occur as the winner for each product. Similar to the prices, the distribution of winning bidders per product varies, e.g., it can be relatively balanced or somewhat skewed. In addition, the bidders with the highest budget for a product do not necessarily win that product due to capacity restrictions. The number of wins per bidder varies between allocations as well, as Table 4c shows: While Bidder 1 always acquires two products, maxing out their capacity, all the other bidders have two different possible numbers of products they acquire. In particular, Bidder 3 never reaches their maximal capacity of two, having comparatively low budgets.

Note that all previously described variations in outcomes base upon the same starting capacities and budgets. However, the order in which the products are assigned to bidders varies. Thus, we conclude that the course of events during the auction significantly impacts the outcome, especially if bidders can run out of capacity.

2) REVENUE

Due to the variation of final prices, different revenues are possible, as Table 4d shows. In the full dataset, revenue ranges from 430 to 490, with a median of 460. Note that the optimal

TABLE 4. Frequency of outcomes. Optimal outcomes on a per-products basis, i.e., disregarding the capacity rule, marked in bold.

(a) Frequency of final prices.

	Price			
Product	60	70	80	90
1	60	820	0	**1524**
2	100	0	**2304**	0
3	0	1204	**1200**	0
4	72	**2332**	0	0
5	0	180	660	**1564**
6	1204	0	**1200**	0

(b) Frequency of wins.

	Bidder			
Product	1	2	3	4
1	**764**	**1580**	60	0
2	**800**	812	792	0
3	**1200**	1004	100	100
4	0	**2332**	72	0
5	**844**	0	0	**1560**
6	**1200**	1004	100	100

(c) Frequency of number of products won.

	Bidder			
Products won	1	2	3	4
0	0	0	1280	644
1	0	0	1124	1760
2	2404	480	0	0
3	0	1924	0	0

(d) Frequency of revenues.

	Dataset	
Revenue	Full	Prediction
430	72	72
440	244	204
450	376	218
460	744	184
470	512	42
480	312	0
490	**144**	**0**

revenue in a setting without capacity rule is 490, i.e., it can even be reached in our capacity-restricted setting. However, it occurs in just 144 out of 2404 outcomes of the full dataset. This observation again highlights that suboptimal results can occur. In the revenue-prediction dataset, taking the minimum revenue for each permutation of products in verification, revenues vary as well, as Table 4d shows.

Revenue can vary in several ways, fostered by the capacity rule. First, assume a fixed order in which products are assigned. Multiple revenues can still occur if there are multiple potential winners for a product, as winner assignment for one product influences the assignment of the following products. Second, assume a fixed assignment of products to bidders. Multiple revenues can still occur because the prices these bidders pay depend on the order of winner assignment. Both these cases occur in our dataset.

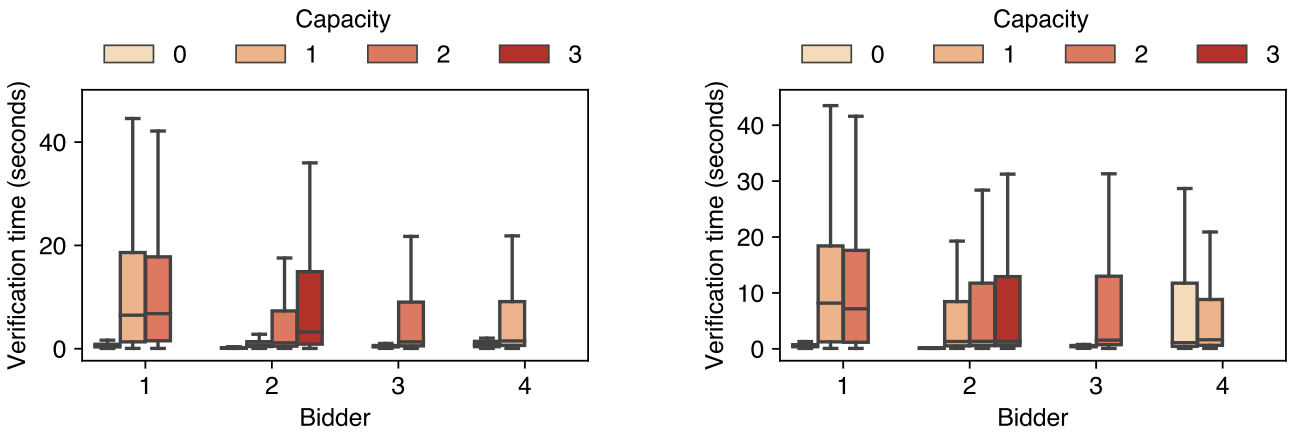
A look at the correlation of final prices to revenue also shows interactions between products: As revenue is the sum of all final prices, increasing a price should increase revenue linearly, at least without capacity considerations. However, Pearson correlations of individual final prices to revenue are only low to moderate in our dataset, having the range [0.11, 0.58]. For comparison, a perfect positive linear dependency would result in a Pearson correlation of 1, while a non-existing linear dependency would result in a Pearson correlation of 0. The highest correlations are for Product 1 and Product 6, which both often have final prices that are clearly below the maximum, c.f. Table 4a, and thus can impact revenue stronger than other products.

3) VERIFICATION RESULT

Verification results are imbalanced in the dataset: Only 14% of verification runs turn out true in the full dataset and 13% in the prediction dataset. This observation makes sense, considering that the iterative verification procedure checks increasing prices until one price is satisfiable, obtaining negative verification results for all lower prices. The fraction of positive verification runs varies between products, bidders, and prices. For the products, different budgets cause different distributions of possible prices and the number of potential winners. Also, bidders with higher budgets can be positively verified as winners more often. For example, Bidder 1, who has the highest average budget, is a potential winner of a product in 86% of the cases in the full dataset, while Bidder 3 only is a potential winner in 14% of the cases. Finally, regarding the prices, higher prices have a higher probability of being verified positively. This finding is a direct consequence of increasing prices until they are satisfiable in the verification procedure – eventually, some valid price must be reached, and then verification of the price for that product stops.

4) VERIFICATION TIME

Verification time shows an asymmetric distribution in the dataset. In the full dataset, it falls in the interval [0.06 s, 47.75 s], with a mean of 6.31 s and a median of 1.22 s. Values are similar in the prediction dataset, with a range of [0.08 s, 44.13 s], a mean of 7.34 s and a median of 1.32 s. These figures indicate a long tail to the right, i.e., while most verification runs finish in under 2 s, there are also some significantly longer ones. On average, verification time gets lower for products that are verified later in the current product order of the permutation. In particular, average verification time decreases if capacities of bidders are reduced, as visible in Fig. 3. This phenomenon might occur because the model checker needs to check fewer markings in the Petri Nets once some products are assigned, and less capacity is available.



(a) Full dataset.

(b) Prediction dataset.

FIGURE 3. Distribution of verification time (excluding outlier points) for different capacities of individual bidders.

Verification time also depends on the property to be verified, i.e., the product, price, and potential winner. Further, there is a negative relationship between verification result and verification time. I.e., a positive verification result takes shorter to obtain on average than a negative one. However, the Pearson correlation, which quantifies linear dependencies, between these two variables is relatively weak, with a value of -0.22 in the full dataset. The Spearman correlation, which captures monotonic relationships, is moderate at best, with a value of -0.45 in the full dataset. Correlations in the prediction dataset are similar.

Finally, the dataset shows a nearly perfect correlation between verification time and two other features measuring the effort of the model checker: the number of markings generated as well as the number of transitions fired when verifying a property against a Petri Net, c.f. Appendix. However, as both these features are only known after running verification, i.e., when verification time is known in any case, we exclude these two features when training prediction models.

B. PREDICTING THE DATA

1) VERIFICATION RESULT

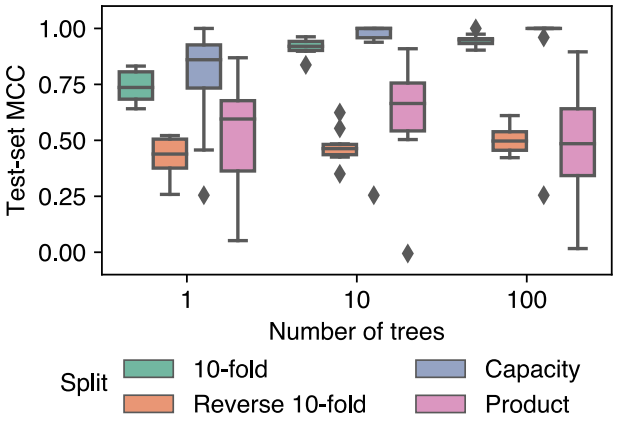
Fig. 4a shows how prediction models perform for the verification result as target variable. Prediction performance varies significantly between different splitting methods. In particular, the product-based split entails a comparatively low prediction performance with large variance, i.e., prediction models do not generalize well over products. This observation makes sense, as there are different budgets for each product, and thus the validity of prices and winners depends on the product. In contrast, prediction performance with capacity-based splits is higher on average. As Fig. 4b shows, capacity-based features are less important for predictions than features describing the property to be verified, i.e., the product, price, and winner. There still is one outlier fold with an MCC of just 0.25 for the capacity-based split, even if random forests with 100 trees are used. This observation indicates that prediction models might have difficulties to generalize to new capacity settings as well.

For a standard 10-fold random split, prediction performance is consistently good and improves with the number of trees in the random forest. This result is particularly encouraging when considering runtime: Even with 100 trees, training and prediction took less than a second on one split of the prediction dataset. For comparison, running verification to obtain this dataset took a few hours. This raises the question of how good prediction performance is with a small training dataset. For reverse 10-fold cross-validation, which only trains on 10% of the data instead of 90%, results are significantly worse than for standard 10-fold cross-validation, as Fig. 4a shows. However, at least the decrease in prediction quality is better than proportional, i.e., prediction quality does not drop to a ninth of its original value.

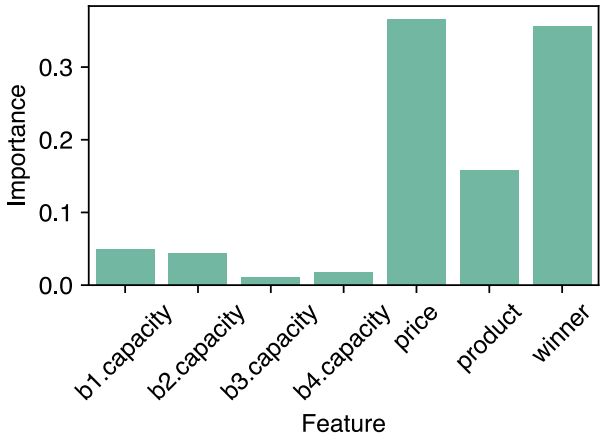
2) REVENUE

If one knows the verification result for all combinations of products, prices, and winners, one can also compute the revenue. However, as described in Section V-C, we take a different approach here, predicting the minimum revenue directly for each permutation of products in the iterative verification procedure. Similar to predicting verification result, prediction performance in standard 10-fold cross-validation is good, as Fig. 5a shows, using R^2 as quality metric. Again, prediction performance is significantly lower if one trains prediction models with just 10% of the data, but still considerably better than random guessing.

Fig. 5b displays feature importance for predicting revenue. The importance of the product position in the permutation significantly varies between products, but it is greater than zero for all products. This observation also explains why prediction performance with a split based on product positions is

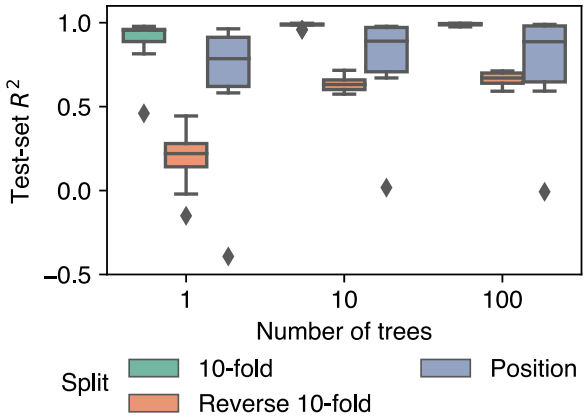


(a) Prediction performance for random forests.

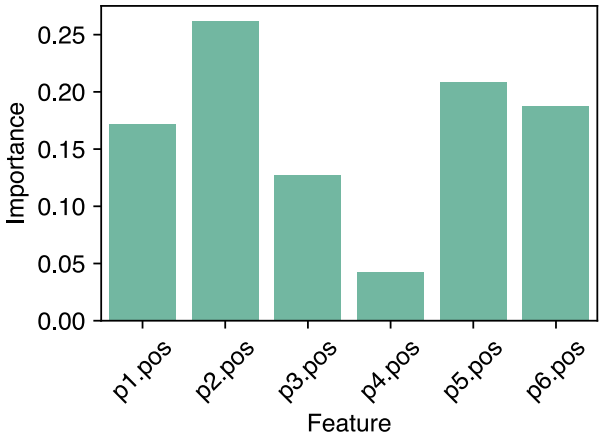


(b) Average feature importance for random forests with 100 trees in 10-fold cross-validation.

FIGURE 4. **Prediction results for verification result as target variable.**



(a) Prediction performance for random forests.



(b) Average feature importance for random forests with 100 trees in 10-fold cross-validation.

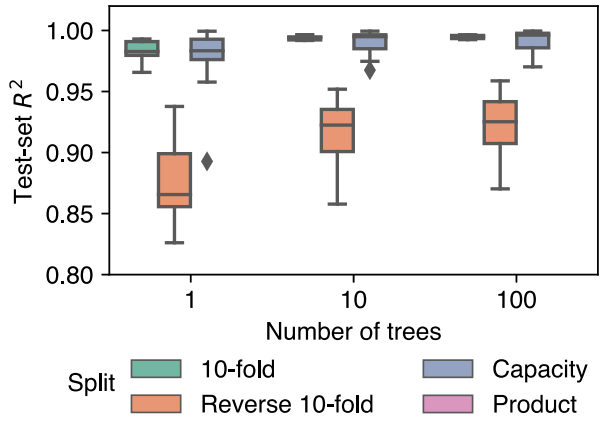
FIGURE 5. **Prediction results for revenue as target variable.**

lower than for a 10-fold random split, c.f. Fig. 5a. The position of Product 4 shows the lowest importance. This finding makes sense, as Product 4 only varies little in final prices and winner assignment, cf. Table 4, thus having a low influence on revenue. The position of Product 2 has the highest importance in predictions, which is surprising, given the low variation of the final price of this product. However, three bidders may win the product with roughly equal frequency. This situation might affect revenue indirectly because winner assignment reduces the capacities of the bidders and thus influences the prices of subsequently assigned products.

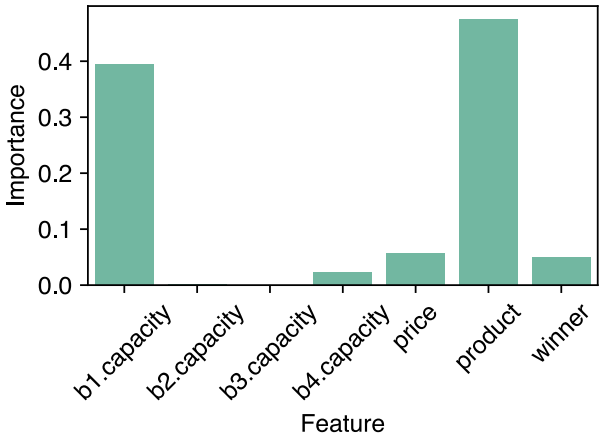
3) VERIFICATION TIME

As for the other prediction targets, prediction performance for verification time strongly depends on the splitting method. As Fig. 6a shows, the standard 10-fold random split yields stable and excellent prediction performance. Reverse 10-fold splitting yields worse and more fluctuating performance, but the drop compared to standard 10-fold is lower than for predicting verification result or revenue. The capacity-based split is good on average as well, but it has strong outliers, which are not visible in the plot. In particular, there are outliers with negative R^2, i.e., worse than the baseline strategy of constantly predicting the mean of the test data. For a product-based split, even the median R^2 is negative, and results do not appear in the plot. Thus, the trained models do not have any predictive power with that splitting method. As for verification result, this indicates that prediction models might have difficulties to generalize over different auction designs.

Fig. 6b display feature importance for predicting verification time. The product to be verified is most important,



(a) Prediction performance for random forests. Y-axis truncated for readability, thus no results for product-based split.



(b) Average feature importance for random forests with 100 trees in 10-fold cross-validation.

FIGURE 6. **Prediction results for verification time as target variable.**

while the other parts of the property to be verified, i.e., price and winner, are less important. In addition, the capacity of Bidder 1 plays an important role, clearly more than all other capacities. As already seen in Fig. 3, this bidder has the highest variation in median verification time between capacity levels. In particular, it makes a big difference for verification time if this bidder still has capacity left or if their capacity already is at zero.

VII. SUMMARY AND DISCUSSION

A. EXPLORATION (Q1), (Q2)

An exploratory analysis of the dataset showed that there is not a unique outcome, but many different outcomes could occur in the auction, cf. Table 4. In particular, each product had multiple potential winners and prices. Most importantly for the auctioneer, revenue varied between the outcomes and turned out to be suboptimal in the majority of the cases. A key factor for this variation was a capacity limit on the number of products each bidder can acquire. Thus, auctioneers should thoroughly analyze potential outcomes when introducing such capacity limits.

B. PREDICTION QUALITY (Q3) AND MODEL COMPLEXITY (Q6)

For a standard k-fold dataset split, we were able to predict the verification result, revenue, and verification time with high fidelity, c.f. Fig. 4a, 5a, and 6a. Unsurprisingly, more complex prediction models often performed better than less complex ones, though not for all prediction scenarios and splitting methods. Generally, as random forests captured the complex, non-linear dependencies in our dataset well, we would recommend them for further studies with auction data. If there are no hardware limitations, a high number of trees should be used.

C. GENERALIZATION (Q5)

From the domain perspective, a high prediction performance implies that one could replace costly verification runs with predictions, which were considerably faster. In particular, training and prediction of one model for one dataset split took less than one second, while obtaining the verification runs to build the dataset took a few hours. For the auction use case, this means the auctioneer could efficiently predict the least favorable outcomes and adapt the auction design accordingly.

However, prediction quality strongly depended on the splitting method in our study. This finding illustrates that the prediction models are rather dataset-specific. The currently trained models might show poor performance if the auction design changes, e.g., new capacity settings or new budgets are introduced. For example, the given dataset has constant budgets, and the prediction models thus can only learn which prices are valid for these budgets. This caveat highlights that one needs to carefully choose a representative training set when predicting verification results. For auctions, one should focus analysis on one auction design and systematically vary specific aspects of that design, e.g., capacities.

One way to obtain a dataset, if one cannot evaluate all combinations of auction-design aspects, is random sampling. To that end, we trained prediction models on only 10% of the dataset. Compared to 10-fold cross-validation, the drop in prediction performance depended on the prediction target. For random forests with 100 trees, the drop in prediction performance was 47% for verification result, 33% for revenue, and 7% for verification time. Depending on the use case, one needs to decide which prediction performance still is acceptable. We recommend starting predictions with a small set of verification runs and increasing dataset size as necessary. Besides random sampling, one could also use active learning

techniques [55] to decide which verification runs to execute next when building the dataset.

D. FEATURE IMPORTANCE (Q4)

Interestingly, the influence of the bidders' capacities on the prediction of verification result and verification time strongly varied between bidders, i.e., some bidders had more influence than others, c.f. Fig. 4b and 6b. Similarly, the importance of individual products on prediction of the revenue varied, c.f. Fig. 5b. Given the asymmetric distribution of budgets and capacities over products and bidders, this observation makes sense. For further studies, a broader analysis with more auction designs would be beneficial so that one could draw more general conclusions. For example, if certain bidders or products have a low influence in a scenario, this knowledge might help to optimize the process model or the verification procedure.

VIII. CONCLUSION AND FUTURE WORK

A. PROBLEM AND APPROACH

Verification can help to analyze outcomes of complex real-world processes like spectrum auctions. However, verification of properties in large process models is costly. To speed this up, one may create a dataset from prior verification runs and analyze this data. In particular, prediction models might allow efficiently estimating verification results or high-level properties of the process model. To that end, we conducted a case study on a large dataset from verifying the outcomes of a spectrum auction. The auction design was similar to the German 4G auction.

B. RESULTS

Our study first explored the dataset and found many different outcomes occurring, most of them suboptimal. Second, we trained prediction models for three targets, i.e., revenue, verification result, and verification time. We saw high prediction performance for evaluation with random splits of the dataset. However, we observed that prediction models generalized less if the splits excluded certain aspects of the auction design, e.g., capacities or products, from the training data. Thus, we advise making the training set of verification runs as representative as possible regarding future input data for predictions.

C. FUTURE WORK IN VERIFICATION

Our case study focused on a specific verification scenario, but the analysis approach is general. For any verification of data-aware processes, one can build a similar dataset by systematically running verification with many different process models and properties. As in our case, valuable features for predictions might originate from data values in the process models, the properties to be verified, and data from the verification process. Additionally, if one verifies properties for a large amount of Petri Nets, extracting features from the graph structure of the Petri Nets might be promising as well.

D. FUTURE WORK IN AUCTION THEORY

In the field of auction theory, our approach seems promising for industrial sales and procurement auctions, e.g., the auction for renewable energy support. In particular, wind offshore auctions are similar in complexity and design to the already analyzed German spectrum auction. In 2020, the United Kingdom (UK) auctioned at least 7 GW of wind offshore, divided into four bidding areas [56]. A capacity rule was included as well [56]. Although high prices were achieved [57], an improvement of the auction design might have led to even higher revenue.

APPENDIX
DATASET FEATURES

In the following, we provide a short description of each of the 30 columns (features) in the full dataset. As described in Section V-C, we only use subsets of these features for predictions.

- *id.product_permutation*: An integer in $[1, 720]$, identifying which of the $6! = 720$ product orders is currently verified.
- *id.iteration*: A positive integer, denoting the iteration number within the current *id.product_permutation*. *id.product_permutation* and *id.iteration* in combination uniquely identify rows in the dataset.
- *id.product_position*: An integer in $[1, 6]$, denoting the position of the currently verified product (*property.product*) in the product order of the current *id.product_permutation*.
- *id.product_case*: A positive integer, distinguishing repeated verification runs, all starting from the same lowest price, for the same *property.product* in the current *id.product_permutation*. This happens if there were multiple potential winners for the previously verified product – the iterative verification procedure checks the consequences of all these assignments in a breadth-first search, c.f. Section IV-D.
- *process.b[1-4].capacity*: An integer in $[0, 3]$, denoting the current capacities of the bidders. These are the only features in the dataset that represent data values from the process model. At the beginning of each *id.product_permutation*, capacities are the same, but once a winner for a product is known, the capacity of this bidder is reduced as soon as verification moves on to the next product.
- *property.formula*: A string representation of the property to be verified. We have extracted several numeric features to ease use for analysis and predictions, which we will describe next. In the formula itself, numerical values for prices and winners have a binary encoding, i.e., even if the formulas seem to contain multiple prices and winners at first glance, they refer to just one price and at most one winner at once.
- *property.product*: An integer in $[1, 6]$, denoting the currently verified product.

- *property.price*: An integer in [59, 90], denoting the price that is currently verified for the *property.product*.
- *property.winner*: An integer in [1, 4], denoting the bidder that is currently verified as winner for the *property.product* with the *property.price*. This feature is empty for iterations where the price is not clear yet.
- *verification.is_final*: A boolean, denoting whether prices and winners for all six products are found. In these rows, all *allocation.* features have a value; else, they are all empty.
- *verification.result*: A boolean, denoting if the current *property.formula* is satisfied in the underlying Petri Net or not.
- *verification.time*: A positive integer, denoting the time (in ms) for verifying the current *property.formula* against the underlying Petri Net. In the prediction dataset, this is a real number due to the creation procedure of that dataset, c.f. Section V-C2.a.
- *verification.markings*: A positive integer, denoting the number of markings generated when verifying the current *property.formula* against the underlying Petri Net.
- *verification.edges*: A positive integer, denoting the number of transitions fired when verifying the current *property.formula* against the underlying Petri Net.
- *allocation.revenue*: A positive integer in [430, 490], denoting the sum of final prices over all products.
- *allocation.p[1-6].price*: An integer in [60, 90], denoting the final price of the corresponding product.
- *allocation.p[1-6].winner*: An integer in [1, 4], denoting the bidder winning the corresponding product.

Additionally, for revenue prediction, we extract the features *order.p[1-6].pos*. These are integers in [1, 6], denoting the positions of Product 1 to Product 6 within the current *id.product_permutation*.

ACKNOWLEDGMENT

(Elaheh Ordoni and Jakob Bach contributed equally to the work).

REFERENCES

[1] T. W. Hazlett, R. E. Mu noz, and D. B. Avanzini, "What really matters in spectrum allocation design," *Nw. J. Tech. Intell. Prop.*, vol. 10, no. 3, pp. 93–123, 2012. [Online]. Available: https://scholarlycommons.law.northwestern.edu/njtip/vol10/iss3/2

[2] L. M. Ausubel, P. Cramton, and P. R. Milgrom, "The clock-proxy auction: A practical combinatorial auction design," in *Handbook of Spectrum Auction Design*. Cambridge, U.K.: Cambridge Univ. Press, 2017, ch. 6, pp. 120–140.

[3] J. H. Kagel and D. Levin, "Behavior in multi-unit demand auctions: Experiments with uniform price and dynamic vickrey auctions," *Econometrica*, vol. 69, no. 2, pp. 413–454, Mar. 2001.

[4] C. Brunner, J. K. Goeree, C. A. Holt, and J. O. Ledyard, "An experimental test of flexible combinatorial spectrum auction formats," *Amer. Econ. J., Microecon.*, vol. 2, no. 1, pp. 39–57, Feb. 2010.

[5] W. Vickrey, "Counterspeculation, auctions, and competitive sealed tenders," *J. Finance*, vol. 16, no. 1, pp. 8–37, 1961.

[6] V. Krishna, *Auction Theory*. New York, NY, USA: Academic, 2009.

[7] C. W. Smith, "Auctions: From Walras to the real world," in *Explorations in Economic Sociology*. New York, NY, USA: Russell Sage Foundation, 1993, ch. 7, pp. 176–192.

[8] O. Kirchkamp and J. P. Reiss, "Heterogeneous bids in auctions with rational and markdown bidders—Theory and experiment," Friedrich Schiller Univ. Jena Max Planck Inst. Econ., Jena, Germany, Tech. Rep. 2008,066, 2008. [Online]. Available: http://hdl.handle.net/10419/31716

[9] S. Gandhi, C. Buragohain, L. Cao, H. Zheng, and S. Suri, "A general framework for wireless spectrum auctions," in *Proc. 2nd IEEE Int. Symp. New Frontiers Dyn. Spectr. Access Netw.*, Apr. 2007, pp. 22–33.

[10] E. Ordoni, J. Mülle, and K. Böhm, "Verification of data-value-aware processes and a case study on spectrum auctions," in *Proc. CBI*, 2020, pp. 181–190.

[11] P. Cramton and A. Ockenfels, "The German 4G spectrum auction: Design and behaviour," *Econ. J.*, vol. 127, no. 605, pp. F305–F324, Oct. 2017.

[12] G. Brero, B. Lubin, and S. Seuken, "Combinatorial auctions via machine learning-based preference elicitation," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 128–136. [Online]. Available: https://www.ijcai.org/proceedings/2018/0018.pdf

[13] M. J. García Rodríguez, V. Rodríguez Montequín, F. Ortega Fernández, and J. M. Villanueva Balsera, "Bidders recommender for public procurement auctions using machine learning: Data analysis, algorithm, and case study with tenders from Spain," *Complexity*, vol. 2020, pp. 1–20, Nov. 2020.

[14] M. Mohri and A. M. Medina, "Learning theory and algorithms for revenue optimization in second-price auctions with reserve," in *Proc. ICML*, 2014, pp. 262–270. [Online]. Available: http://proceedings.mlr.press/v32/mohri14.pdf

[15] S. Verwer, Y. Zhang, and Q. C. Ye, "Auction optimization using regression trees and linear models as integer programs," *Artif. Intell.*, vol. 244, pp. 368–395, Mar. 2017.

[16] M. Amrani, L. Lúcio, and A. Bibal, "ML+FV = ♡? a survey on the application of machine learning to formal verification," 2018, *arXiv:1806.03600*.

[17] E. E. Mandouh and A. G. Wassal, "Estimation of formal verification cost using regression machine learning," in *Proc. IEEE Int. High Level Design Validation Test Workshop (HLDVT)*, Oct. 2016, pp. 121–127.

[18] M. Yasrebi, V. Rafe, H. Parvin, and S. Nejatian, "An efficient approach to state space management in model checking of complex software systems using machine learning techniques," *J. Intell. Fuzzy Syst.*, vol. 38, no. 2, pp. 1761–1773, Feb. 2020.

[19] W. Zhu, H. Wu, and M. Deng, "LTL model checking based on binary classification of machine learning," *IEEE Access*, vol. 7, pp. 135703–135719, 2019.

[20] P. Milgrom, "Putting auction theory to work: The simultaneous ascending auction," *J. Political Economy*, vol. 108, no. 2, pp. 245–272, Apr. 2000.

[21] F. Gul and E. Stacchetti, "Walrasian equilibrium with gross substitutes," *J. Econ. Theory*, vol. 87, no. 1, pp. 95–124, Jul. 1999.

[22] S. Seifert and K.-M. Ehrhart, "Design of the 3G spectrum auctions in the U.K. and germany: An experimental investigation," *German Econ. Rev.*, vol. 6, no. 2, pp. 229–248, May 2005.

[23] M. Bichler, V. Gretschko, and M. Janssen, "Bargaining in spectrum auctions: A review of the German auction in 2015," *Telecommun. Policy*, vol. 41, nos. 5–6, pp. 325–340, Jun. 2017.

[24] Z. Chen, L. Huang, L. Li, W. Yang, H. Miao, M. Tian, and F. Wang, "PS-TRUST: Provably secure solution for truthful double spectrum auctions," in *Proc. INFOCOM*, vol. 2014, pp. 1249–1257.

[25] M. Al-Ayyoub and H. Gupta, "Truthful spectrum auctions with approximate revenue," in *Proc. INFOCOM*, Apr. 2011, pp. 2813–2821.

[26] S. Ghilardi, A. Gianola, M. Montali, and A. Rivkin, "Petri nets with parameterised data: Modelling and verification (extended version)," 2020, *arXiv:2006.06630*.

[27] D. Knuplesch, L. T. Ly, S. Rinderle-Ma, H. Pfeifer, and P. Dadam, "On enabling data-aware compliance checking of business process models," in *Proc. ER*, 2010, pp. 332–346.

[28] H. Groefsema, N. R. T. P. van Beest, and A. Armas-Cervantes, "Efficient conditional compliance checking of business process models," *Comput. Ind.*, vol. 115, Feb. 2020, Art. no. 103181.

[29] A. Meyer and M. Weske, "Data support in process model abstraction," in *Proc. ER*, vol. 2012, pp. 292–306.

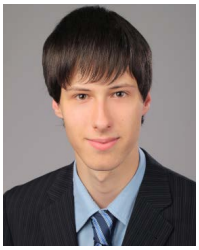
[30] S. Haarmann, K. Batoulis, and M. Weske, "Compliance checking for decision-aware process models," in *Proc. BPM*, 2018, pp. 494–506.

[31] R. Mrasek, J. Mülle, and K. Böhm, "A new verification technique for large processes based on identification of relevant tasks," *Inf. Syst.*, vol. 47, pp. 82–97, Jan. 2015.

[32] J. Mülle, C. Tex, and K. Böhm, "A practical data-flow verification scheme for business processes," *Inf. Syst.*, vol. 81, pp. 136–151, Mar. 2019.

[33] R. Bobrik, M. Reichert, and T. Bauer, "View-based process visualization," in *Proc. BPM*, 2007, pp. 88–95.
[34] X. Lai, A. Balakrishnan, T. Lange, M. Jenihhin, T. Ghasempouri, J. Raik, and D. Alexandrescu, "Understanding multidimensional verification: Where functional meets non-functional," *Microprocessors Microsyst.*, vol. 71, Nov. 2019, Art. no. 102867.
[35] E. M. Elmandouh and A. G. Wassal, "Guiding formal verification orchestration using machine learning methods," *ACM Trans. Design Autom. Electron. Syst.*, vol. 23, no. 5, pp. 1–33, Oct. 2018.
[36] D. Buchs, S. Klikovits, A. Linard, R. Mencattini, and D. Racordon, "A model checker collection for the model checking contest using Docker and machine learning," in *Proc. Petri Nets*, 2018, pp. 385–395.
[37] S. Matsubara, S. Yamaguchi, and M. A. Bin Ahmadon, "Generating and analyzing data set of workflow-nets," in *Proc. 8th Int. Symp. Comput. Netw. Workshops (CANDARW)*, Nov. 2020, pp. 471–473.
[38] K. Pei, Y. Cao, J. Yang, and S. Jana, "Towards practical verification of machine learning: The case of computer vision systems," 2017, *arXiv:1712.01785*.
[39] W. Xiang, P. Musau, A. A. Wild, D. Manzanas Lopez, N. Hamilton, X. Yang, J. Rosenfeld, and T. T. Johnson, "Verification for machine learning, autonomy, and neural networks survey," 2018, *arXiv:1810.01989*.
[40] M. Weiss, B. Lubin, and S. Seuken, "SATS: A universal spectrum auction test suite," in *Proc. AAMAS*, 2017, pp. 51–59. [Online]. Available: https://www.ifaamas.org/Proceedings/aamas2017/pdfs/p51.pdf
[41] P. Milgrom, *Putting Auction Theory to Work*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[42] A. M. Kwasnica and K. Sherstyuk, "Multiunit auctions," *J. Econ. Surv.*, vol. 27, no. 3, pp. 461–490, Jul. 2013.
[43] M. P. Van der Aalst, "The application of Petri nets to workflow management," *J. Circuits Syst. Comput.*, vol. 8, no. 1, pp. 21–66, Feb. 1998.
[44] E. M. Clarke, E. A. Emerson, and A. P. Sistla, "Automatic verification of finite-state concurrent systems using temporal logic specifications," *ACM Trans. Program. Lang. Syst.*, vol. 8, no. 2, pp. 244–263, 1986.
[45] E. M. Clarke, *Model Checking*. Cambridge, MA, USA: MIT Press, 2018.
[46] N. Lohmann, E. Verbeek, and R. Dijkman, "Petri net transformations for business processes—A survey," in *Transactions on Petri Nets and Other Models of Concurrency II*. Berlin, Germany: Springer, 2009, pp. 46–63.
[47] A. Wald, "Contributions to the theory of statistical estimation and testing hypotheses," *Ann. Math. Stat.*, vol. 10, no. 4, pp. 299–326, 1939. [Online]. Available: https://www.jstor.org/stable/2235609
[48] J. von Neumann and O. Morgenstern, *Theory of Games and Economic Behavior*. Princeton, NJ, USA: Princeton Univ. Press, 1944.
[49] M. D. Resnik, *Choices: An Introduction to Decision Theory*. Minneapolis, MI, USA: Univ. Minnesota Press, 1987.
[50] M. Sniedovich, "Wald's maximin model: A treasure in disguise!" *J. Risk Finance*, vol. 9, no. 3, pp. 287–291, May 2008.
[51] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.
[52] B. W. Matthews, "Comparison of the predicted and observed secondary structure of t4 phage lysozyme," *Biochimica Biophys. Acta-Protein Struct.*, vol. 405, no. 2, pp. 442–451, 2016, doi: 10.1016/0005-2795(75)90109-9.
[53] G. James, D. Witten, T. Hastie, and R. Tibshirani, "Linear regression," in *An Introduction to Statistical Learning: With Applications in R*. New York, NY, USA: Springer, 2013, ch. 3, pp. 59–126.
[54] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, and P. Prettenhofer, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011. [Online]. Available: http://jmlr.org/papers/v12/pedregosa11a.html
[55] B. Settles, "Active learning literature survey," Dept. Comput. Sci., Univ. Wisconsin-Madison, Madison, WI, USA, Tech. Rep. 1648, 2009. [Online]. Available: http://digital.library.wisc.edu/1793/60660
[56] The Crown Estate. (2019) *Information Memorandum: Introducing Offshore Wind Leasing Round 4*. Accessed: Jul. 29, 2021. [Online]. Available: https://www.thecrownestate.co.U.K./media/3321/tce-r4-information-memorandum.pdf
[57] S. Twidale. (2021). *RWE, Total, BP Among Winners in UK Offshore Wind Farm Auction*. Accessed: Jul. 29, 2021. [Online]. Available: https://www.reuters.com/article/us-britain-windpower-auction-idUSKBN2A80RN

ELAHEH ORDONI received the M.Sc. degree in algorithms and computations from the University of Tehran, Tehran, Iran, in 2017. She is currently pursuing the Ph.D. degree with the Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany. Her research interest includes supporting data in workflows to guarantee the correctness of business process models.



JAKOB BACH received the B.Sc. degree in industrial engineering and management and the M.Sc. degree in informatics from the Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany, in 2015 and 2017, respectively, where he is currently pursuing the Ph.D. degree in informatics. His research interests include data science and feature selection.

ANN-KATRIN FLECK received the degree in mathematics from the Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany. She is currently pursuing the Ph.D. degree (since 2018) and analyzes measures to reduce uncertainties in procurement auctions, especially for renewable energy. She is a consultant with Takon GmbH with a counseling focus on strategic decisions. Her work includes consulting the German Federal Ministry of Economic Affairs and Energy to design auctions for renewable energy support and companies in industrial procurement auctions. She is part of the consortium of the EU grant Horizon2020 project AURES II where she game-theoretically and experimentally analyzes auctions for renewable energy support.