Data driven design for online industrial auctions

Data driven design for online industrial auctions

Qing Chuan Ye[1] , **Jason Rhuggenaath**[2] , **Yingqian Zhang**[3] , **Sicco Verwer**[4] and **Michiel Jurgen Hilgeman**[5]

[1]Eramus University Rotterdam
[2,3]Eindhoven University of Technology
[4]Delft University of Technology
[5]Troostwijk Auctions
ye@ese.eur.nl, {j.s.rhuggenaath, yqzhang}@tue.nl, s.e.verwer@tudelft.nl,
m.hilgeman@troostwijkauctions.com

Abstract

Designing auction parameters for online industrial auctions is a complex problem due to highly heterogeneous items. Currently, online auctioneers rely heavily on their experts in auction design. In this paper, we propose a data driven auction design framework that seamlessly combines prediction models and knowledge from experts into an optimization model. We show the proposed data driven approach improves upon the design from the experts for starting prices and display positions of items.

1 Introduction

Internet auctions have been upcoming since the early 2000s and most traditional auctions have also been converted into internet auctions due to the growth and widespread use of the internet, and the ease of bidding from the comfort of your own home. Industrial auctions are a mean for companies to sell their assets and inventories. Online auctioneers try to set up the best possible auctions using several auction parameters, such as presentation of the lots, timing of the auctions and the starting prices.

Classical economic theory such as [Myerson, 1981; Milgrom and Milgrom, 2004] provide some guideline on auction mechanism frameworks. However, they work under strong assumptions about rationality and valuation function of bidders. As many studies showed, bidders often behave irrationally due to several psychological phenomena. [Ku *et al.*, 2005] call the phenomenon of irrational and frantic behavior of bidders which lead to overbidding "auction fever". Participants at auctions enjoy the thrill of winning [Cheema *et al.*, 2012]. [Heyman *et al.*, 2004] describe the opponent effect that is "an increase in the subjective value of winning the auction when the behavior of other bidders in the auctions is perceived to be competitive." Another phenomenon is that bidders experience a feeling of ownership when they have the highest bid during an auction. This feeling of temporally ownership during an auction is named "pseudo-endowment effect" in [Ariely and Simonson, 2003] and "quasi-endowment" in [Heyman *et al.*, 2004], which leads to a higher end price.

In practice, auctioneers rely on their experience and empirical analysis in the auction literature to tune auction parameters for daily operations. For example, they often define starting prices low to make the auctioned items attractive to a wide audience, but for items that are harder to sell, the starting prices are set relatively high. This pricing strategy is supported by [Ku *et al.*, 2006] which states that lower starting prices reduce the barrier to bid, but lead to lower end prices when the market entry and participation are reduced.

As many auctions are conducted throughout the years, data is collected on the characteristics of auctions and the bids that are received. This historical data can give insights into how the auctions are performing, and machine learning methods could help in learning relations between characteristics of auctions, auction parameters, and auction outcomes. Existing research has typically focused on one specific kind of item with very similar specifications (e.g., [Van Heijst *et al.*, 2008]), on predicting outcome for an on-going auction (e.g., [Zhang *et al.*, 2010]), on predicting revenue with simulated auction data (e.g., [Verwer and Zhang, 2012]), or on using reinforcement learning for auction mechanism design (e.g., [Tang, 2017]). There is a lack of work on helping online industrial auctioneers set up auction parameters using machine learning models, probably due to the fact that there is a wide variety of items put on sale in online industrial auctions.

The ability of predicting how well an auction will perform prior to the start comes in handy for auctioneers. If an item is expected to be a low-performing item, the auctioneer can take certain actions to influence the auction outcome. For instance, the starting selling price of the item can be modified, or the location where the item is displayed on the website can be changed to attract more attention. In this paper, we take a real-world industrial auction data set and investigate how we can improve upon the expert's design using insights learned from data. More specifically, we first construct a classification model that predicts the expected performance of auctions. We propose a data-driven auction design framework (called DDAD) that combines the expert's knowledge with the learned prediction model, in order to find the best parameter values, i.e., starting price and display positions of the items, for a given new auction. The prediction model is evaluated, and the new design for several auctions is discussed and validated with the auction experts.

Our contributions are as follows:

- We propose an optimization framework that seamlessly integrates prediction model components, i.e., constructed constraints between features and their relation to the predictions, with expert knowledge and domain constraints in a mathematical optimization model.
- Interestingly, our results demonstrate that despite the high variety of items and prices in industrial online auctions, simple classification models such as decision trees are able to predict the auction outcomes quite accurately, using expert knowledge and popularity of items as features.
- We show that the new design, which was validated by the auction experts, improves the expected revenue of the online auctioneer, evaluated by the classification model.

We start with describing the auction setting and the auction optimization problem we aim to solve.

2 Auction setting and problem formulation

The online auction company conducts many auctions throughout the year, selling many different kinds of items. The items that are auctioned cover a wide range, from kitchenware to farming equipment and building contractors equipment. Each item that is being sold through the auction is called a lot. A collection of lots from one or more sellers is called a sale. A sale can be categorized in one of four categories: bank, dealer, liquidator, or volunteer. The items get appraised by experts and receive an estimated value (*EstValue*), along with a starting price for the lot, and they are assigned to a main and subcategory within the sale. A lot can also be put on allocate, which means that the seller has set a reserve price, and the lot will not be sold when the winning bid is lower than the reserve price. Lots are assigned a lot number, which is a unique number within the sale and indicates where the lot will be shown in the sale on the website, as the lots are automatically sorted on lot number. The lower the lot number, the higher the lot will appear on the website. Sales are announced through their website well in time. Once an auction starts, bidders typically have a few weeks to bid on the lots. The auction is a form of online *English auction*, where the auctioneer opens the auction of a lot with a starting price, and buyers place increasingly higher bids. The item is sold to the highest bidder at a price equal to his or her bid.

The highest bid, or *EndPrice*, on its own does not say much about the success of a lot in an auction, if one does not know what kind of lot it was. Two lots might have the same EndPrice, but one could be considered a cheap item, whereas the other could be considered expensive. Therefore, the *multiplier* is used by the auctioneer as an auction performance indicator. The multiplier ties the EndPrice and the EstValue (see Table 1) together by taking the ratio between the two, i.e.,

$$multiplier = \frac{EndPrice}{EstValue}.$$

The multiplier therefore indicates how well an item has performed in the auction compared to the expectations of the expert. Based on expert opinion (and business needs), we choose to split the lots into three classes: (1) a multiplier equal to 0 is in the unsold class 0, or simply *unsold*; (2) a multiplier higher than 0 and lower than 0.8 denotes the low-performing class (*low*); (3) a multiplier of 0.8 or higher denotes the class with expected performance (*high*).

The auctioneer is particularly interested in items that may not be sold or may be sold for a lower price than expected. If an item is expected to not perform well, the auctioneer can take certain actions to update the auction parameters before the auction starts to influence the auction outcome. The important auction parameters include for example the starting prices of items and their display positions.

The auction optimization problem is defined as follows. Given a sale of N items or lots, determine the starting price and the lot number (i.e., display position) for each item, such that the expected outcome is maximized.

3 Auction data and feature engineering

We are provided with data collected over ten months. From this data, we select a subset of the data from the following branches: construction, agricultural industry and consumer. The main motivation for this subset is the fact that these branches make up about 75% of total revenue. In summary, Table 1 shows an overview of the available variables in the database of the auction company.

Variable	Type	Description
LotNr	ordinal	unique ID and position of a lot within a sale
Allocate	binary	lot has a reserve price
EstValue	numeric	estimated value of the lot
StartPrice	numeric	starting price of the lot
EndPrice	numeric	price at which the lot is sold
Seller	ordinal	type of seller
CloseTime	ordinal	time of day the sale closes
Weekday	ordinal	day of the week the sale closes

Table 1: Description of the variables in the data and their respective types.

The data set consists of a total of 24,451 lots. There are 18,528 lots in the EstValue range of $[0, 200)$. This means that there are many lots consisting of small items with a low estimated value. The multiplier is very sensitive to small variations in the end price when the estimated value is small. We would also like to focus on predicting the larger lots, as these are more important to predict correctly. Therefore, we filter out lots with an estimated value of lower than 200. Furthermore, there are 80 lots with an estimated value of 20,000 and higher. As these lots are rare and should be treated by experts as special cases, we filter out these lots as well. We end up with a total of 5,843 lots. Figures 1, 2 and 3 show the distribution of the filtered lots over different ranges of estimated value, starting price and end price, respectively. Figure 2 shows that the starting price of lots is lower in general than its estimated value, as the graph shows a shift to the left compared to Figure 1. The lots with an estimated value of in between 200 and 300 mostly have a starting price of in between 100 and 200, although there are 146 lots that have an even lower starting price of lower than 100. In general, lots have starting prices approximately 0.5 or 0.67 times their estimated value. In Figure 3 we can see that about half of the lots have an end price lower than 500. This coincides with Figure 1, which has roughly half of the lots with an estimated value of lower than 500. This would indicate that the experts' estimates are generally quite okay. However, the end price of about a quarter of these lots are lower than 200, whereas the estimated value is 200 or higher. This shows that experts are overestimating the value on relatively cheaper lots. For the other lots, the end price coincides roughly with the estimated value. This of course does not mean that the experts are correct with their estimates all the time, but they are performing quite well on average. For all these features we can say that there are many lots in the lower regions, whereas the feature values get higher, the number of lots decreases, creating a long tail. This means that the models will be trained more heavily on the lots with lower values, whereas they will be more susceptible to variance once the feature values get higher. This again shows that we are dealing with a large variety of items, and it will therefore be difficult to construct a model that will perform well on every item in the auction.

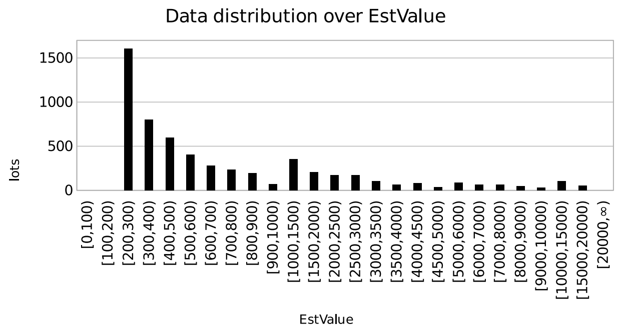

Figure 1: Data distribution over various ranges of estimated value.

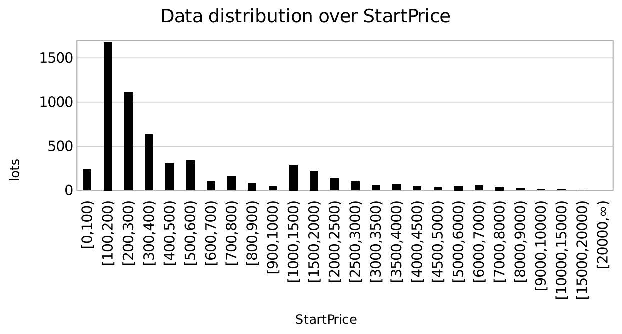

Figure 2: Data distribution over various ranges of starting price.

Feature construction and selection. In a similar vein as defining the multiplier, the starting price of a lot can be tied to its estimated value. Therefore, we introduce the variable *SPEV*, defined as

$$SPEV = \frac{StartPrice}{EstValue},$$

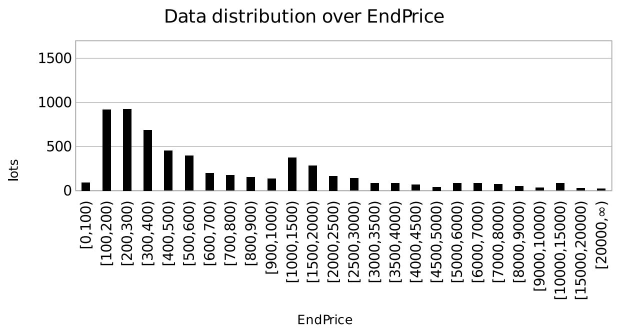

Figure 3: Data distribution over various ranges of end price.

which indicates how far the starting price is from the estimated value of the lot initially. In addition, auctioneers deem to believe that scarcity is important. The simple economic rule of supply and demand is valid. If supply is lower, the price gets higher. If an auctioneer offers one unique item, all attention and demand are focused on this specific item. When the auctioneer offers two similar items, the attention and the number of bidders are spread out, which results in less bidding activity and therefore lower prices. Hence, we create additional features LotsSale, LotsSaleMain, and LotsSaleSub. LotsSale indicates the total number of lots within the same sale. LotsSaleMain and LotsSaleSub present an even deeper level of scarcity, where LotsSaleMain shows the number of lots within the same main category within the same sale, whereas LotsSaleSub shows the number of lots within the same subcategory of that main category. We use all these constructed features together with the variables described in Table 1, with the exception of EndPrice and multiplier, as *feature variables*.

In order to find out the relation between the multiplier and the feature variables, we conduct a correlation test on the data set. We make use of Spearman's rank correlation. The correlation between the multiplier and the feature variables in the data set with all lots, in which unsold lots are assigned a multiplier of 0, and with just the sold lots can be found in Table 2.

Feature	multiplier (all)	multiplier (sold)
LotNr	-0.08	-0.12
Allocate	-0.34	-0.03
EstValue	-0.07	0.03
StartPrice	0.06	0.16
Seller	-0.11	-0.08
CloseTime	-0.05	-0.08
Weekday	-0.02	-0.05
SPEV	0.37	0.33
LotsSale	0.03	-0.03
LotsSaleMain	-0.06	-0.07
LotsSaleSub	-0.08	-0.12

Table 2: Spearman's rank correlation between the multiplier and feature variables over the entire data set, and over only the sold lots.

In both sets, the feature variable that stands out with a positive correlation with the multiplier is SPEV ($0.37/0.33$). The positive correlations between the multiplier and the StartPrice and SPEV indicate that higher starting prices result in higher multipliers, with SPEV being a better indicator. This is in line with the findings of [Ariely and Simonson, 2003], who remarks that a low starting price may be used to attract more bidders, but will not necessarily result in a higher end price. As the correlations between the multiplier and EstValue ($-0.07/0.03$) seem to be rather low and not consistent, higher valued lots will not necessarily result in higher multipliers, but there are cases in which the starting price is relatively high compared to the estimated value, which will result in a higher multiplier. In addition, LotsSaleMain ($-0.06/-0.07$) and LotsSaleSub ($-0.08/-0.12$) stand out with negative correlations. This indicates that the fewer lots there are of the same main and subcategory within the same sale, the higher the multiplier will be. This coincides with the intuition that scarcity will make the lot more wanted, and therefore will generate a higher multiplier. LotsSale ($0.03/-0.03$) does not seem to have the same effect, as it might be too generic, as there are items within the same sale that can be very different from each other. Furthermore, the negative correlations of LotNr ($-0.08/-0.12$) indicate that the lower the LotNr, which means that it is higher on the page, the higher the multiplier will be. This is also as expected, as more bidders will see the lot when it is higher up on the page. With Seller ($-0.11/-0.08$) we see that sales from banks and liquidators result in slightly higher multipliers than sales from dealers and volunteers, which also matches the auctioneer's expectations. CloseTime ($-0.05/-0.08$) and Weekday ($-0.02/-0.05$) seem to indicate that sales ending earlier on a day and in the week have a slightly higher multiplier, but the effect is small. Allocate ($-0.34/-0.03$) has a significant difference in correlation between all lots and just the sold lots. This is because lots on allocate have a reserve price, which result in unsold lots with a multiplier of 0. When only considering sold lots, there seems to be barely any difference between lots being on allocate and not.

We include the following features in our prediction models: SPEV, StartPrice, EstValue, LotsSale, LotsSaleMain, LotsSaleSub, LotNr, Weekday and Allocate. The features SPEV, StartPrice, EstValue are included because relationship between the starting price relative to the estimated value is important when auctioneers optimize the auction design. LotsSale, LotsSaleMain, LotsSaleSub are selected since they jointly capture the effect of scarcity. LotNr is selected because it is related to the visibility of a lot. Weekday is selected in consultation with auction experts in order to capture potential time-effects. Allocate is selected since it can potentially predict when a lot will be sold or not. Finally, note that SPEV, StartPrice, and LotNr are design parameters that are directly controlled by the auctioneer.

Other feature selection methods such as recursive feature selection yielded similar results. As some features are auction design variables that we will optimize after building predictive model, feature selection methods based on for example PCA are not preferred.

4 Auction design optimization model

We treat the problem of evaluating the expected performance of auctioning items in the new sale as a classification problem. In the literature, there are various ways of utilizing the prediction models for optimizing new designs. For example, in [Gabel and Riedmiller, 2008] and [Huyet, 2006], predictions are used as fitness functions to evaluate the quality of a solution. Another line of research, called Empirical Model Learning [Lombardi *et al.*, 2017], investigates embedding of the components of prediction models into combinatorial models. See [Lombardi and Milano, 2018] for an overview. In [Verwer *et al.*, 2017], the authors encode the prediction model using Mixed Integer Linear Programming (MILP) language to find optimal auction sequence in a sequential auction using simulated action data. We extend the encoding of [Verwer *et al.*, 2017] to combine with the expert and domain knowledge in the MILP model. Our proposed auction design optimization framework DDAD is illustrated in Figure 4, where historical auction data is used to build a prediction model. A mathematical optimization model, i.e., integer linear programming (ILP) model, is defined to solve the given auction optimization problem. This ILP model takes into account both (1) the domain and expert knowledge on variables and objectives, and (2) the internal structure of the prediction model, i.e., the learned relations of different variables to predictions. We use classification trees as prediction models, and the translation from the learned classification tree to a set of linear constraints is extended from [Verwer *et al.*, 2017].

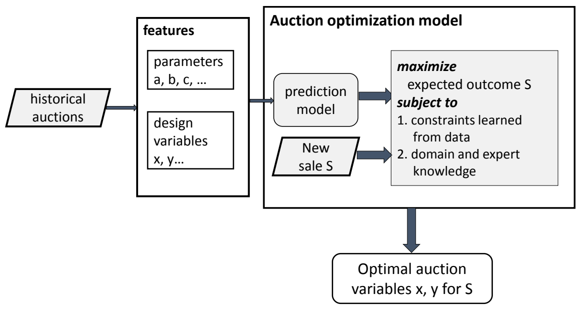

Figure 4: Components of the data driven auction design model.

Decision variables. Given a set I of N lots, we use a decision variable s_r to denote the starting price of lot $r \in I$, and the following variables to encode any possible index of lots: $x_{i,r} \in \{0, 1\}$. Lot r is given an index of i, $1 \le i \le N$, if and only if $x_{i,r} = 1$. Thus, if $x_{3,1}$ is equal to 1, it means that the first lot is assigned 3 as its lot id. Each lot has one lot id, and every lot is required to have a unique lot id. Hence,

$$\begin{array}{rcll} \sum_{1\le r\le N} x_{i,r} & = & 1 & \text{for all } 1 \le i \le N \\ \sum_{1\le i\le N} x_{i,r} & = & 1 & \text{for all } 1 \le r \le N \end{array}$$

Any assignment of ones and zeros to the x variables that satisfies these two types of constraints corresponds to a valid lot id assignments of all lots. The starting prices s_r have the following bounds, based on the advice from auction experts: $0.4 \times EV_r \le s_r \le 1.0 \times EV_r$, for all $r \in I$.

Computing feature values. Let $Feat$ be the set of features that are used to build the classification tree, i.e., LotNrRel, Allocate, EstValue, StartPrice, LotsSale, LotsSaleMain, LotsSaleSub, Weekday, SPEV. We use F ($F \subseteq Feat$) to denote the features that are used in the tree. The features LotNrRel, StartPrice, SPEV are related to our decision variables, and hence if they are in F, their values need to be translated as follow.

$$\begin{array}{rcll} \text{StartPrice}_r & = & s_r & \text{for } r \in I \\ \text{LotNrRel}_r & = & \frac{\{i \mid x_{i,r}=1\}}{N} & \text{for } r \in I \\ \text{SPEV}_r & = & \frac{s_r}{EV_r} & \text{for } r \in I \end{array}$$

Objective function. We denote the predicted class of lot r by binary variables $p_{r,c}$, where $c \in C = \{0, 1, 2\}$ is the class label. $p_{r,0} = 1$ indicates that lot r is predicted to be in the lowest performing class 0. The objective of the auction design is to maximize the expected performance of all lots in the sale, i.e.,

$$\max \textstyle\sum_{1 \le r \le N} \sum_{c \in C} c \cdot p_{r,c}$$

Every lot r can only end up in one class, i.e.,

$$\textstyle\sum_{c \in C} p_{r,c} = 1 \text{ for all } r \in I$$

Encoding classification tree We translate the classification tree models into ILP using linear constraints based on the encoding in [Verwer *et al.*, 2017]. We introduce a set of binary variables $z_{l,r}$, representing whether a leaf node l is reached for lot r. The internal (decision) nodes of the trees can be represented implicitly by the constraints on these new z variables. Intuitively, we encode that a z variable has to be false when the binary test of any of its parent nodes fails. By additionally requiring that exactly one z variable is true at every index, we fully encode the learned trees.

Let D be the set of all decision nodes in the classification tree. Every decision node in D contains a Boolean constraint $f \le t$, which is true if and only if feature f has a value less than or equal to a constant t. A key insight of our encoding is that every such Boolean constraint directly influences the value of several z variables: if it is true, then all z variables representing leafs in the right subtree are false; if it is false, then all that represent leafs in the left subtree are false. In this way, we require only two constraints per Boolean constraint in order to represent all possible paths to leaf nodes.

$$\begin{array}{ll} \mathtt{fv}_f + (M_f - c) \cdot \sum_{l \in L} z_{l,r} \le M_f & \text{for all } r \in I, (f \le t) \in D \\ \mathtt{fv}_f + (m_f - c) \cdot \sum_{l \in L'} z_{l,r} \ge m_f & \text{for all } r \in I, (f \le t) \in D \end{array}$$

where $\mathtt{fv}_f$ is a calculation of feature f's value, L and L' are the leaf nodes in the left and right subtrees of the decision node with constraint $(f \le t)$ in the tree, and M_f and m_f are the maximum and minimum values of feature f. For the feature calculation we simply replace $\mathtt{fv}_f$ with the right-hand sides of the corresponding feature definitions.

The above constraints ensure that when $z_{l,r}$ obtains a value of 1, all of the binary test in the parent nodes on the path to l in the tree for lot r return true. By construction of the trees, this ensures that at most one z variable is true for every r.

$$\textstyle\sum_l z_{l,r} \quad = \quad 1 \quad \text{for all } r \in I$$

The predictions of the trees are given by the z variable that is true. We multiply this z variable with the class prediction in the leaf node it represents to obtain the prediction, and store it in the p variables used to compute the objective value.

$$p_{r,c} = \textstyle\sum_{l \in L_r} v_l \cdot z_{l,r} \quad \text{for all } r \in I$$

where v_l is the constant prediction of leaf l in the tree.

In this way, we build a mathematical optimization model DDAD based on data and domain knowledge. This model can be solved by any off-the-shelf optimization solvers like CPLEX [IBM, 2018] and GUROBI [Gurobi Optimization, 2018], and guarantees to find the optimal decision variables, i.e., starting prices and lot numbers, for items in the new sale, such that the expected performance is maximized.

5 Results

We use the provided data to investigate how the DDAD performs. First, we select the data that we will use in our models, and we construct a training and test set. Then we use existing tree based classifiers to predict the multiplier of lots using the feature variables. A variety of different classification models with different parameters are used to observe the performance on our data. Thereafter, one classification model will be chosen to be used with DDAD. We use Scikit-learn in Python for the classification models, and we use CPLEX to solve DDAD.

5.1 Classification model

We first randomize the data set in order to obtain representative training and test data sets. Because the data set consists of multiple sales made up of numerous lots, we have to make sure that the sales stay intact. We randomly shuffle the sales and then use the first 70% of the 5,843 lots as our training data set, and the remaining 30% as our test data set. As the sales within the training and test data sets also have to remain intact, we do not cut off the training data set at exactly 70% of the total data, but we also include all the lots in the last sale of the training data set. Finally, we balance the training data set by undersampling lots with class label 2 such that the total number of lots in class 1 and 2 are equal. This results in a training data set of 3,087 lots (45 sales, 58.4%), and a test data set of 1,747 lots (32 sales, 41.5%). Table 3 shows the number of lots in each class. We also used other sampling alternatives for class balancing, but these did not give very different results.

Multiplier	Category	#Lots Train	#Lots Test
0	low	237	46
$(0, 0.80)$	med	1425	690
$[0.80, \infty)$	high	1425	1011
Total lots		3087	1747

Table 3: Number of lots in the low-, medium-, and high-performing classes in the training and test set.

We perform k-fold cross validation on the training data for different classification models with varying parameters in order to find the best classification model. We need to account for the fact that the data is ordered in sales and time. Therefore, we shuffle the training data set before applying k-fold

	Accuracy		Train	accuracy per class				Test	accuracy per class			
	mean	var (+/-)	accuracy	low	med	high	kappa	accuracy	low	med	high	kappa
CART3	0.64	0.05	0.64	0.94	1.00	0.23	0.36	0.56	1.00	0.99	0.25	0.26
CART5	0.67	0.04	0.69	0.97	0.64	0.69	0.45	0.66	0.61	0.36	0.87	0.28
CART7	0.69	0.06	0.73	0.97	0.76	0.66	0.53	0.59	0.63	0.70	0.52	0.24
CART10	0.72	0.05	0.81	1.00	0.76	0.84	0.67	0.67	0.63	0.58	0.73	0.35
rfc3	0.66	0.05	0.67	1.00	0.86	0.42	0.42	0.65	1.00	0.72	0.58	0.34
rfc5	0.69	0.03	0.73	1.00	0.85	0.56	0.52	0.67	0.89	0.63	0.68	0.35
rfc7	0.72	0.03	0.79	1.00	0.88	0.66	0.63	0.67	0.78	0.64	0.68	0.36
rfc10	0.75	0.04	0.89	1.00	0.93	0.84	0.81	0.68	0.87	0.65	0.70	0.39
adacR	0.55	0.06	0.56	0.97	0.82	0.22	0.22	0.42	0.65	0.80	0.16	0.01
adac	0.61	0.07	0.62	1.00	0.59	0.58	0.34	0.65	1.00	0.58	0.68	0.32
bagc	0.75	0.02	0.98	1.00	0.99	0.97	0.97	0.61	0.65	0.59	0.63	0.25

Table 4: Mean and variance of the accuracy in 10-fold cross-validation for the shuffled training data, and the accuracy for each class, and Cohen's kappa, for the training and test data set.

cross validation. This ensures that the obtained accuracy from the model will be representative of the performance of the model with arbitrary data. For our classification models, we use ones that are readily available in Scikit-learn.

First of all, we use CART decision trees with maximum depths of 3, 5, 7 and 10. The Gini impurity is used to measure the quality of a split. We also use random forest classifiers (RFC), which are bootstrap aggregated decision trees, with the same maximum depths as the CART decision trees and the same Gini impurity criterion. The number of trees in a forest is set to 100. Next, we use AdaBoost classifiers, using both the SAMME.R real boosting algorithm (AdaCR), and the SAMME discrete boosting algorithm (AdaC) [Hastie *et al.*, 2009]. A decision tree classifier is set as the base estimator, and we use a maximum of 50 estimators with a learning rate of 1.0. Finally, a bagging classifier (BagC) is used, which is an ensemble meta-estimator. Again, we use a decision tree classifier as the base estimator, and we use a maximum of 50 estimators.

Table 4 shows the results of the different classification models on the data based on the selected features. The second and third columns show the mean and 95% confidence interval of the accuracy in a 10-fold cross validation on the training data. The next four columns show the overall accuracy and accuracy for each class, for the training set, while the last four columns show the same for the test set. The results show that simple CART classifiers are competitive with the ensemble classifiers. Among the CART models, CART with depth 10 (CART10) shows the best performance as it has the highest accuracy in 10-fold cross-validation, on training, and on testing data sets. Note that CART10 has a reasonably high accuracy across the different classes, whereas the smaller CART models tend to perform poorly for a particular class. The random forest classifiers generally outperform CART10 but the differences are not that large. We use CART10 as input for the auction design optimization model since: (i) the performance on testing data is similar to the best random forest classifier; and more importantly, (ii) we can leverage the translation from the learned classification tree to a set of linear constraints as described in Section 4.

5.2 New auction design

We randomly take five sales from the test set and construct the optimization model from the learned classification tree to determine the starting price and lot number of the lots. Table 5 reports various accuracy for these sales, and the differences between old and new design parameters. The third column shows the fraction of lots that is being classified as high according to the optimization model by tweaking the starting price and lot number. We observe that the model is able to change the starting price and lot number in such a way that the learned classification tree classifies most lots, and in three sales even all lots, as high. The model assigns the lot number very differently from the expert, with the ranking being completely different in all cases. There seems to be no easily distinguishable pattern on which lot is assigned a lower number to be more visible to bidders. This is in contrast to the expert who assigns lots with higher estimated values lower lot numbers, so that these items appear higher on the list that is shown to bidders.

Upon closer inspection, we notice that overall the model assigns a higher starting price to lots compared to the expert. Many lots are assigned a slightly different SPEV. On the other hand, we see that the relatively few times that the model assigns a lower starting price to a lot. The lot in question is relatively more expensive. Experts usually deem these lots to be potentially more popular and will set the starting price high, whereas the model does not take into account these properties as strictly. These differences to experts' design strategies are interesting. As the proposed data driven model demonstrates higher expected outcomes of auctions with new designs, it may indicate that the auction experts' belief on how starting prices and lot numbers influence auctions is biased. Having said that, the learned classification model used in DDAD might also be biased due to high variety of items in different sales. We leave it as future work to investigate how to incorporate imperfect prediction models into decision models.

6 Conclusion

We propose an auction parameter design framework that integrates a classification tree that predicts the expected performance of auctions into an auction design optimization model. Currently, the starting prices and lot numbers are defined by

Sale (lots)	Acc	DDAD High	Higher SPEV	Avg Diff SPEV (+/-)
1 (47)	0.43	0.96	0.62	+0.20/-0.22
2 (36)	0.53	0.81	0.72	+0.17/-0.21
3 (6)	0.50	1.00	0.33	+0.15/-0.13
4 (71)	0.62	1.00	0.85	+0.25/+0.10
5 (56)	0.41	1.00	0.68	+0.13/-0.11

Table 5: Design results for the new sales using DDAD.

experts. With the implementation of our proposed approach, the experts would only have to determine the expected value but leave the design of starting prices and lot numbers to the model. We have shown that the proposed approach is effective as it improves upon the design from the auction experts.

Auctioneers are in particular interested in predicting low-performing items well. In the future, we will take into account different costs for different classes, using cost-sensitive learning [Elkan, 2001]. We will extend the work of [Verwer and Zhang, 2017; Verwer and Zhang, 2019] to learn classification models as a multi-objective optimization problem to incorporate different costs and learning objectives into learning algorithms.

References

[Ariely and Simonson, 2003] Dan Ariely and Itamar Simonson. Buying, bidding, playing, or competing? value assessment and decision dynamics in online auctions. *Journal of Consumer psychology*, 13(1-2):113–123, 2003.

[Cheema *et al.*, 2012] Amar Cheema, Dipankar Chakravarti, and Atanu R Sinha. Bidding behavior in descending and ascending auctions. *Marketing Science*, 31(5):779–800, 2012.

[Elkan, 2001] Charles Elkan. The foundations of cost-sensitive learning. In *International joint conference on artificial intelligence*, volume 17, pages 973–978. Lawrence Erlbaum Associates Ltd, 2001.

[Gabel and Riedmiller, 2008] Thomas Gabel and Martin Riedmiller. Adaptive reactive job-shop scheduling with learning agents. *International Journal of Information Technology and Intelligent Computing*, 2(4):1–30, 2008.

[Gurobi Optimization, 2018] LLC Gurobi Optimization. Gurobi optimizer reference manual, 2018.

[Hastie *et al.*, 2009] Trevor Hastie, Saharon Rosset, Ji Zhu, and Hui Zou. Multi-class adaboost. *Statistics and its Interface*, 2(3):349–360, 2009.

[Heyman *et al.*, 2004] James E Heyman, Yesim Orhun, and Dan Ariely. Auction fever: The effect of opponents and quasi-endowment on product valuations. *Journal of interactive Marketing*, 18(4):7–21, 2004.

[Huyet, 2006] A.L. Huyet. Optimization and analysis aid via data-mining for simulated production systems. *European Journal of Operational Research*, 173(3):827–838, September 2006.

[IBM, 2018] IBM. Ibm ilog cplex optimizer, 2018.

[Ku *et al.*, 2005] Gillian Ku, Deepak Malhotra, and J Keith Murnighan. Towards a competitive arousal model of decision-making: A study of auction fever in live and internet auctions. *Organizational Behavior and Human decision processes*, 96(2):89–103, 2005.

[Ku *et al.*, 2006] Gillian Ku, Adam D Galinsky, and J Keith Murnighan. Starting low but ending high: A reversal of the anchoring effect in auctions. *Journal of Personality and social Psychology*, 90(6):975, 2006.

[Lombardi and Milano, 2018] Michele Lombardi and Michela Milano. Boosting combinatorial problem modeling with machine learning. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18)*, pages 5472–5478, 2018.

[Lombardi *et al.*, 2017] Michele Lombardi, Michela Milano, and Andrea Bartolini. Empirical decision model learning. *Artificial Intelligence*, 244:343 – 367, 2017. Combining Constraint Solving with Mining and Learning.

[Milgrom and Milgrom, 2004] Paul Milgrom and Paul Robert Milgrom. *Putting auction theory to work*. Cambridge University Press, 2004.

[Myerson, 1981] Roger B Myerson. Optimal auction design. *Mathematics of operations research*, 6(1):58–73, 1981.

[Tang, 2017] Pingzhong Tang. Reinforcement mechanism design. In *IJCAI*, 2017.

[Van Heijst *et al.*, 2008] Dennis Van Heijst, Rob Potharst, and Michiel van Wezel. A support system for predicting ebay end prices. *Decision Support Systems*, 44(4):970–982, 2008.

[Verwer and Zhang, 2012] Sicco Verwer and Yingqian Zhang. Revenue prediction in budget-constrained sequential auctions with complementarities. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 3*, pages 1399–1400. International Foundation for Autonomous Agents and Multiagent Systems, 2012.

[Verwer and Zhang, 2017] Sicco Verwer and Yingqian Zhang. Learning decision trees with flexible constraints and objectives using integer optimization. In *International Conference on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pages 94–103. Springer, 2017.

[Verwer and Zhang, 2019] Sicco Verwer and Yingqian Zhang. Learning optimal classification trees using a binary linear program formulation. In *33rd AAAI Conference on Artificial Intelligence*, 2019.

[Verwer *et al.*, 2017] Sicco Verwer, Yingqian Zhang, and Qing Chuan Ye. Auction optimization using regression trees and linear models as integer programs. *Artificial Intelligence*, 244:368–395, 2017.

[Zhang *et al.*, 2010] Shu Zhang, Wolfgang Jank, and Galit Shmueli. Real-time forecasting of online auctions via functional k-nearest neighbors. *International Journal of Forecasting*, 26(4):666–683, 2010.