2023 IEEE Symposium Series on Computational Intelligence (SSCI)
Mexico City, Mexico. December 5-8, 2023

Interpretation of Neural Network Players for a Generalized Divide the Dollar Game Using SHAP Values

Garrison W. Greenwood
Dept. of Elect. & Computer Engr.
Portland State University
Portland, OR 97207–0751 USA
Email: greenwd@pdx.edu

Hussin Abbass
School of Engr. and Info. Tech.
University of New South Wales
Canberra, ACT2600, Australia
Email: h.abbass@adfa.edu.au

Aya Hussein
School of Engr. and Info. Tech.
University of New South Wales
Canberra, ACT2600, Australia
Email: a.hussein@adfa.edu.au

***Abstract*—Machine learning models can make accurate predictions but trust in the models depends on being able to understand why those predictions were made. Unfortunately, machine learning models are black boxes making interpretation difficult. Previously we used an evolutionary algorithm to evolve triplets of neural network players for instances of the Generalized Divide-the-Dollar, which is an economic bargaining game. The players produced fair bids with high bid totals, which is a desirable outcome, but no attempt was made to understand why the players performed so well. In this paper, we interpret the behavior of those neural networks using SHapley Additive exPlanations (or SHAP). Surprisingly, the neural network players exhibited both altruistic and exploitative behavior. Both a global and a local interpretation analysis is presented. The experiments conducted in this work demonstrate a simple method for understanding players' strategies in multi-player games.**

***Index Terms*—explainable AI, machine learning, model interpretation**

I. Introduction

Machine learning (ML) algorithms often operate as black-box function-approximators. They receive inputs and render output predictions, while often there remain questions about how their function approximation abilities work. Explainable AI (abbreviated XAI) is a set of methods that attempt to get some answers.

The vast majority of XAI studies have focused on explaining the behaviours of data-driven models, e.g. [1]–[3]. Despite the increasing interest in goal-driven agents, studies addressing the explainability issues are still sparse in this area. A recent review showed that there is a scarcity of XAI studies in multi-agent systems [4]. One of the aims of this paper is to contribute a technique for using XAI to interpret agent behaviors in a multi-agent system.

Greenwood and Ashlock [5] used an evolutionary algorithm to evolve neural network (NN) players for a 3-player game. In this paper, we use XAI methods to advance our understanding of the predictions made by these NN players.

Bargaining situations arise whenever two or more people must agree for mutual benefit over some economic transaction. Nash [6] created a nonzero-sum two-person game to study bargaining. *Divide the dollar* is a simplified version of Nash's game. Each round two players simultaneously bid on how much of $1 they are willing to accept. The players get a payoff equal to their bid if the bid total does not exceed $1; otherwise the players get a zero payoff. In a *generalized divide the dollar* (GDD) game, the $1 is split by $n > 2$ players[1].

Greenwood and Ashlock [5] had used a Covariance Matrix Adaptation Evolution Strategy (CMA-ES) algorithm to evolve NN players for a 3-player GDD game [5]. The input features were the bids of a player's opponents in the previous two rounds. The NN regression output was a player's bid. Three NNs were evolved that were effective and yet exhibited "fair" bidding—i.e., the bid total was at or near $1, but the bids were roughly equal so no player was exploited. Although the NNs performed well, they remain to be black boxes, falling short of explanations on how they bid or how they produced a particular prediction.

In the ML field *interpretability* has received considerable attention although precise definitions are still lacking. In a broad sense it involves extracting relevant knowledge from a ML model either contained in the data used for training or learned by the model. In this work we focused on *post hoc interpretability* [7] where a practitioner analyzes a previously trained model to gain insight into the learned relationships. There are two categories of post hoc interpretability:

- **Global interpretability:** which input features were most important, over an entire training/test set, in determining the model's outputs.
- **Local interpretability:** given a single input or a subspace of input features, which input features were most important in producing the observed output.

Some ML models, such as linear regression models or tree-based models (e.g., random forest), are interpretable while

[1]The bids become very small in a GDD game if a dollar is split by too many players. Therefore, with $n > 3$ players the amount split is increased to $n/2. Otherwise the rules remain unchanged.

more complex, black box models such as NNs are not. We need a *model-agnostic* interpretation method that does not require any knowledge about the underlying model. One approach is to construct an *explainer model*, such as a decision tree, that shadows and approximates the complex model's predictions (at least for a single input data point x) [8]. The black box model may not be interpretable, but the explainer model is. If the explainer model's prediction is a good approximation, then interpreting it approximates a good interpretation of the black box model it shadows.

The basic advantage in shadowing a model with another is to gain the benefits of both. A neural network are non-linear approximators, offering smooth approximation in areas of non-linear behaviours, which results in more accurate predictions. The shadower model, such as a decision tree, may need to grow infinitely large to reach the same level of smooth approximation. Instead, we can place parsimony pressures on the shadower, sacrificing some accuracy to gain on comprehensibility and compactness of the model.

SHAP values can then be extracted from the explainer model to indicate which features play prominent roles in a model's decision making. Efficient methods exist to extract SHAP values from an explainer model. These SHAP values, which were derived from the game theoretical optimal Shapley values, will produce the interpretation.

The ML model in our work are NNs representing players for a 3-player GDD game. XGBoost [9], a popular ML algorithm, was used for an explainer/shadower model. Both global and local interpretations of the model were conducted.

II. Background

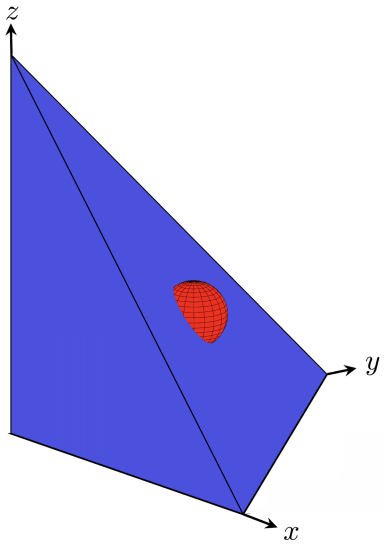


Fig. 1: The coordinated subspace for a 3-player GDD game. The subsidy region is the hemisphere protruding from the 2-simplex face.

A. GDD game

Divide-the-dollar was originally created as a 2-player game. Each round, both players would simultaneously submit a bid stating what fraction of a dollar they were willing to accept. If the bid total was a dollar or less the players would get their bid as a payoff. However, if the bid total was over a dollar both players got nothing.

GDD extends the basic divide the dollar game in two ways. First, it is now an N-player game with $N > 2$. Second, a small subsidy is available. This subsidy allows players to still get a non-zero payoff even if the bid total goes over one dollar. However, the subsidy is available if and only if the bids are "fair" (i.e. near uniform). That is, no player receives a payoff substantially higher than another player. For example, in a 3-player game with bids of $\{0.61, 0.20, 0.22\}$ the bid total is \$1.03 but one player would receive a much higher payoff. Suppose the bids were $\{0.33, 0.35, 0.35\}$. Again the bid total is \$1.03, but the payoffs are roughly the same. The subsidy would only be available in the second case.

Bid totals are called *coordinated* if the players receive a non-zero payoff and are called *uncoordinated* otherwise. For an N-player game, one can think of each bid as a coordinate of a point in $\mathbb{R}^N$. Associated with each point is a number equal to the sum of its coordinates—i.e., the bid total. Fig. 1 shows a subspace in $\mathbb{R}^3$. The slanted isosceles triangle is a 2-simplex where all bids total to exactly one dollar. The coordinated bid subspace is the 2-simplex and all points in the pyramid beneath it.

A hemisphere is shown protruding from the 2-simplex. This hemisphere and its interior represent the subsidy region. Notice, it is centered on the 2-simplex at the strictly-uniform bids [1/3 1/3 1/3]; thus, only fair bid sets are able to use a subsidy. For $N > 3$, the subsidy region is a hypersphere centered at the point $\left[\frac{1}{N}\ \frac{1}{N}\ \ldots\ \frac{1}{N}\right]$.

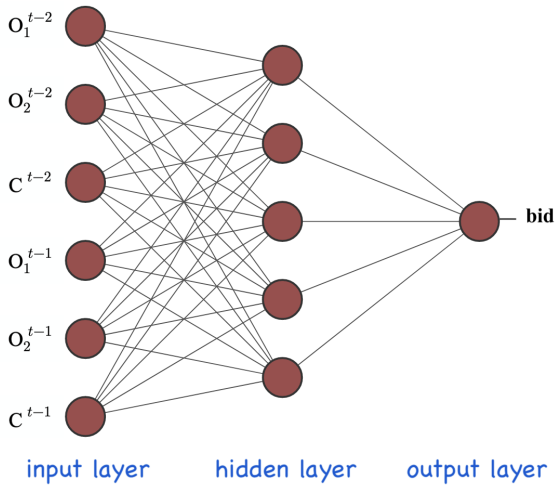


Fig. 2: The NN architecture for GDD players. See text for notation description.

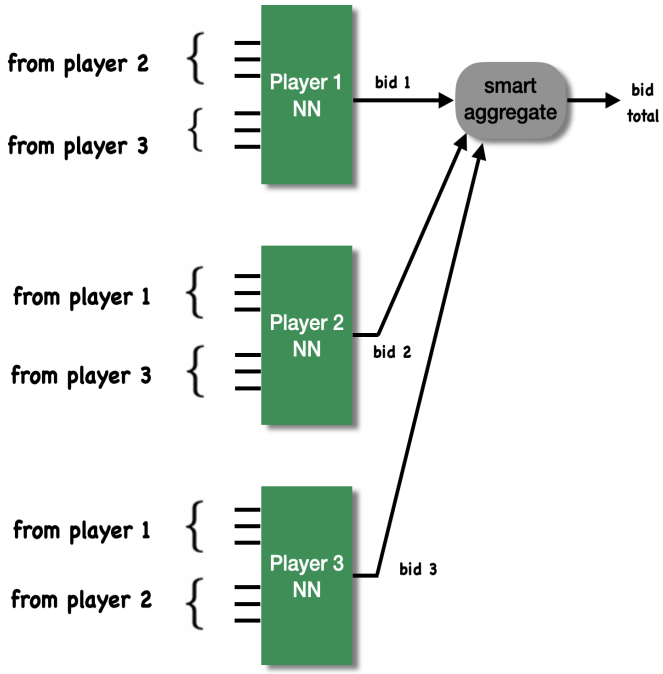


Fig. 3: The 3-player GDD framework.

B. The NN player architecture

Fig. 2 shows the architecture of a NN player. At every step t each NN receives 6 input features. Three features are the bids of its two opponents O_1 and O_2 at time step $t-1$ plus a C two-valued input indicating whether the bid total at that time step was coordinated or non-coordinated; with C=0.5 if bids are coordinated or 0.0 if non-coordinated bids. Another 3 similar set of features are provided for time step $t-2$.

Fig. 3 shows the configuration of a 3-player GDD game. Players are represented by a NN with prior bids made by its opponents in previous rounds as feature inputs. Let b_i be the current bid of the i-th NN player in an N-player game. Each b_i is a coordinate of a point in $\mathbb{R}^N$. The smart aggregator block outputs $\sum_i b_i$ if this point is on or beneath the simplex surface or inside the subsidy hypersphere. Otherwise it outputs 0.

For convenience, a special notation was used to label the feature inputs to the NNs. NN2 and NN3 are the 2 opponents of NN1. Consider the following feature vector input to NN1:

$$\{\mathrm{O}_1^{t-2}\ \mathrm{O}_2^{t-2}\ \mathrm{C}^{t-2}\ \mathrm{O}_1^{t-1}\ \mathrm{O}_2^{t-1}\ \mathrm{C}^{t-1}\} = \{\mathrm{O22\ O32\ C2\ O21\ O31\ C1}\}$$

O22 stands for the NN2 bid at time $t-2$. It is the O_1^{t-2} input in Fig. 1. O32 is the NN3 bid at time $t-2$; it is input O_2^{t-2}. C2 is the C^{t-2} input. Table I lists the input features for each NN.

C. SHAP

Consider a game where at the end of each round a payoff is distributed among the players. How much of the payoff should each player receive? A fair distribution would be based on a player's contribution, but often there are coalitions of players which makes individual contributions not so obvious. Lloyd Shapley [10] answered the question optimally by introducing what are called *Shapley values*. These values specify each player's contribution to a game outcome over all possible player coalitions. In mathematical terms, they are the average marginal contribution over all possible player coalitions.

TABLE I: Feature inputs for NN players

Neural Network	Input Feature Set
NN1	O22, O32, C2 O21, O31, C1
NN2	O12, O32, C2 O11, O31, C1
NN3	O12, O22, C2 O11, O21, C1

Unfortunately computing exact Shapley values using (II-C) is impractical for more than a handful of features because the complexity rises exponentially with the number of features. Lundberg and Lee [11] introduced a method called SHapley Additive exPlanations (or SHAP) to interpret ML black box models through Shapley values. Games now become ML models, players now become input features and characteristic functions now become model predictions.

SHAP belongs to the family of *additive feature attribution models*. These models explain more complex models (such as NNs) using a linear combination of binary variables.

$$g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z_i' \tag{1}$$

where $z' \in \{0,1\}^M$ and M is the number of features. ϕ_i is the effect feature i has on the ML model's prediction.

More specifically, let $x \in R^M$ be an input feature vector to a ML model. Often a simplified feature vector $x' \in \{0,1\}^M$ is constructed for x that has x'_j equal to 0 if the j-th feature is missing and 1 if present. Thus, coalitions (subsets of features from x) are represented by M-bit binary strings. A sample coalition vector x for input to the ML model is then created via a mapping function $h_x(x')$. For example, let $x = [x_1 x_2 x_3]$ and $x' = [1\,1\,1]$. Then $h_x(x') = [x_1\, x_2\, x_3]$ whereas $x' = [1\,1\,0]$ would produce $h_x(x') = [x_1\ x_2\ -]$ which is missing the third feature.

Let $z' \subseteq x'$. Then z' represents a coalition of features in x'. Let $f_x(z')$ be the output of the ML model for a feature coalition that includes feature i and $f_x(z'/i)$ the model output for a coalition that excludes feature i. Then the SHAP value ϕ_i for feature i is

$$\phi_i = \sum_{z' \subseteq x'} \frac{|z'|!\,(M - |z'| - 1)!}{M!} \left[f_x\left(z'\right) - f_x\left(z'/i\right)\right] \tag{2}$$

where $|z'|$ is the number of 1's in the vector z' and M is the number of features. Equation (2) sums over all possible coalitions of feature vector x (x').

A sample coalition vector may have one or more features missing. A ML model, however, cannot make predictions with

missing features. There are several ways of handling missing features. One simple method is to compute the expected value for each feature in the database. These expected values can be used as substitutes for missing features. Alternatively a random observation can be chosen from the database and its features used as substitutes.

SHAP computations are available in an open-source library that efficiently computes Shapley values in ML domains. It contains algorithms to find exact Shapley values for tree ensemble models (TreeSHAP), and approximate Shapley values for deep learning models (DeepSHAP). Kernel SHAP is model-agnostic because it can approximate Shapley values for any type of model. SHAP routines are available in several programming languages including python, R and MATLAB. Further details can be found in [12], [13].

III. Methods

The goal is to explain why NN players make particular bids. Fig. 4 shows the SHAP framework used for model explanation. The ML model is first fit to a database. Then observations from the database and the model are fed to SHAP, which computes ϕ_i for each feature i. Equation (**??**) provides the explanation.

Unfortunately we could not just substitute a player's NN for the ML model due to the absence of a database (sample). NNs are normally trained with a database, but in this case, they were designed through a cooperative, co-evolutionary process and evolved through interactions among individuals in the population. A CMA-ES algorithm evolved the NN synaptic weights and fitness was determined via a series of 15 round tournaments against other NN players. Highly fit NNs offer fair—i.e., roughly uniform—bids with bid totals near the surplus amount. A set of three, highly fit NN players were successfully evolved (see [5] for details) and SHAP is used to interpret their behavior.

A process similar to creating global surrogate models was used to build the needed database for SHAP analysis. An observation was constructed by applying random inputs to a NN player and recording the prediction (a bid). The baseline bid for the three players was [1/3 1/3 1/3] because this represents a fair bid set. Random noise was then added to the baseline to get feature vectors. For example, referring to Fig. 2,

$$O_1^{t-1} = 1/3 + \mathcal{N}(0, \sigma) \tag{3}$$

where $\mathcal{N}(0, \sigma)$ is a normally distributed random variable with zero mean and standard deviation σ. In this work $\sigma = 0.025$ is used. Other features are computed similarly. The C_1 and C_2 values are set since all NN outputs at $t-1$ and $t-2$ are known. This process was repeated to build a database with 2000 observations.

There are three NNs which means twelve O_i^{t-j} values are needed per observation. However, that does not mean (3) is applied twelve times. Consider NN1. O_1^{t-1} and O_1^{t-2} are the previous bids of NN3. But those feature values are also inputs to NN2. Random feature reuse reduces the application of (3) to six times per observation.

Some metrics, such as RMSE, measure model prediction accuracy. Conversely, R^2 measures database fit; the closer to 1.0, the better the fit. In the ML community R^2 is frequently used to compare different regression models on the same database. It is calculated as follows:

$$R^2 = 1 \; - \; \frac{\sum \left(y_{\text{pred}} - y_{\text{mean}}\right)^2}{\sum \left(y_{\text{actual}} - y_{\text{mean}}\right)^2} \tag{4}$$

Notice it is possible to have a negative R^2 value if the model is very poorly fit to the data.

The ML model shown in Fig. 4 is XGBoost. Effectively it is a global surrogate model. The XGBoost model had an $R^2 = 0.996$ value indicating it approximates the NN model quite well. Thus, any assumptions made about the XGBoost model holds for the NN model. Put another way, SHAP values can interpret ML models. It is impossible to extract SHAP values for a NN model, but SHAP values can be extracted for an XGBoost model. If the two models behave similarly, then interpreting the XGBoost model will also interpret the NN model.

IV. Results

In this section both local and global interpretations of the NNs players are presented. XGBoost, a gradient boosting algorithm, is used as the shadower/explainer of the NNs players that needs to be interpreted. TreeSHAP can efficiently extract the SHAP values from boosted tree models such as XGBoost. One thousand trees were involved with a tree depth of 5 and a learning rate of $\eta = 0.02$. The maximum available subsidy was \$0.05. Three feature vectors are chosen for local interpretation: one for a coordinated bid with a bid total less than or equal to \$1, a subsidized bid, and an uncoordinated bid. Decision plots are used to show the results.

In decision plots the input features are listed from top to bottom in order of importance—i.e., the top feature had the greatest impact on the model's prediction. A vertical line shows the baseline value. This is the output of a model with no input features; it is the average model prediction in the training set. The raw feature value is shown in parenthesis. SHAP is an additive attribution method so adding a feature increases or decreases the prediction relative to the baseline value. A segmented line shows how each SHAP value impacts the prediction. The final prediction is where this segmented line intersects the number line at the top of the plot. Blue lines indicate a final prediction lower than the baseline and red lines higher than the baseline.

Decision plots add SHAP values from the smallest to the largest value. In each case initially all three NNs behave similarly but differences appear as features are added. Coordinated bids tend to be smaller bids although, as shown in Fig. 5, NN3 exploited the smaller bids of the two opponents. Fig. 6 shows an uncoordinated bid total. All three bids were high which indicates all three players were equally responsible for the high bid total. In this case all players received no payoff. The subsidized case, shown in Fig. 7 is quite interesting. All three

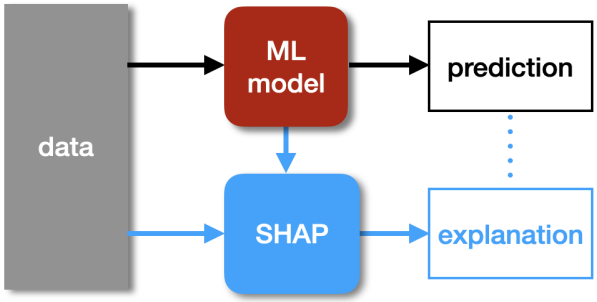


Fig. 4: The SHAP framework.

players had high bids, which is necessary to take advantage of a subsidy, but two players substantially reduced their bids when the third player consistently submitted high bids. This suggests altruistic behavior; getting some payoff is better than getting no payoff.

It is interesting to see the interaction between players. Players try to get coordinated bid totals because otherwise there is no payoff. Consequently, opponent bids above the baseline in previous rounds tend to drive current bids lower. But low opponent bids in previous rounds can produce just the opposite effect in opponents. There are two ways of looking at this behavior. Players that reduce their bids because previously opponents bid high suggests altruism. Conversely, players that increase their bids because opponents previously bid low indicates exploitative behavior.

Referring to Fig. 5, all three NNs made predictions below the baseline value. NN1 considered opponent bids in the previous round as important, and, since those bids are high, its final prediction is low. For NN2 the large bid from NN3 in the previous round drove the prediction lower but the below baseline NN1 bid drove it higher. The NN3 output bid consistently moved lower except the low bid from NN1 in the previous round drove it to a higher final prediction.

Low opponent bids in previous rounds tends to increase the bid in the current round. Fig. 6 shows an uncoordinated bid decision plot. At this data point, all three NNs saw low opponent bids in previous rounds and consequently, significantly increased their bid. Such behavior increases the likelihood of uncoordinated bid totals.

Decision plots for a subsidized bid total are shown in Fig. 7. NN1 dramatically increased its bid because the input features indicated the other opponent bids were low in the previous two rounds and the bid totals were coordinated. NN2 and NN3 had similar behavior. Both would have had a high bid if it were not for the very high NN1 bid in the previous round. Indeed, O11 was the most influential feature for both NNs.

A global interpretation is depicted in Fig. 8. The beeswarm plot shows all three NNs see the opponent's bids in the previous round $(t-1)$ have the most influence on current predictions; a kind of a tit-for-tat strategy.

The global interpretation shown in Fig. 8 is interesting. All plots indicate bids from the previous round had the most influence on current bid predictions. Notice the red portion is to the left for all Oij bid inputs but to the right on all Cj inputs. High opponent bids lead to lower current bids which promotes altruistic behavior. Conversely, coordinated bids in the previous rounds tend to increase the current bid. This could explain the exploitative behavior.

The beeswarm plots also indicate feature importance globally. The top-to-down ordering indicates which features, on average, affected the NN bids the most. Opponent bids in round $t-2$ had the least affect on a NN player's bid and that was true for all three NNs.

V. Final Remarks

SHAP values provide an effective method for interpreting the NN bids. But this suggests another use that has not been investigated before.

Finally, an evolutionary algorithm was used to design three NNs modeling GDD game players that offered fair bids with high bid totals. The global interpretation shows opponent bids in round $t-2$ have the least affect on a player's bid. The evolutionary process used the full 18 feature input set. It would be interesting to see if this evolutionary algorithm can find three players, with the desirable behavior, while using a reduced feature input set that did not include the $t-2$ round opponent bids.

References

[1] F. Xu, H. Uszkorei, Y. Du, W. Fan, D. Zhao, and J. Zhu. Explainable ai: A brief survey on history, research areas, approaches and challenges. In *Natural Language Processing and Chinese Computing*. Springer International Publishing, 2019.

[2] P. Angelov, E. Soares, R. Jiang, N. Arnold, and P. Atkinson. Explainable artificial intelligence: an analytical review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 11(5):e1424, 2021.

[3] D. Minh, H. Wang, Y. Li, and T. Nguyen. Explainable artificial intelligence: a comprehensive review. *Artificial Intelligence Review*, 55(5):3503–3568, 2022.

[4] F. Sado, C. Loo, W. Liew, M. Kerzel, and S. Wermter. Explainable goal-driven agents and robots - a comprehensive review. *ACM Comput. Surv.*, Nov 2022.

[5] G. Greenwood and D. Ashlock. Evolving neural networks for a generalized divide the dollar game. In *2022 IEEE Cong. on Evol. Comput. (CEC)*, pages 1–8, 2022.

[6] J. Nash. The bargaining problem. *Econometrica*, 18(2):155–162, 1950.

[7] W. Murdoch, C. Singh, K. Kumbier, R. Abbasi-Asi, and B. Yua. Definitions, methods, and applications in interpretable machine learning. *PNAS*, 116(44):22071–22080, 2019.

[8] H. Abbass, M. Towsey, and G. Finn. C-Net: A method for generating non-deterministic and dynamic multivariate decision trees. *Knowledge and Information Systems*, 3:184–197, 2001.

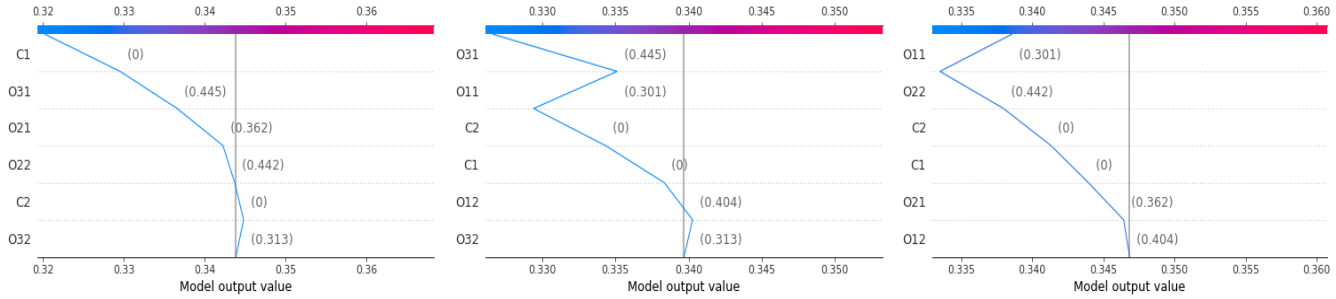
[9] https://xgboost.readthedocs.io/en/stable/.

[10] L. Shapley. *7. A Value for n-Person Games. Contributions to the Theory of Games II (1953)*, pages 69–79. Princeton University Press, 1997.

[11] S. Lundberg and S. Lee. A unified approach to interpreting model predictions. In *NIPS'17: Proc. 31st Int'l. Conf. Neural Infor. Proc. Sys.*, pages 4768–4777, 2017.

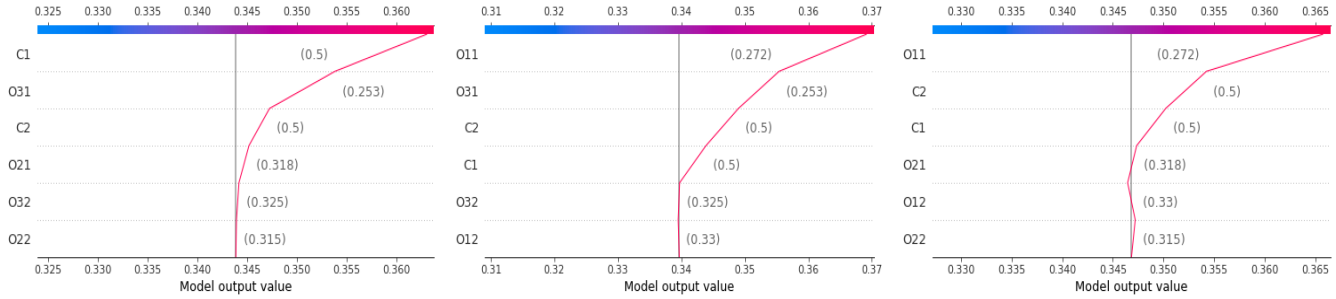
[12] S. Lundberg, G. Erion, H. Chen, A. DeGrave, J. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal, and S. Lee. From local explanations to global understanding with explainable AI for trees. *Nature Mach. Intell.*, 2(1):56–67, 2020.

[13] https://github.com/slundberg/shap.



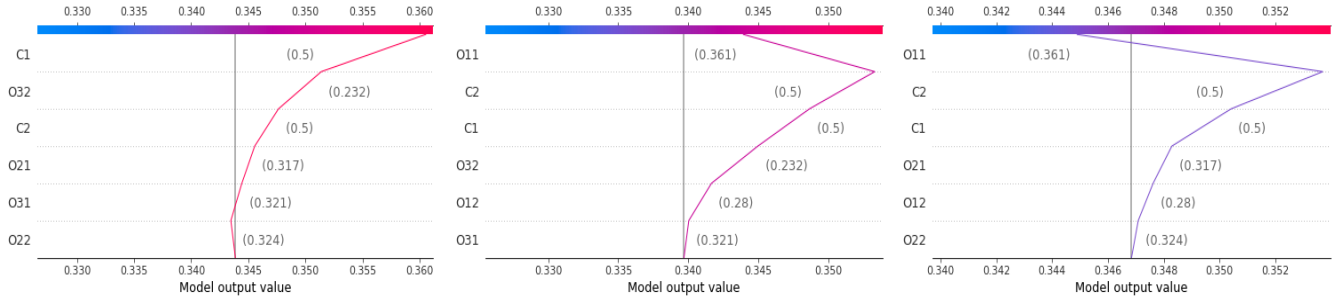
(a) NN1 (b) NN2 (c) NN3

Fig. 5: Decision plots for a coordinated bid total.



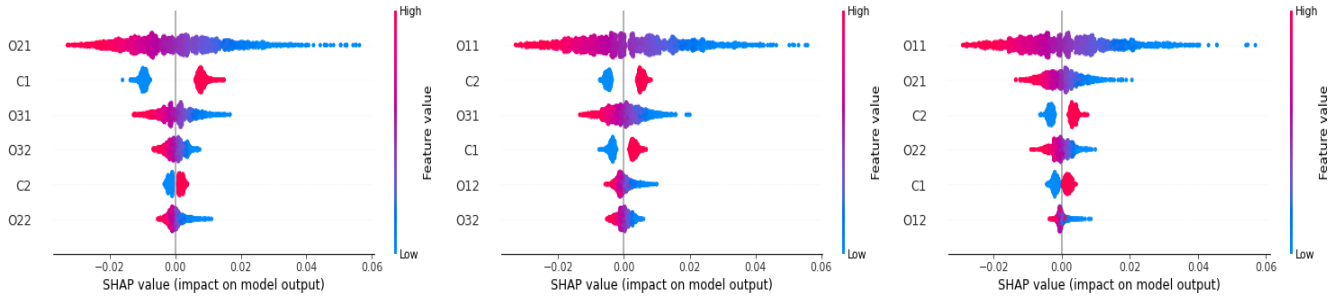
(a) NN1 (b) NN2 (c) NN3

Fig. 6: Decision plots for an uncoordinated bid total.



(a) NN1 (b) NN2 (c) NN3

Fig. 7: Decision plot for a subsidized bid total.



(a) NN1 (b) NN2 (c) NN3

Fig. 8: Beeswarm plots showing global interpretation of the three NNs.