Massachusetts Institute of Technology

Operations Research Center

15.095 Machine Learning Under A Modern Optimization Lens

Instructor: Prof. D. Bertsimas

Incorporating Fairness and Interpretability using Optimization: A Monotonic Constraint Approach

Angela Lin aglin@mit.edu
Asterios Tsiourvas atsiour@mit.edu

Cambridge, MA

Monday 21st August, 2023

1 Motivation

Fairness in Machine Learning (ML) models is critical for building safe and responsible AI systems. ML is increasingly being used in a wide range of decision-making scenarios that have serious implications for individuals and society, including financial lending, hiring, online advertising, pre-trial and immigration detention, child maltreatment screening, health care, social services, and education. While accuracy is one metric for evaluating the performance of a ML model, we must also consider the practical implications of deploying the model in a real-world situation. Building fair models involves understanding bias introduced by the data, and ensuring that the final model provides equitable predictions on the basis of characteristics.

Interpretability in ML models helps users understand why and how predictions are made, which is critical in determining whether decisions made from these predictions are fair or not. If we cannot explain why our model is making certain predictions, we lose users' confidence and trust, and it is also difficult for us to intervene and improve the model. Therefore, an important goal of ML models is to be able to explain to humans why a particular prediction was made - especially about unexpected events - among many others. Interpretability is a desired trait of ML models that if achieved can bridge the gap of trust between machines and humans.

In order to address these goals, we study Isotonic Regression. Isotonic (or monotonic) Regression is a general technique used in statistics where the fitted "line" is monotonic with respect to all or to a subset of the model features. When ethical predictions are needed, enforcing monotonic predictions can improve the fairness and the intepretability of ML models. For instance, it would be unfair for a ML model to select person A over person B if person B has better or equal metrics than A in all features. For example, let's take a look at Table 1.

Features	Student A	Student B
GPA	4.5	3.0
SAT score	$1,400$	$1,200$
National Awards	5	2
Prediction	Reject	Accept

Table 1: Example of an unfair ML model

In this example, the ML model suggests that Student A should be rejected, while Student B should be accepted. However, Student A outperforms Student B in all metrics. This sort of unfairness may be unacceptable in practice, or at the very least, it would be very hard to explain why the model made this choice. In such cases monotonicity can improve the interpretability and the fairness of ML models.

Another example is when monotonicity is the underlying truth or a natural property. In these cases, monotonic constraints can improve the accuracy and interpretability of predictions. For instance, this is the case when predicting the demand as a function of price. It is natural to assume that when price increases, the demand decreases. When monotonicity is a natural property in the underlying truth, enforcing monotonic predictions can help the model make more accurate and interpretable predictions.

2 General Framework

In this project we use the following general framework for Isotonic Regression. We are given a set of n observations $\{(x^1, y^1), ..., (x^n, y^n)\}$ where $x^i \in \mathbb{R}^p, y^i \in \mathbb{R}, \forall i \in [n]$. $\alpha \subset [1, ..., p]$ is the subset of features (components of x) that we wish to impose monotonic predictions upon. The general problem can be formulated as

$$\begin{array}{ll} \min_f & \sum_{i=1}^n \mathcal{L}(f(x^i), y^i) \\ \text{subject to} & f(x^j) \leq f(x^i), \ \forall x^j_\alpha \leq x^i_\alpha, \ (i,j) \in [n] \times [n] \end{array} \tag{1}$$

where $x_\alpha \leq x'_\alpha$ denotes inequality for **all** indices in α, i.e. $x_j \leq x'_j \ \forall j \in \alpha$ and $\mathcal{L}(\cdot, \cdot)$ is the loss function.

In addition to enforcing monotonic predictions on the training data, we expect monotonic predictions on test data as well. Indeed, as the number of samples, n, grows large, the more likely it is that our predictions on the test set will be monotonic in the features α as well. In our experimental results, even with few samples, our test set predictions were always monotonic with respect to the features α.

3 Models

3.1 Linear Regression

In the Linear Regression setting, we replace $f(x^i)$ with $\beta x^i + \beta_0 = \sum_{j=1}^{p} \beta_j x_j^i + \beta_0$, $\mathcal{L}(y^i, f(x^i))$ with $(y^i - f(x^i))^2$ and the model becomes

$$\begin{array}{ll} \min_{\beta,\beta_0} & \sum_{i=1}^{n} (\beta x^i + \beta_0 - y^i)^2 \\ \text{subject to} & \beta x^j + \beta_0 \leq \beta x^i + \beta_0, \ \forall x_\alpha^j \leq x_\alpha^i, \ (i,j) \in [n] \times [n] \end{array} \tag{2}$$

where $x_\alpha \leq x'_\alpha$ denotes inequality for all indices in α, i.e. $x_j \leq x'_j \quad \forall j \in \alpha$.

This is a quadratic optimization problem with linear constraints that can be solved by Gurobi. There are $\leq \binom{n}{2}$ constraints (order of $O(n^2)$) in this formulation. To improve scalability, we implemented a lazy constraint callback function so we only need to add the constraints that are violated each time.

3.2 Logistic Regression

In the Logistic Regression setting we replace $f(x^i)$ with $\frac{e^{\beta x^i + \beta_0}}{1 + e^{\beta x^i + \beta_0}}$, $\mathcal{L}(y^i, f(x^i))$ with $\log(1 + e^{-y^i f(x^i)})$, and the model becomes

$$\begin{array}{ll} \min_{\beta,\beta_0} & \sum_{i=1}^{n} \log(1 + e^{-y^i f(x^i)}) \\ \text{subject to} & \beta x^j + \beta_0 \leq \beta x^i + \beta_0, \ \forall x_\alpha^j \leq x_\alpha^i, \ (i,j) \in [n] \times [n] \end{array} \tag{3}$$

where $x_\alpha \leq x'_\alpha$ denotes inequality for all indices in α, i.e. $x_j \leq x'_j \quad \forall j \in \alpha$.

This is a convex optimization problem with linear constraints that can be solved by Gurobi. Regarding the number of constraints and the scalability of the method the same results hold as in the Linear Regression case.

3.3 Optimal Trees

In a regression tree setting, we define that $f(x^i) = f^i$, which is the prediction made by the Optimal Tree for point i. We solve a mixed integer linear optimization problem (based on formulation (10.2) on page 183 of [4]) in order to fit our isotonic regression tree. This formulates an optimal regression tree with parallel splits and constant predictions for each leaf. We choose parallel splits for their interpretability. The variables in the optimization problem are:

- f^i = the prediction for training point i
- β_t = the prediction for leaf node t
- z_{it} = indicator for whether training point i is assigned to leaf node t
- d_t = indicator for whether or not there's a split on branch node t
- a_{jt} = indicator for whether the j^{th} variable is used for the split at branch node t
- b_t = the split value for the split at branch node t
- l_t = indicator for whether or not leaf t has any points assigned to it
- L_i = the prediction error for point i
- C = the complexity of the tree

The user-defined parameters are:

- N_{min} = the minimum number of points assigned to each leaf
- λ = the complexity parameter
- max depth = the maximum depth of the tree

L is the set of all leaf nodes; B is the set of all branch nodes; $R(t)$ is the set of right ancestors of leaf node t; and $L(t)$ is the set of left ancestors of leaf node t.

The full formulation is:

$$\begin{array}{ll}
\min_f & \sum_{i=1}^{n} L_i + \lambda C \\
\text{subject to} & f^j \leq f^i,\ \forall x_\alpha^j \leq x_\alpha^i,\ (i,j) \in [n] \times [n] \\
& L_i \geq |f^i - y^i|,\ \forall i \in [n] \\
& f^i - \beta_t \geq -M(1 - z_{it}),\ \forall i \in [n], t \in L \\
& f^i - \beta_t \leq M(1 - z_{it}),\ \forall i \in [n], t \in L \\
& C = \sum_{t \in B} d_t \\
& a_m^T x^i \geq b_m - M(1 - z_{it}),\ \forall i \in [n], t \in L, m \in R(t) \\
& a_m^T x^i + \epsilon \leq b_m + M(1 - z_{it}),\ \forall i \in [n], t \in L, m \in L(t) \\
& \sum_{t \in L} z_{it} = 1, \forall i \in [n] \\
& z_{it} \leq l_t \forall i \in [n], t \in L \\
& \sum_{i=1}^{n} z_{it} \geq N_{min} l_t,\ \forall t \in L \\
& \sum_{j=1}^{p} a_{jt} = d_t,\ \forall t \in B \\
& d_t \leq d_{p(t)},\ \forall t \in B \setminus \{1\} \\
& z_{it}, l_t, a_{jt}, d_t \in \{0, 1\}
\end{array} \tag{4}$$

where the first constraint enforces monotonic predictions with respect to α. (Note: since the Optimal Trees MIO formulation takes quite some time for Gurobi to solve, we didn't have the ability to cross-validate our parameters over a wide range. For this same reason, we also used a small training data set and small set of features.)

3.4 Neural Networks

In the Neural Network setting we used the TensorFlow Lattice [1] approach by Google to create monotonic Neural Networks. TensorFlow Lattice is an open-source framework that allows users to implement interpretable lattice based models by injecting domain knowledge into the learning process. A lattice, in the TensorFlow setting, is defined as a look-up table that approximates input-output relationships in the data. It creates a grid over the input space and learns values for the output by learning values for the vertices of this grid. For each incoming test point, the output is a linear interpolation from the vertices values close to the input's mapping. The lattice structure allows to inject domain knowledge such as monotonicity, convexity/concavity, unimodality and pairwise trust. Regarding monotonicity we can impose that the output (which is the linear interpolation of the vertices of the lattice) should only increase/decrease with respect to a set of input features. One other way to impose mononicity on Neural Networks with ReLU activations is the technique proposed by Liu et al. [6] which was one of the main inspirations for this work. Since we did not implement this approach, we describe it in the Appendix.

4 Data

4.1 Classification Use Case: Law Data Set

Our first data set is a law school data set from University of California Irving (UCI) Machine Learning Repository. Each row in this data set corresponds to a law student, and the columns are the student's age, family income, part-time status, gender, undergraduate GPA, LSAT score, and whether or not the student passes the bar exam. We choose our set of features to impose monotonic predictions upon α = {undergraduate GPA, LSAT score} because it would be difficult to explain why we predict a student with lower GPA *and* LSAT score to pass the bar exam over another student with higher GPA AND LSAT score. An exploration of this data set can be found on the Appendix.

4.2 Regression Use Case: Coffee Data Set

Our second data set is from an external retail collaborator of our advisor, Prof. Georgia Perakis. Each row corresponds to the sales of a coffee product in a particular week, and the columns are the product number, product brand, price, the cumulative week, the week of the year, promotion percent (percent discounted), and the total volume sold. We subset the data to include only 1 brand and 1 product, and we predict the total volume sold (which we assume is the true demand) using the price, the promotion percent, the cumulative week, and the week of the year. We choose our set of features to impose monotonic predictions upon $\alpha = \{$-negative price, promotion percent $\}$ because we expect the ground truth to be that the demand for a product goes up as the price decreases. An exploration of this data set can be found on the Appendix.

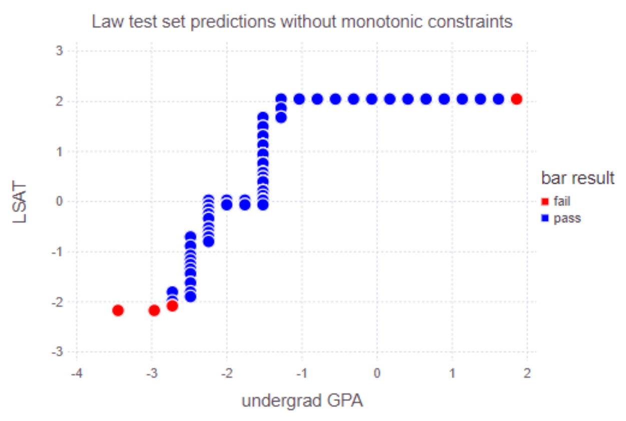
5 Computational Results

5.1 Classification

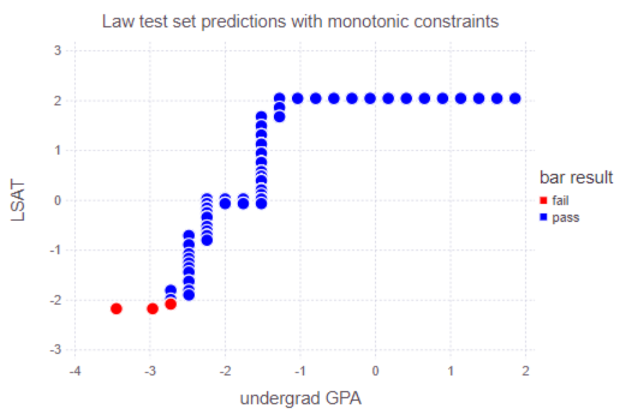
We re-scale all features to have zero mean and unit variance. We train all models with and without monotonic constraints on a training set of 596 samples and make predictions on a test set of 234 samples. We plot predicted bar exam result against $\alpha = \{$undergraduate GPA, LSAT score$\}$ on the test set.

5.1.1 Logistic Regression

Figure 1: Logistic Regression on Law Test Set



(a) Logistic Regression w/o Monotonic Constraints



(b) Logistic Regression with Monotonic Constraints

The model without monotonic constraints (plot (a)) predicts that the student with the highest LSAT score and GPA fails the bar exam while predicting that many students with lower LSAT and GPA pass. The predictions made by the model with monotonic constraints (plot (b)) do not violate monotonicity and are therefore more logical and explainable.

	w/o monotonic constraints	w/ monotonic constraints	percentage improvement
% Correctly Classified	92.9%	94.9%	+2.1%
AUC	0.83	0.85	+2.2%

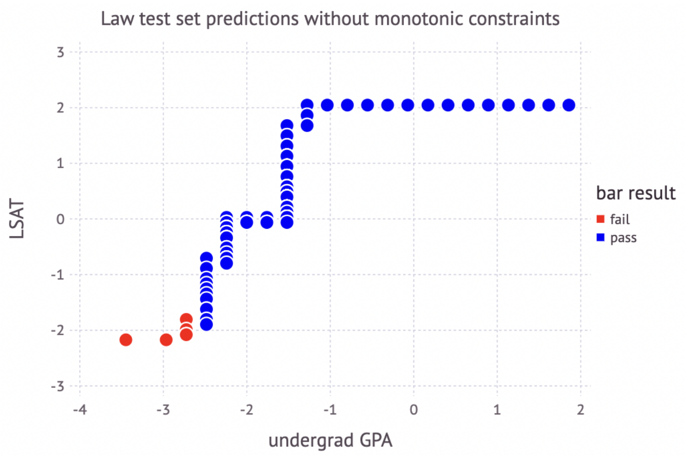
Table 2: Logistic Regression Metrics on Law Test Set

We see an improvement in both out-of-sample classification accuracy and out-of-sample AUC when using the model with monotonic constraints to predict.

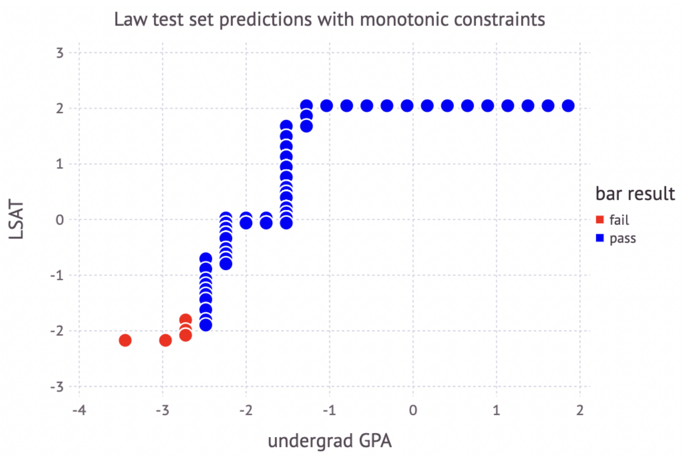
5.1.2 Optimal Trees

We trained Optimal Trees with max depth = 3, $\lambda = 0.01$, and $N_{min} = 5$.

Figure 2: Optimal Trees on Law Test Set



(a) Optimal Trees w/o Monotonic Constraints



(b) Optimal Trees with Monotonic Constraints

Optimal trees make monotonic predictions with respect to α on this data set whether monotonic predictions are explicitly enforced or not. In fact, the optimal tree trained without monotonic constraints is identical to the tree trained with monotonic constraints.

	w/o monotonic constraints	w/ monotonic constraints	percentage improvement
% Correctly Classified	94.9%	94.9%	0.0%
AUC	0.618	0.618	0.0%

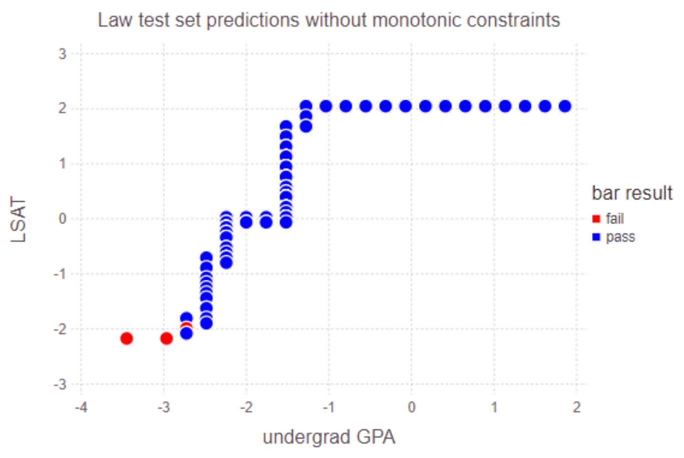
Table 3: Optimal Tree Metrics on Law Test Set

This result indicates that optimal trees may have a natural advantage on making monotonic predictions.

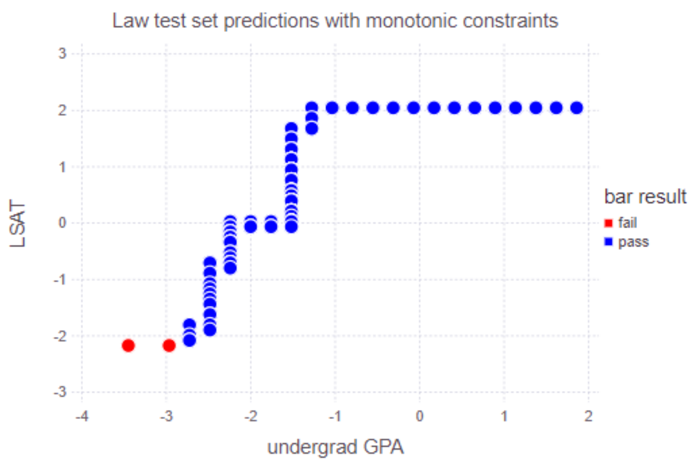
5.1.3 Neural Networks

We trained a 2−layer feed-forward neural networks with 100 neurons per layer.

Figure 3: Neural Network on Law Test Set



(a) Neural Network w/o Monotonic Constraints



(b) Neural Network with Monotonic Constraints

The model without monotonic constraints makes a non-monotonic prediction (near the bottom left of plot (a)). The predictions made by the model with monotonic constraints (plot (b)), do not violate monotonicity, and are therefore more logical and explainable.

	w/o monotonic constraints	w/ monotonic constraints	percentage improvement
% Correctly Classified	94.1%	94.9%	+0.4%
AUC	0.85	0.86	+0.8%

Table 4: Neural Network Metrics on Law Test Set

We see an improvement in both out-of-sample classification accuracy and out-of-sample AUC when using the model with monotonic constraints to predict.

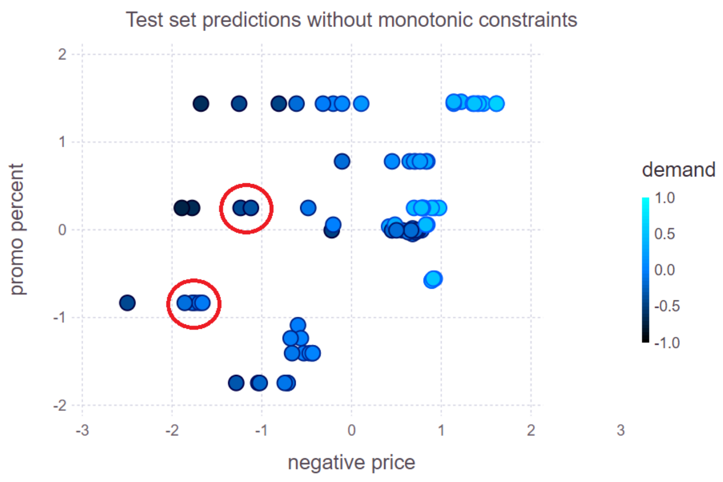
Conclusion: In the classification use case, monotonic constraints improve the out-of-sample accuracy and AUC of all three ML models (which means that they capture a part of the underlying truth) and also lead to more ethical/fair and interpretable models.

5.2 Regression

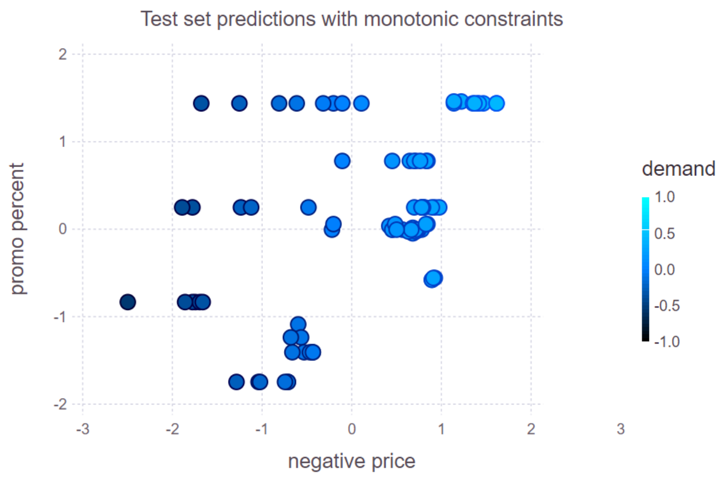
We re-scale all features and the target variable to have zero mean and unit variance. We train all models with and without monotonic constraints on a training set of 130 samples and make predictions on a test set of 86 samples. We plot predicted demand against α = {-negative price, promotion percent} on the test set.

5.2.1 Linear Regression

Figure 4: Linear Regression on Coffee Test Set



(a) Linear Regression w/o Monotonic Constraints



(b) Linear Regression with Monotonic Constraints

The model without monotonic constraints (plot (a)) makes non-monotonic predictions (some examples where a higher demand is predicted for a higher price and lower promotion percent are circled in red).
The predictions made by the model with monotonic constraints (plot (b)) do not violate monotonicity and are more interpretable.

	w/o monotonic constraints	w/ monotonic constraints	percentage improvement
R^2	−0.078	0.032	+141.3%
MSE	0.99	0.89	−10.3%

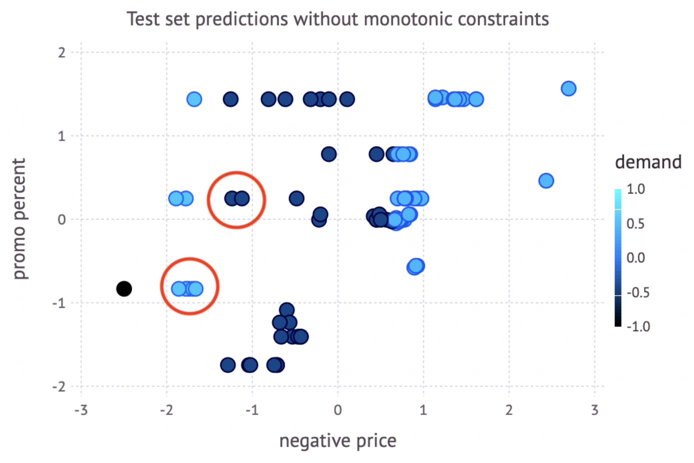
Table 5: Linear Regression Metrics on Coffee Test Set

We see an improvement in both out-of-sample R^2 and out-of-sample MSE when using the model with monotonic constraints to predict.

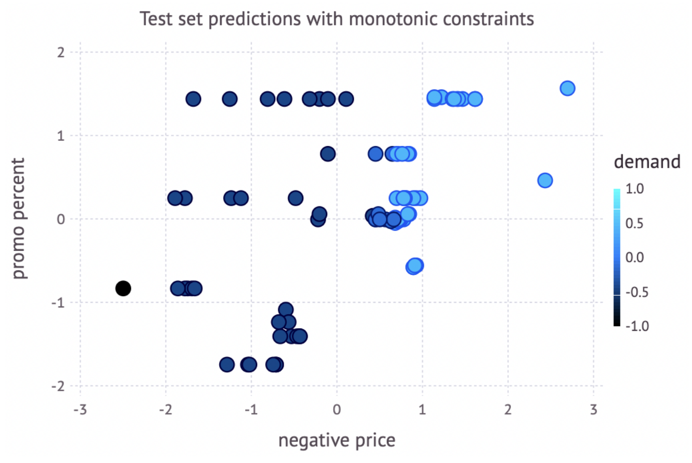
5.2.2 Optimal Trees

We trained Optimal Trees with max depth $= 3$, $\lambda = 0.01$, and $N_{min} = 5$.

Figure 5: Optimal Trees on Coffee Test Set



(a) Optimal Trees w/o Monotonic Constraints



(b) Optimal Trees with Monotonic Constraints

The model without monotonic constraints (plot (a)) makes non-monotonic predictions (some examples where a higher demand is predicted for a higher price and lower promotion percent are circled in red).
The predictions made by the model with monotonic constraints (plot (b)) do not violate monotonicity and are more interpretable.

	w/o monotonic constraints	w/ monotonic constraints	percentage improvement
R^2	-0.087	-0.008	$+90.7\%$
MSE	1.007	0.904	-10.2%

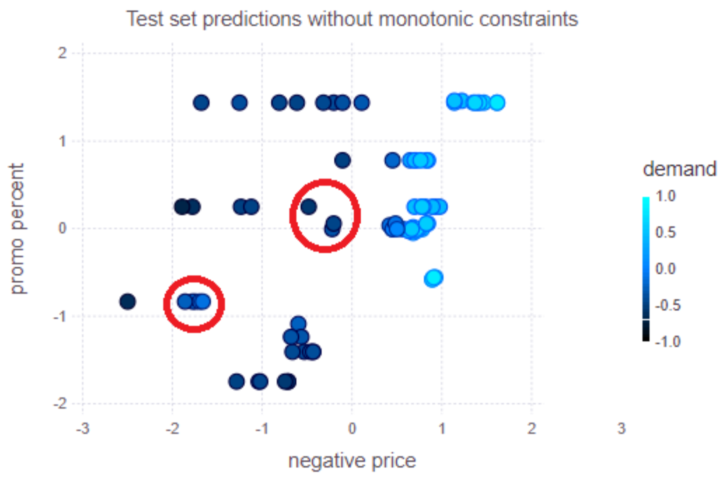
Table 6: Optimal Tree Metrics on Coffee Test Set

We see an improvement in both out-of-sample R^2 and out-of-sample MSE when using the model with monotonic constraints to predict.

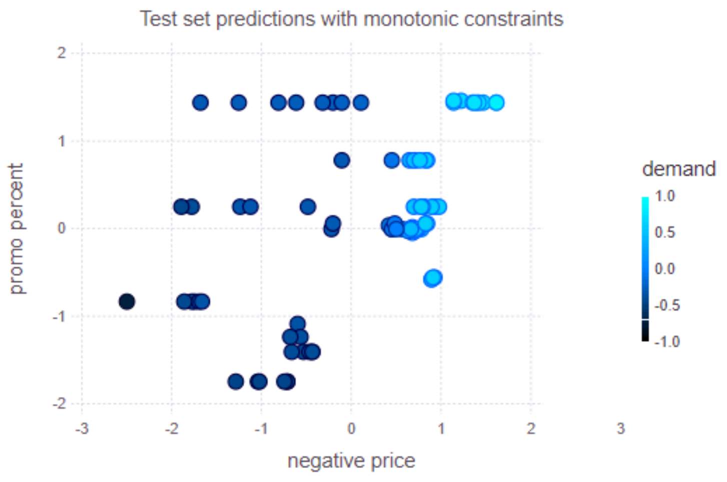
5.2.3 Neural Networks

We trained a 2−layer feed-forward neural networks with 100 neurons per layer.

Figure 6: Neural Networks on Coffee Test Set



(a) Neural Network w/o Monotonic Constraints



(b) Neural Network with Monotonic Constraints

The model without monotonic constraints (plot (a)) makes non-monotonic predictions (some examples where a higher demand is predicted for a higher price and lower promotion percent are circled in red).
The predictions made by the model with monotonic constraints (plot (b)) do not violate monotonicity and are more interpretable.

	w/o monotonic constraints	w/ monotonic constraints	percentage improvement
R^2	0.086	-0.045	+89.3%
MSE	1.00	0.96	-3.9%

Table 7: Neural Network Metrics on Coffee Test Set

We see an improvement in both out-of-sample R^2 and out-of-sample MSE when using the model with monotonic constraints to predict.

Conclusion: Again, in the regression case monotonic constraints improve the out-of-sample R^2 and MSE of all three ML models (which means that they capture the underlying truth about demand and price) and also produce more interpretable models that are easier to be explained.

6 Limitations and Future Work

The size of the data sets that we were able to train our models with was limited by the scalability of our Optimal Tree implementation. We reduced both the number of features and samples used so that our model would solve in a reasonable amount of time. We believe this is why our out-of-sample performance on the regression task was subpar. We could improve the scalability of this method by adding our monotonic constraints to Interpretable AI (IAI)'s implementation and using their local search procedure for training Optimal Trees. Moreover, our implementation doesn't have a specialized way of dealing with categorical variables, but the IAI software already has this implemented, so again, using their implementation would be beneficial. We could also add more constraints to our isotonic models to get more holistic regression models, as we saw in class. For example, we could enforce sparsity, robustness, non-multicollinearity, statistical significance, etc.

7 Conclusions

In this project, we worked towards the creation of fair and interpretable ML models using monotonic constraints. Using the optimization techniques learned in class, we incorporated monotonic constraints into some of the most widely used ML models, such as Linear/Logistic regression, Optimal Trees and feed-forward Neural Networks. Our experiments showed that in both classification and regression settings monotonic constraints can indeed improve the fairness and interpretability of these models, as enforcing monotonic predictions with respect to certain features led to more ethical and/or easily understandable models. Our experiments also showed that monotonic constraints can improve the predictive power of models by capturing the underlying truth of the data. Finally, monotonic constraints can be incorporated easily into well-known commercial solvers. Since the number of constraints required is polynomial with respect to to the number of training samples, our monotonic models have the potential to be scalable when implemented using commercial solvers.

8 Meeting Notes 2/15/22

Let $x, y \in \mathcal{X}$. We want for every $x, y \in \mathcal{X}$ such that $x_j \leq y_j,\ \forall j \in \alpha \subseteq [d]$, to have

$$\beta^T x + \beta_0 \leq \beta^T y + \beta_0 \implies \beta^T(x-y) \leq 0,\ \forall x, y \in \mathcal{X}, x_j \leq y_j,\ \forall j \in \alpha \subseteq [d] \tag{5}$$

By taking the maximum:

$$\max_{x,y \in \mathcal{X}, x_j \leq y_j,\ \forall j \in \alpha} \beta^T(x-y) \leq 0 \tag{6}$$

By taking the dual we can obtain certified monotonicity.

For any $x, y \in \mathcal{X}$ such that $x_j \leq y_j\ \forall j \in \alpha \subseteq [d]$, we want

$$\beta^T(x-y) \leq 0 \tag{7}$$

$$\beta_\alpha^T(x_\alpha - y_\alpha) \leq -\beta_{\alpha^c}^T(x_{\alpha^c} - y_{\alpha^c}) \tag{8}$$

This is the same as

$$\max_{x,y \in \mathcal{X} s.t. x_j \leq y_j,\ \forall j \in \alpha} \beta_\alpha^T(x_\alpha - y_\alpha) \leq \min_{x,y \in \mathcal{X} s.t. x_j \leq y_j,\ \forall j \in \alpha} -\beta_{\alpha^c}^T(x_{\alpha^c} - y_{\alpha^c}) \tag{9}$$

9 Acknowledgments

We'd like to thank our TAs for generously donating their time to give us constructive feedback as well as Prof. Bertsimas and Prof. Perakis for their guidance.

References

[1] Martin Abadi et al. TensorFlow Lattice: Large-scale machine learning on heterogeneous systems, 2021.

[2] R. E. Barlow and H. D. Brunk. The isotonic regression problem and its dual. *Journal of the American Statistical Association*, 67(337):140–147, 1972.

[3] D. Bertsimas and J. Dunn. *Machine Learning Under a Modern Optimization Lens*. Dynamic Ideas LLC, 2019.

[4] Dimitris Bertsimas and Jack Dunn. Optimal classification trees. *Machine Learning*, 106(7):1039–1082, 2017.

[5] Qiyang Han, Tengyao Wang, Sabyasachi Chatterjee, and Richard J. Samworth. Isotonic regression in general dimensions. *The Annals of Statistics*, 47(5):2440 – 2471, 2019.

[6] Xingchao Liu, Xing Han, Na Zhang, and Qiang Liu. Certified monotonic neural networks. 2020.

[7] Ryan J. Tibshirani, Holger Hoefling, and Robert Tibshirani. Nearly-isotonic regression. *Technometrics*, 53(1):54–61, 2011.

A Succinct Explanatory Analysis: Law Data

As mentioned in the main text, for this data set each row corresponds to a law student, and the columns are the student's age, family income, part-time status, gender, undergraduate GPA, LSAT score, and finally whether or not the student passes the bar exam. A description of the data can be found in Table 8.

	lsat	ugpa	fam_inc	age	parttime	pass_bar
mean	151.923154	2.855369	3.600671	29.000000	0.085570	0.927852
std	6.972563	0.558619	0.843123	5.402899	0.279964	0.258950
min	133.500000	1.900000	1.000000	20.483221	0.000000	0.000000
25%	147.000000	2.400000	3.000000	26.483221	0.000000	1.000000
50%	152.000000	2.600000	4.000000	27.483221	0.000000	1.000000
75%	159.000000	3.400000	4.000000	30.483221	0.000000	1.000000
max	159.000000	4.000000	5.000000	82.483221	1.000000	1.000000

Table 8: Description of Law Data Set

It can be seen that both the family income variable and the part-time variables are discretized. In Fig. 7 we present the correlation matrix of all features. It can be seen that undegradiate GPA is highly correlated (0.82) with the LSAT score. For all other features there is weak correlation. In Fig. 8 we present the empirical distribution estimation of the features. It can be seen that for the first three features the distributions are concentrated, stable and do not have severe outliers. For the last three the results are not so balanced and outliers may occur. Regarding outliers we present for the first three features that appear to not have outliers in Fig. 9 the corresponding box plots to verify our observations. It can be seen that our initial guess was almost correct since we do not observe severe outliers except on the age feature.

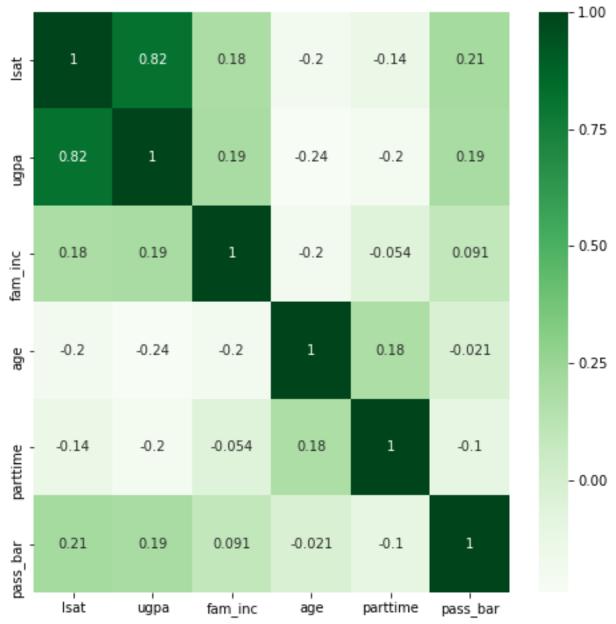


Figure 7: Correlation Matrix of Law Data Set

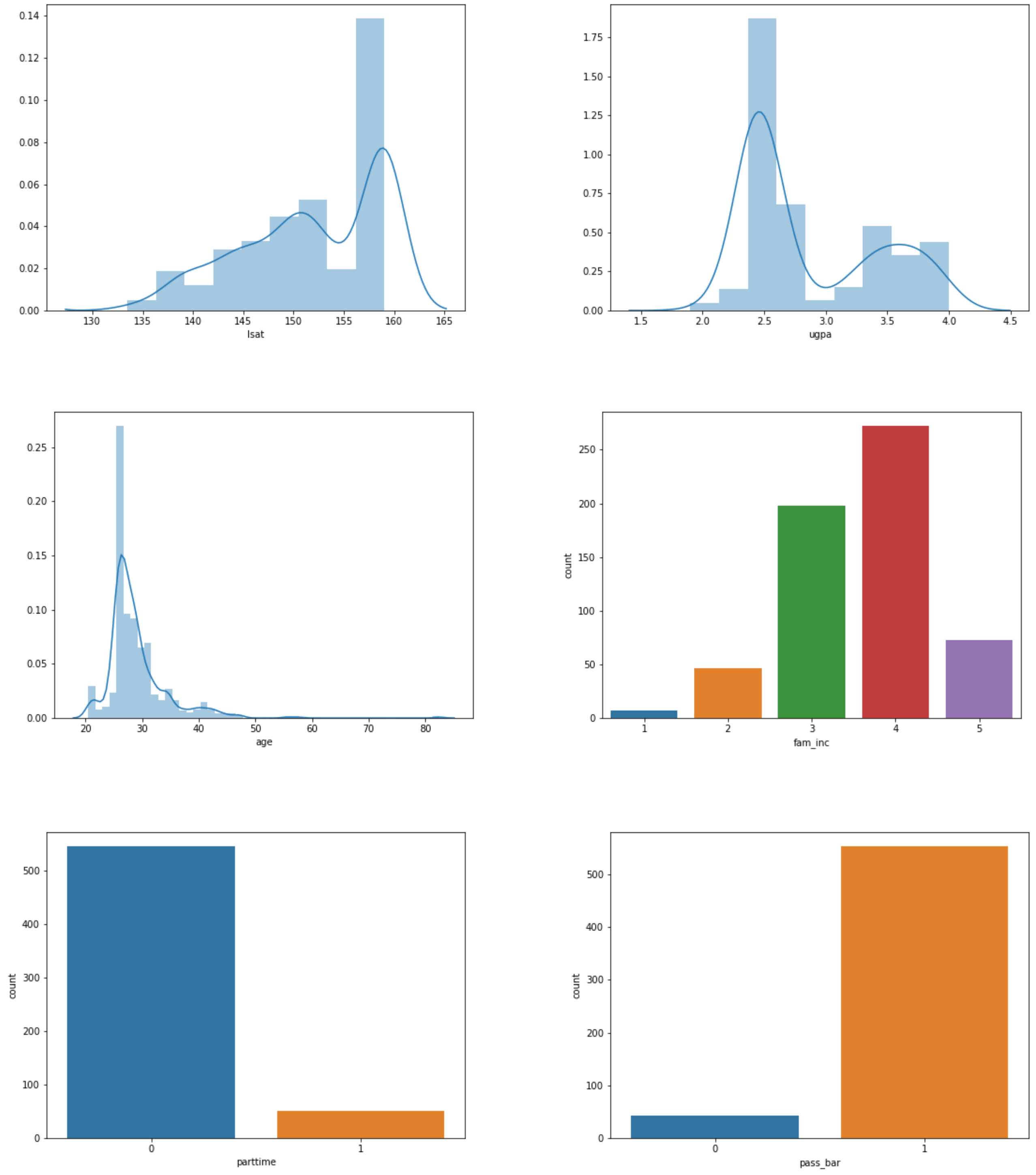


Figure 8: Empirical Distribution of Law Data Set Features

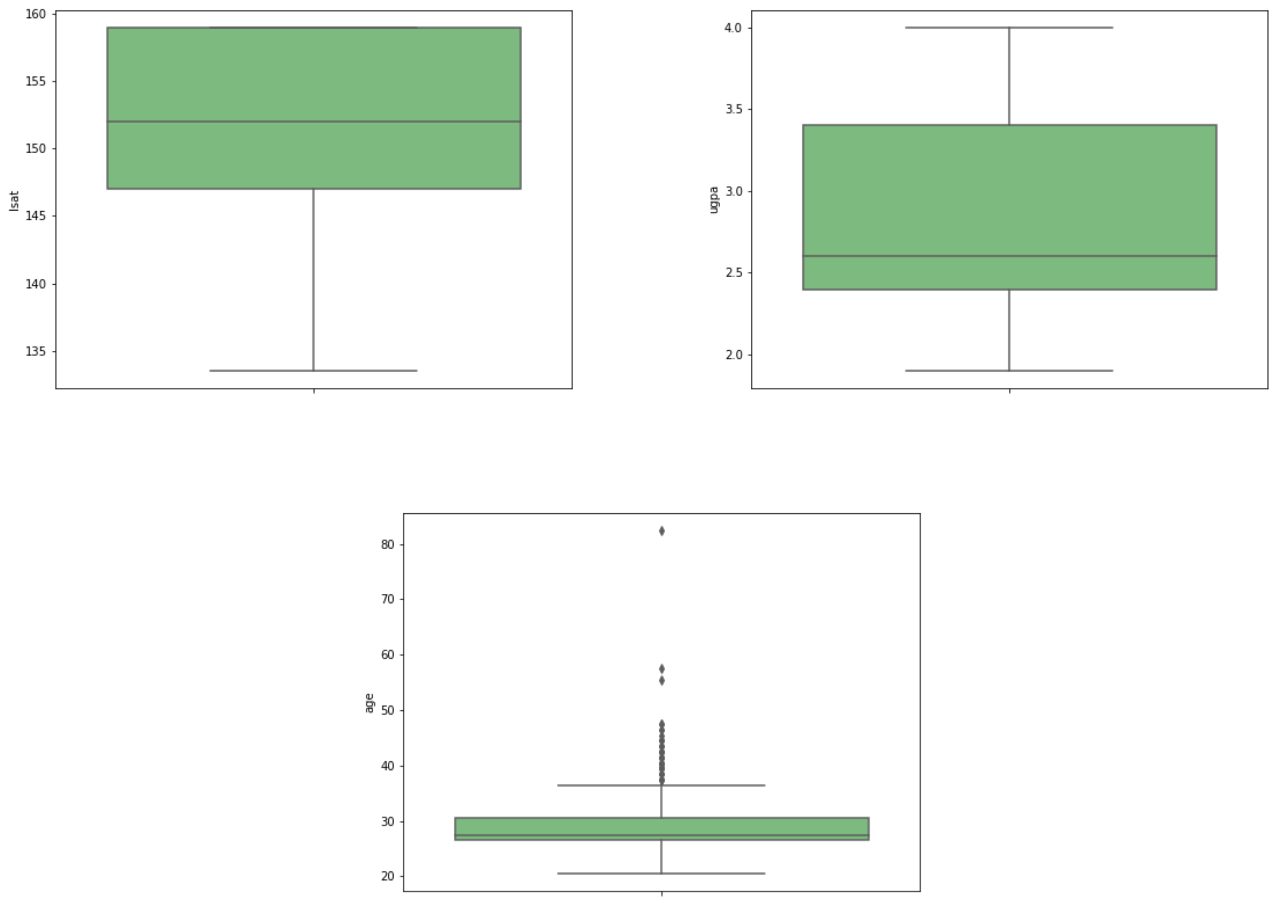


Figure 9: Box Plots of Law Data Set Features

B Succinct Explanatory Analysis: Coffee Data

As mentioned in the main text, for this data set each row corresponds to the sales of a coffee product in a particular week. We subset the data to include only 1 brand and 1 product, and we predict the total volume sold (TOTVOLUME) using the price (PRICEPAID), the promotion percent (PROMPERCENT), the cumulative week (WKCUM), as well as the week of the year (WK). The data we used contains only numerical values and no value is missing. A description of the data can be found in Table 9.

	WKCUM	WK	PRICEPAID	PROMPERCENT	TOTVOLUME
mean	72.776923	24.061538	0.858838	0.178800	137.015385
std	31.748348	16.119108	0.057819	0.047393	95.194781
min	5.000000	5.000000	0.751000	0.100000	19.000000
25%	49.000000	10.000000	0.811250	0.143000	65.250000
50%	74.000000	21.000000	0.850500	0.182000	117.000000
75%	95.000000	42.000000	0.896000	0.219000	178.500000
max	116.000000	50.000000	1.008000	0.254000	485.000000

Table 9: Description of Coffee Data Set

In Fig. 10 we present the correlation matrix of all features. It can be seen that WKCUM is positively correlated (0.59) to PRICEPAID, WK is negatively correlated (-0.55) to PROMPERCENT and PRICEPAID is negatively correlated (-0.44) to PROMPERCENT. For all other features there is weak correlation. In Fig. 11 we present the empirical distribution estimation of the features. It can be seen that all feature distributions are concentrated, stable and do not have severe outliers. Regarding outliers we present in Fig. 12 the corresponding box plots that verify our initial observation (there are though some outliers in TOTVOLUME).

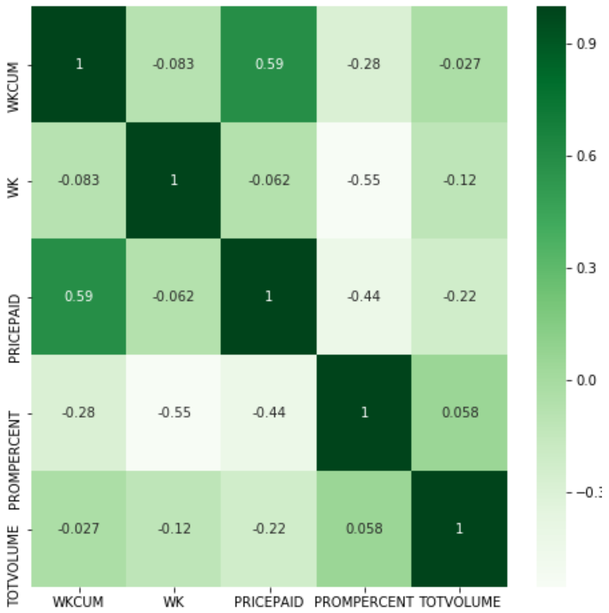


Figure 10: Correlation Matrix of Coffee Data Set

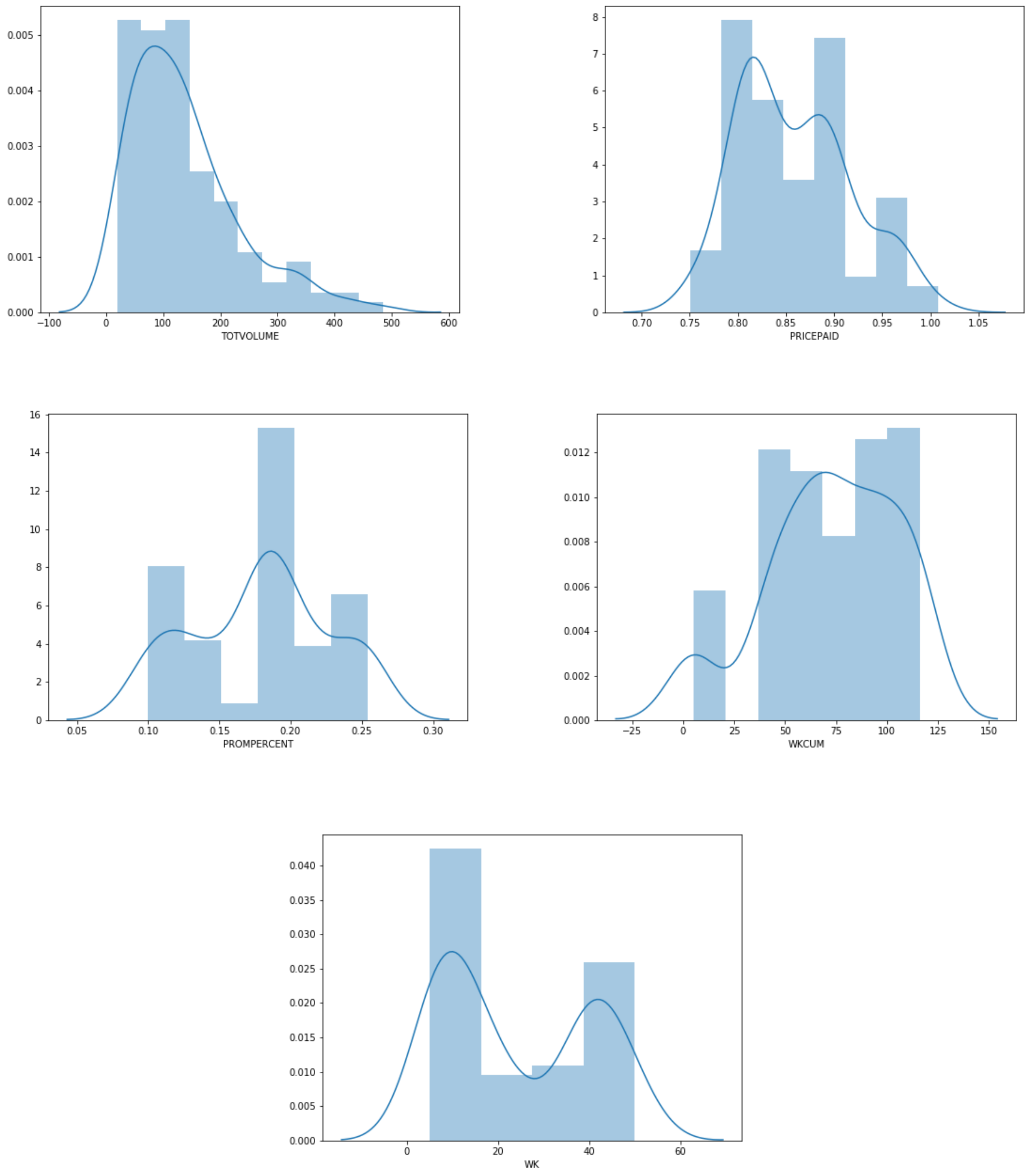


Figure 11: Empirical Distribution of Coffee Data Set Features

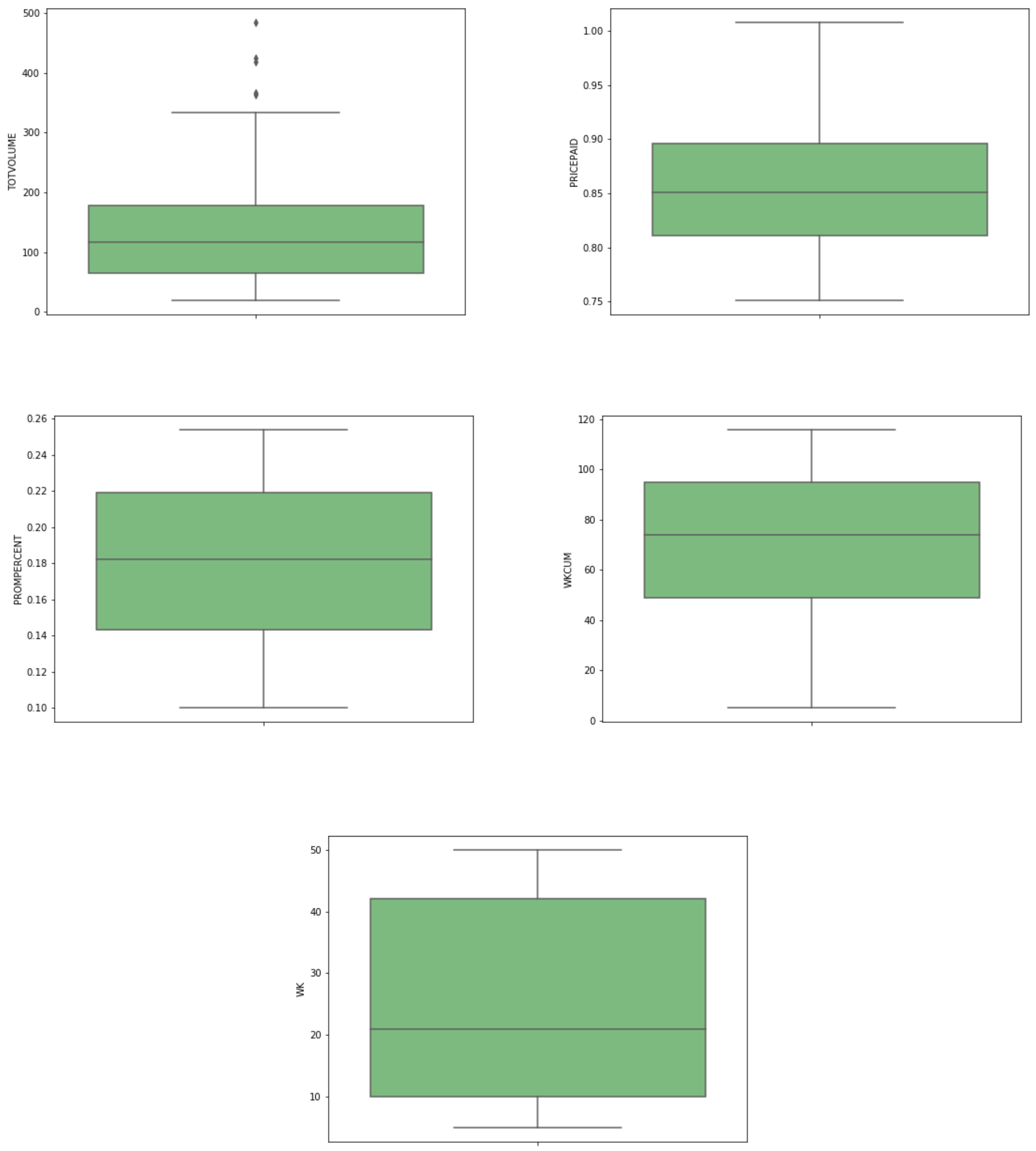


Figure 12: Box Plots of Coffee Data Set Features

C Certified Monotonic Neural Networks

In this work (Liu et al., NeurIPS 2020), the authors propose a method that certifies the monotonicity of the general piece-wise linear neural networks by solving a mixed integer linear programming problem. This method allows training neural networks with heuristic monotonicity regularizations that can be gradually increased until the learned network is certified monotonic. The methods do not require specific constraints on the weight space and can be applied to any network architecture.

C.1 Description of the Method

Monotonicity with respect to certain inputs is a desirable property of the machine learning predictions to effectively address the fairness, interpretability, and generalization issues in practice. We begin by presenting the concept of monotonicity.

Let $f(\boldsymbol{x})$ be a neural network mapping from an input space $\mathcal{X}$ to $\mathbb{R}$. The case where $\mathcal{X}$ is a rectangle region in $\mathbb{R}^d$, i.e. $\mathcal{X} = \bigotimes_{i=1}^{d}[l_i, u_i]$, is considered. The input $\boldsymbol{x}$ is assumed to be partitioned into $\boldsymbol{x} = [\boldsymbol{x}_\alpha, \boldsymbol{x}_{-\alpha}]$, where α is a subset of $[1, ..., d]$, $-\alpha$ is its complement and $\boldsymbol{x}_\alpha := [x_i : i \in \alpha]$ is the corresponding subvector of $\boldsymbol{x}$. Moreover the space of $\boldsymbol{x}_\alpha$ and $\boldsymbol{x}_{-a}$ is denoted by $\mathcal{X}_a$ and $\mathcal{X}_{-a}$. We say that f is partially monotonic w.r.t. $\boldsymbol{x}_\alpha$ if:

$$f(\boldsymbol{x}_\alpha, \boldsymbol{x}_{-\alpha}) \leq f(\boldsymbol{x}'_\alpha, \boldsymbol{x}_{-\alpha}),\ \forall \boldsymbol{x}_a \leq \boldsymbol{x}'_\alpha,\ \forall \boldsymbol{x}_a, \boldsymbol{x}'_\alpha \in \mathcal{X}_\alpha, \boldsymbol{x}_{-a} \in \mathcal{X}_{-\alpha} \tag{10}$$

where $\boldsymbol{x_\alpha} \leq \boldsymbol{x'}_\alpha$ denotes the inequality for all the elements.

Regarding individual monotonicity we the say that a function is not monotonic w.r.t. $\boldsymbol{x}_\alpha$ if we can find another testing example $\hat{\boldsymbol{x}} = [\hat{\boldsymbol{x}}_\alpha, \hat{\boldsymbol{x}}_{-\alpha}]$ such that:

$$f(\hat{\boldsymbol{x}}) > f(\boldsymbol{x}),\ s.t.\ \hat{\boldsymbol{x}}_\alpha \leq \boldsymbol{x}_\alpha, \hat{\boldsymbol{x}}_{-\alpha} = \boldsymbol{x}_{-\alpha} \tag{11}$$

If such examples exist, then the fairness of the system is challenged. Therefore, the authors begin by discussing how to verify individual monotonicity or otherwise find monotonic adversarial examples. More specifically, given a data point $\boldsymbol{x}$ and a neural network f the authors want to either verify the non-existence of any monotonicity examples or detect all of such examples. Detecting an adversarial example can be modeled in the following optimization problem.

$$\hat{\boldsymbol{x}}^*_\alpha = \arg\max_{\boldsymbol{x}' \in \mathcal{X}} f(\boldsymbol{x}'_\alpha, \boldsymbol{x}_{-\alpha})\ s.t.\ \boldsymbol{x}'_\alpha \leq \boldsymbol{x}_\alpha,\ \boldsymbol{x}'_{-\alpha} = \boldsymbol{x}_{-\alpha} \tag{12}$$

If $f(\hat{\boldsymbol{x}}^*) > f(\boldsymbol{x})$, then $\hat{\boldsymbol{x}}^*$ is a monotonic adversarial example. Since most neural networks use piece-wise linear activation functions (such as ReLU e.t.c.), this problem can be transformed from a non-convex optimization problem to a mixed integer linear programming problem as follows. For simplicity we will present the method for a two-layer ReLU network. The method can be easily extended to multi-layer networks. Let $f(\boldsymbol{x})$ be a two-layer ReLU network:

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} a_i ReLU(\boldsymbol{w}_i^T \boldsymbol{x} + b_i) \tag{13}$$

The key observation is that the ReLU activation can be rewritten as the following set of mixed integer linear constraints as follows:

$$y_i = ReLU(\boldsymbol{w}_i^T \boldsymbol{x} + b_i) \iff y_i \in \mathcal{C}(\boldsymbol{x}, \boldsymbol{w}_i, \boldsymbol{b}_i) \tag{14}$$

where

$$\mathcal{C}(\boldsymbol{x}, \boldsymbol{w}_i, \boldsymbol{b}_i) = \{y |\ y \geq 0,\ y \geq \boldsymbol{w}_i^T \boldsymbol{x} + b_i,\ y \leq u_i z,\ y \leq \boldsymbol{w}_i^T \boldsymbol{x} + b_i - l_i(1-z),\ z \in \{0, 1\}\} \tag{15}$$

where z is a binary variable that indicates whether ReLU is activated or not and $u_i = \sup_{\boldsymbol{x} \in \mathcal{X}}\{\boldsymbol{w}_i^T \boldsymbol{x} + b_i\}$ and $l_i = \inf_{\boldsymbol{x} \in \mathcal{X}}\{\boldsymbol{w}_i^T \boldsymbol{x} + b_i\}$ are the maximum and the minimum values of the output respectively. It can be seen that both

u_i and l_i can be calculated easily when $\mathcal{X}$ is a rectangular interval in $\mathbb{R}^d$. So, the optimization problem mentioned in equation (12) can be framed as:

$$\max_{\boldsymbol{x}'} \sum_{i=1}^{n} a_i y_i, \;\; s.t. \; \boldsymbol{x}'_\alpha \leq \boldsymbol{x}_\alpha, \;\; y_i \in \mathcal{C}(\boldsymbol{x}, \boldsymbol{w}_i, b_i), \;\; \forall i \in [n] \tag{16}$$

However, it is also important to check the global monotonicity for all the points in the input domain as well. We know that for a differentiable function f, it is monotonic w.r.t. $\boldsymbol{x}_\alpha$ on $\mathcal{X}$ iff $\partial_{x_l} \geq 0$ for all $l \in \alpha, \boldsymbol{x} \in \mathcal{X}$. This can be checked by solving:

$$U_\alpha := \min_{\boldsymbol{x}, l \in \alpha} \{\partial_{x_l} f(\boldsymbol{x}), \;\; \boldsymbol{x} \in \mathcal{X}\} \tag{17}$$

If $U_\alpha \geq 0$, then monotonicity is verified. This optimization problem can be turned into a MILP for ReLU networks. We have that:

$$\partial_{x_l} f(\boldsymbol{x}) = \sum_{i=1}^{n} \mathbb{1}(\boldsymbol{w}_i^T \boldsymbol{x} + b_i \geq 0) a_i w_{i,l} \tag{18}$$

The indicator function can be transformed into the following set of mixed integer linear constraints as follows:

$$z_i = \mathbb{1}(\boldsymbol{w}_i^T \boldsymbol{x} + b_i \geq 0) \iff z_i \in \mathcal{G}(\boldsymbol{x}, \boldsymbol{w}_i, b_i)$$

where

$$\mathcal{G}(\boldsymbol{x}, \boldsymbol{w}_i, \boldsymbol{b}_i) = \{z_i | \boldsymbol{w}_i^T + b_i \leq u_i z, \boldsymbol{w}_i^T + b_i \geq l_i(1 - z_i), \;\; z_i \in \{0, 1\}\} \tag{19}$$

Therefore, equation (17) can be transformed in the following optimization problem.

$$U_\alpha = \min_{\boldsymbol{x}, l \in \alpha} \{\sum_{i=1}^{n} a_i w_{i,l} z_i \;\; s.t. \; z_i \in \mathcal{G}(\boldsymbol{x}, \boldsymbol{w}_i, b_i), \boldsymbol{x} \in \mathcal{X}\} \tag{20}$$

Such a problem can be solved by commercial MILP solvers. However, it is impractical to obtain exact solution when the number of integers is too large. Fortunately, most MILP solvers can stop earlier, provide a lower bound of the optimal value and therefore verify the monotonicity without solving the problem problem exactly. A simple example of lower bound can be obtained by the linear relaxation.

C.2 Training Procedure

In order to incorporate MILPs into to the training procedure, the authors propose a simple algorithm that trains the network with a data-driving monotonicity regularization and gradually increases the regularization magnitude until the network passes the monotonicity verification. The algorithm for a neural network f is:

Step 1: First, train a neural network f by:

$$\min_f \mathcal{L}(f) + \lambda R(f) \tag{21}$$

where $\mathcal{L}(f)$ is the training loss and $R(f) = \mathbb{E}_{x \sim Uni(\mathcal{X})}[\sum_{l \in \alpha} \max(0, -\partial_{x_l} f(x))^2]$ is a penalty that characterizs the violation of monotonicity. Moreover, λ is the corresponding coefficient and $Uni(\mathcal{X})$ is the uniform distribution on $\mathcal{X}$. $R(f) = 0$ implies that f is monotonic with respect to $\boldsymbol{x}_\alpha$. Since the value of $R(f)$ is intractable, it is approximated by drawing samples in the input domain during iterations of the gradient descent.

Step 2: After a number of epochs, we calculate U_α or a lower bound of it. It is sufficient to verify that $U_\alpha \geq 0$, then f is monotonic and the algorithm terminates. Else, we increase λ and we go back to step 1.