FasterRisk: Fast and Accurate Interpretable Risk Scores

Anonymous Author(s)
Affiliation
Address
email

Abstract

Over the last century, *risk scores* have been the most popular form of predictive model used in healthcare and criminal justice. Risk scores are sparse linear models with integer coefficients; often these models can be memorized or placed on an index card. Typically, risk scores have been created either without data or by rounding logistic regression coefficients, but these methods do not reliably produce high-quality risk scores. Recent work used mathematical programming, which is computationally slow. We introduce an approach for efficiently producing a collection of high-quality risk scores learned from data. Our approach involves producing a pool of almost-optimal sparse continuous solutions, each with a different support set, using a beam-search algorithm. Each of these continuous solutions is transformed into a separate risk score through a "star search," where a range of multipliers are considered before rounding the coefficients sequentially to maintain low logistic loss. Our algorithm returns all of these high-quality risk scores for the user to consider. This method completes within minutes and can be impactful in a broad variety of applications.

1 Introduction

Risk scores are sparse linear models with integer coefficients that predict risks. They are possibly the most popular form of predictive model for high stakes decisions through the last century and are the standard form of model used in criminal justice [3, 19] and medicine [17, 23, 29, 26, 31]. Their history dates back to at least the criminal justice work of Burgess [7], where individuals were assigned integer point scores between 0 and 21 based on their criminal history and demographics that determined their probability of "making good or of failing upon parole." Other famous risk scores are arguably the most widely-used predictive models in healthcare. These include the APGAR score [2], developed in 1952 and given to newborns, and the CHADS_2 score [16], which estimates stroke risk for atrial fibrillation patients. Figure 1 shows an example risk score, which estimates risk of a breast lesion being malignant.

1.	Oval Shape	-2 points		...
2.	Irregular Shape	4 points	+	...
3.	Circumscribed Margin	-5 points	+	...
4.	Spiculated Margin	2 points	+	...
5.	Age $\geq$ 60	3 points	+	...
		SCORE	=	

SCORE	-7	-5	-4	-3	-2	-1
RISK	6.0%	10.6%	13.8%	17.9%	22.8%	28.6%

SCORE	0	1	2	3	4	≥ 5
RISK	35.2%	42.4%	50.0%	57.6%	64.8%	71.4%

Figure 1: Risk score on the mammo dataset [14], whose population is biopsy patients. It predicts risk of malignancy of a breast lesion. Risk score is from FasterRisk on a fold of a 5-CV split.

Risk scores have the benefit of being easily memorized; usually their names reveal the full model – for instance, the factors in CHADS_2

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

are past **C**hronic heart failure, **H**ypertension, **A**ge≥75 years, **D**iabetes, and past **S**troke (where past stroke receives **2** points and the others each receive 1 point). For risk scores, counterfactuals are often trivial to compute, even without a calculator. Also, checking that the data and calculations are correct is easier with risk scores than with other approaches. In short, risk scores have been created by humans for a century to support a huge spectrum of applications, because humans find them easy to understand.

Traditionally, risk scores have been created in two main ways: (1) without data, with expert knowledge only (and validated only afterwards on data), and (2) using a semi-manual process involving manual feature selection and rounding of logistic regression coefficients. That is, these approaches rely heavily on domain expertise and rely little on data. Unfortunately, the converse (relying on data) leads to computationally hard problems: optimizing risk scores over a global objective on data is NP-hard, because in order to produce integer-valued scores, the feasible region must be the integer lattice. There have been only a few approaches to design risk scores automatically [4, 5, 8, 9, 15, 27, 28, 33, 34, 35], but each of these has a flaw that limits its use in practice: the optimization-based approaches use mathematical programming solvers (which require a license) that are slow and scale poorly, and the other methods are randomized greedy algorithms, producing fast but much lower-quality solutions. We need an approach that exhibits the best of both worlds: speed fast enough to operate in a few minutes on a laptop and optimization and search capability as powerful as that of the mathematical programming tools. Our method, FasterRisk, lies at this intersection.

One may wonder why simple rounding of ℓ_1-regularized logistic regression coefficients does not yield sufficiently good risk scores. Past works [32, 34] explain this as follows: the sheer amount of ℓ_1 regularization needed to get a very sparse solution leads to large biases and worse loss values, and rounding goes against the performance gradient. For example, consider a set of ℓ_1 coefficients found as [1.45, .87, .83, .47, .23, .15, ...]. This model would be worse than its unregularized counterpart, because of the bias due to the large ℓ_1 term. Its rounded solution is [1,1,1,0,0,0,..], which leads to even worse loss. One could attempt instead to multiply by a constant and then round, but which constant? There are an infinite number of choices. And, even if some value of the multiplier led to minimal loss due to rounding, the bias from the ℓ_1 term still limits the quality of the solution.

The algorithm presented here does not have these disadvantages. The steps are: (1) Fast subset search with ℓ_0 optimization (avoiding the bias from ℓ_1). This requires the solution of an NP-hard problem, but our fast subset selection algorithm is able to solve this quickly. We proceed from this accurate sparse continuous solution, preserving both sparseness and accuracy in the next steps. (2) Find a pool of diverse continuous sparse solutions that are almost as good as the solution found in (1) but with different support sets. (3) A "star ray" search, where we search for feasible integer-valued solutions along multipliers of each item in the pool from (2). By using multipliers, the search space resembles a ray of a star because it extends each coefficient in the pool outwards from the origin to search for solutions. To find integer solutions, we perform a local search (a form of sequential rounding). This method yields high performance solutions: we provide a theoretical upper bound on the loss difference between the continuous sparse solution and the rounded integer sparse solution.

Through extensive experiments, we show that our proposed method is computationally fast and produces high-quality integer solutions. This work thus provides valuable and novel tools to create risk scores for professionals in many different fields, such as healthcare, finance, and criminal justice.

2 Related Work

Optimization-based approaches: Risk scores, which model $P(y = 1|\boldsymbol{x})$, are different than threshold classifiers, which predict either $y = 1$ or $y = -1$ given $\boldsymbol{x}$. Most work in the area of optimization of integer-valued sparse linear models focuses on classifiers, not risk scores [4, 5, 8, 27, 28, 32, 35, 37]. This difference is important, because a classifier generally cannot be calibrated well for use in risk scoring: only its single decision point is optimized. Despite this, several works use the hinge loss to calibrate predictions [5, 8, 27]. All of these optimization-based algorithms use mathematical programming solvers (i.e., integer programming solvers), which tend to be slow and cannot be used on larger problems. However, they can handle both feature selection and integer constraints.

To directly optimize risk scores, typically the logistic loss would be used. The RiskSLIM algorithm [34] optimizes the logistic loss regularized with ℓ_0 regularization, subject to integer constraints on the coefficients. RiskSLIM uses callbacks to a MIP solver, alternating between solving linear programs and using branch-and-cut to divide and reduce the search space. The branch-and-cut procedure needs to keep track of unsolved nodes, whose number increases exponentially with the size of the feature space. Thus, RiskSLIM's major challenge is scalability.

Local search-based approaches: As discussed earlier, a natural way to produce a scoring system or risk score is by selecting features manually and rounding logistic regression coefficients or hinge-loss solutions to integers [9, 10, 34]. While rounding is fast, rounding errors discussed earlier can cause the solution quality to be much worse than that of the optimization-based approaches. Several works have proposed improvements over traditional rounding. In Randomized Rounding [9], each coefficient is rounded up or down randomly, based on its continuous coefficient value. However, randomized rounding does not seem to perform well in practice. Chevaleyre [9] also proposed Greedy Rounding, where coefficients are rounded sequentially. While this technique provides theoretical guarantees for greedy rounding for the hinge loss, we have identified a serious flaw in this argument, rendering the bounds incorrect (see Appendix B). The RiskSLIM paper [34] proposed SequentialRounding, which, at each iteration, chooses a coefficient to round up or down, making the best choice according to the regularized logistic loss. This gives better solutions than other types of rounding, because the coefficients are considered together through their performance on the loss function, not independently.

A drawback of SequentialRounding is that it considers rounding up or down only to the nearest integer from the continuous solution. By considering *multipliers*, we consider a much larger space of possible solutions. The idea of multipliers (i.e., "scale and round") is used for medical scoring systems [10], though, as far as we know, it has been used only with traditional rounding rather than SequentialRounding, which could easily lead to poor performance, and we have seen no previous work that studies how to perform scale-and-round in a systematic, computationally efficient way. While the general idea of scale-and-round seems simple, it is not: there are an infinite number of possible multipliers, and, for each one, a number of possible nearby integer coefficient vectors that is the size of a hypercube, expanding exponentially in the search space.

Sampling Methods: The Bayesian method of Ertekin et al. [15] samples scoring systems, favoring those that are simpler and more accurate, according to a prior. "Pooling" [34] creates multiple models through sampling along the regularization path of ElasticNet. As discussed, when regularization is tuned high enough to induce sparse solutions, it results in substantial bias and low-quality solutions (see [32, 34] for numerous experiments on this point). Note that there is a literature on finding diverse solutions to optimization problems [1], but it only focuses on linear objective functions.

Contributions: Our contributions include the three-step framework for producing risk scores, the beam-search based algorithm for logistic regression with bounded coefficients, the search algorithm to find pools of diverse high-quality continuous solutions, the star search technique using multipliers, and a theorem guaranteeing the quality of the star search results.

Limitations: FasterRisk does not provide provably optimal solutions to an NP-hard problem, which is how it is able to perform in reasonable time for practitioner's use. FasterRisk's models should not be interpreted as causal. FasterRisk creates very sparse generalized additive models and thus has limited capacity. FasterRisk's models inherit flaws from data it was trained on. FasterRisk is not yet customized to a given application, which can be done in future work.

3 Methodology

Define dataset $\mathcal{D} = \{1, \boldsymbol{x}_i, y_i\}_{i=1}^n$ (1 is a static feature corresponding to the intercept) and scaled dataset as $\frac{1}{m} \times \mathcal{D} = \{\frac{1}{m}, \frac{1}{m}\boldsymbol{x}_i, y_i\}_{i=1}^n$. Our goal is to produce high-quality risk scores within a few minutes on a small personal computer. We start with an optimization problem similar to RiskSLIM's [34], which minimizes the logistic loss subject to sparsity constraints and integer coefficients:

$$\min_{\boldsymbol{w}, w_0} L(\boldsymbol{w}, w_0, \mathcal{D}), \quad \text{where } L(\boldsymbol{w}, w_0, \mathcal{D}) = \textstyle\sum_{i=1}^n \log(1 + \exp(-y_i(\boldsymbol{x}_i^T \boldsymbol{w} + w_0))) \tag{1}$$

$$\text{such that} \quad \|\boldsymbol{w}\|_0 \leq k \text{ and } \boldsymbol{w} \in \mathbb{Z}^p, \ \ \forall j \in [1, .., p] \ w_j \in [-5, 5], \ \ w_0 \in \mathbb{Z}.$$

In practice, the range of these box constraints $[-5, 5]$ is user-defined and can be different for each coefficient. (We use 5 for ease of exposition.) The $\|\boldsymbol{w}\|_0$ or integer constraints make the problem NP-hard, and this is a difficult mixed-integer nonlinear program. Transforming the original features to all possible dummy variables, as done in other methods [20], changes the model into a (flexible) generalized additive model; even when transformed into risk scores, they can still be as accurate as the best machine learning models [34, 36].

Algorithm 1 FasterRisk($\mathcal{D}$,k,C,B,ϵ,T,N_m) $\rightarrow \{(\boldsymbol{w}^{+t}, w_0^{+t}, m_t)\}_t$

Input: dataset $\mathcal{D}$ (consisting of feature matrix $\boldsymbol{X} \in \mathbb{R}^{n \times p}$ and labels $\boldsymbol{y} \in \mathbb{R}^n$), sparsity constraint k, coefficient constraint $C = 5$, beam search size $B = 10$, tolerance level $\epsilon = 0.3$, number of attempts $T = 50$, number of multipliers to try $N_m = 20$.

Output: a pool P of scoring systems $\{(\boldsymbol{w}^t, w_0^t), m^t\}$ where t is the index enumerating all found scoring systems with $\|\boldsymbol{w}^t\|_0 \le k$ and $\|\boldsymbol{w}^t\|_\infty \le C$ and m^t is the corresponding multiplier.

1: Call Algorithm 2 SparseBeamLR($\mathcal{D}, k, C, B$) to find a high-quality solution $(\boldsymbol{w}^*, w_0^*)$ to the sparse logistic regression problem with continuous coefficients satisfying a box constraint, i.e., solve Problem (1). (Algorithm SparseBeamLR will call Algorithm ExpandSuppBy1 as a subroutine, which grows the solution by beam search.)

2: Call Algorithm 5 CollectSparseDiversePool($(\boldsymbol{w}^*, w_0^*), \epsilon, T$), which solves Problem (4). Place its output $\{(\boldsymbol{w}^t, w_0^t)\}_t$ in pool P. $P \leftarrow P \cup \{(\boldsymbol{w}^t, w_0^t)\}_t$.

3: Send each member t in the pool P, which is $(\boldsymbol{w}^t, w_0^t)$, to Algorithm 3 StarRaySearch $(\mathcal{D}, (\boldsymbol{w}^t, w_0^t), C, N_m)$ to perform a line search among possible multiplier values and obtain an integer solution $(\boldsymbol{w}^{+t}, w_0^{+t})$ with multiplier m_t. Algorithm 3 calls Algorithm 6 Auxiliary-LossRounding which conducts the rounding step.

Return the collection of risk scores $\{(\boldsymbol{w}^{+t}, w_0^{+t}, m_t)\}_t$. If desired, return only the best model according to the logistic loss.

To make the solution space substantially larger than $[-5, -4, ..., 4, 5]^p$, we use *multipliers*. The problem becomes:

$$\min_{\boldsymbol{w}, w_0, m} L\left(\boldsymbol{w}, w_0, \frac{1}{m}\mathcal{D}\right), \text{ where } L\left(\boldsymbol{w}, w_0, \frac{1}{m}\mathcal{D}\right) = \sum_{i=1}^{n} \log\left(1 + \exp\left(-y_i \frac{\boldsymbol{x}_i^T \boldsymbol{w} + w_0}{m}\right)\right) \tag{2}$$
$$\text{such that } \|\boldsymbol{w}\|_0 \le k, \boldsymbol{w} \in \mathbb{Z}^p, \quad \forall j \in [1, .., p] \; w_j \in [-5, 5], \quad w_0 \in \mathbb{Z}, \quad m > 0.$$

Note that the use of multipliers does not weaken the interpretability of the risk score: the user still sees integer risk scores comprised of values $w_j \in \{-5, -4, .., 4, 5\}$, $w_0 \in \mathbb{Z}$ and points are computed from them. Only the risk conversion table is calculated differently, as $P(Y = 1|\boldsymbol{x}) = 1/(1 + e^{-f(\boldsymbol{x})})$ where $f(\boldsymbol{x}) = \frac{1}{m}(\boldsymbol{w}^T \boldsymbol{x} + w_0)$.

Our method proceeds in three steps, as outlined in Algorithm 1. In the first step, it approximately solves the following **sparse logistic regression** problem with a box constraint (but not integer constraints), detailed in Section 3.1 and Algorithm 2.

$$(\boldsymbol{w}^*, w_0^*) \in \operatorname*{argmin}_{\boldsymbol{w}, w_0} L(\boldsymbol{w}, w_0, \mathcal{D}), \; \|\boldsymbol{w}\|_0 \le k, \boldsymbol{w} \in \mathbb{R}^p, \forall j \in [1, ..., p], \; \boldsymbol{w}_j \in [-5, 5], w_0 \in \mathbb{R}. \tag{3}$$

The algorithm gives an accurate and sparse real-valued solution $(\boldsymbol{w}^*, w_0^*)$.

The second step produces **many near-optimal sparse logistic regression solutions**, again without integer constraints, detailed in Section 3.2 and Algorithm 5. Algorithm 5 uses $(\boldsymbol{w}^*, w_0^*)$ from the first step to find a set $\{(\boldsymbol{w}^t, w_0^t)\}_t$ such that for all t and a given threshold $\epsilon_{\boldsymbol{w}}$:

$$(\boldsymbol{w}^t, w_0^t) \text{ obeys } L(\boldsymbol{w}^t, w_0^t, \mathcal{D}) \le L(\boldsymbol{w}^*, w_0^*, \mathcal{D}) \times (1 + \epsilon_{\boldsymbol{w}}) \tag{4}$$
$$\|\boldsymbol{w}^t\|_0 \le k, \; \boldsymbol{w}^t \in \mathbb{R}^p, \quad \forall j \in [1, ..., p], \; w_j^t \in [-5, 5], w_0^t \in \mathbb{R}.$$

After these steps, we have a pool of almost-optimal sparse logistic regression models. In the third step, for each coefficient vector in the pool, we **compute a risk score**. It is a feasible integer solution $(\boldsymbol{w}^{+t}, w_0^{+t})$ to the following, which includes a positive multiplier $m^t > 0$:

$$L\left(\boldsymbol{w}^{+t}, w_0^{+t}, \frac{1}{m^t}\mathcal{D}\right) \le L(\boldsymbol{w}^t, w_0^t, \mathcal{D}) + \epsilon_t, \tag{5}$$
$$\boldsymbol{w}^{+t} \in \mathbb{Z}^p, \quad \forall j \in [1, ..., p], w_j^+ \in [-5, 5], w_0 \in \mathbb{Z},$$

where we derive a tight theoretical upper bound on ϵ_t. A detailed solution to (5) is shown in Algorithm 6 in Appendix A. We do this for a large range of multipliers in Algorithm 3. This third step yields a large collection of risk scores, all of which are approximately as accurate as the best sparse logistic regression model that can be obtained. All steps in this process are fast and scalable.

Algorithm 2 SparseBeamLR($\mathcal{D}$,k,C,B) $\rightarrow$ $(\boldsymbol{w}, w_0)$

Input: dataset $\mathcal{D}$, sparsity constraint k, coefficient constraint C, and beam search size B.
Output: a sparse continuous coefficient vector $(\boldsymbol{w}, w_0)$ with $\|\boldsymbol{w}\|_0 = k$, $\|\boldsymbol{w}\|_\infty \leq C$.

1: Define N_+ and N_- as numbers of positive and negative labels, respectively.
2: $w_0 \leftarrow \log(-N_+/N_-)$, $\boldsymbol{w} \leftarrow \mathbf{0}$ ▷*Initialize the intercept and coefficients.*
3: $\mathcal{F} \leftarrow \emptyset$ ▷*Initialize the collection of found supports as an empty set*
4: $(\mathcal{W}, \mathcal{F}) \leftarrow$ ExpandSuppBy1$(\mathcal{D}, (\boldsymbol{w}, w_0), \mathcal{F}, B)$.
5: **for** $t = 2, ..., k$ **do** ▷*Beam search to expand the support*
6: $\quad\mathcal{W}_{\text{tmp}} \leftarrow \emptyset$
7: $\quad$**for** $(\boldsymbol{w}', w_0') \in \mathcal{W}$ **do** ▷*Each of these has support $t-1$*
8: $\quad\quad(\mathcal{W}', \mathcal{F}) \leftarrow$ ExpandSuppBy1$(\mathcal{D}, (\boldsymbol{w}', w_0'), \mathcal{F}, B)$. ▷*Returns $\leq B$ vectors with supp. t.*
9: $\quad\quad\mathcal{W}_{\text{tmp}} \leftarrow \mathcal{W}_{\text{tmp}} \cup \mathcal{W}'$
10: $\quad$**end for**
11: $\quad$Reset $\mathcal{W}$ to be the B solutions in $\mathcal{W}_{\text{tmp}}$ with the smallest logistic loss values.
12: **end for**
13: Pick $(\boldsymbol{w}, w_0)$ from $\mathcal{W}$ with the smallest logistic loss.
14: Return $(\boldsymbol{w}, w_0)$.

3.1 High-quality Sparse Continuous Solution

There are many different approaches for sparse logistic regression, including ℓ_1 regularization [30], ElasticNet [38], ℓ_0 regularization [12, 20], orthogonal matching pursuit (OMP) [13, 21], but none of these approaches seem to be able to handle both the box constraints and the sparsity constraint in Problem 3, so we developed a new approach. This approach, in Algorithm 2, SparseBeamLR, uses beam search for best subset selection: each iteration contains several coordinate descent steps to determine whether a new variable should be added to the support, and it clips coefficients to the box $[-5, 5]$ as it proceeds. Hence the algorithm is able to determine, before committing to the new variable, whether it is likely to decrease the loss while obeying the box constraints. This beam search algorithm for solving (3) implicitly uses the assumption that one of the best models of size k implicitly contains variables of one of the best models of size $k-1$. This type of assumption has been studied in the sparse learning literature [13] (Theorem 5). However, we are not aware of other works applying box constraints or beam search for sparse logistic regression. In Appendix E, we show that our proposed method has higher solution qualities than the OMP method presented in [13].

Algorithm 2 calls the ExpandSuppBy1 Algorithm, which has two major steps. The detailed algorithm can be found in Appendix A. For the first step, given a solution $\boldsymbol{w}$, we perform optimization on each single coordinate j outside of the current support $supp(\boldsymbol{w})$:

$$d_j^* = \underset{d \in [-5,5]}{\operatorname{argmin}} L(\boldsymbol{w} + d\boldsymbol{e}_j, w_0, \mathcal{D}) \text{ for } \forall j \text{ where } w_j = 0. \tag{6}$$

We find d_j^* for each j through an iterative thresholding operation, which is done on all coordinates in parallel, iterating several (~ 10) times:

$$\text{for iteration } i\text{: } d_j \leftarrow \text{Threshold}(j, d_j, \boldsymbol{w}, w_0, \mathcal{D}) := \min(\max(c_{d_j}, -5), 5), \tag{7}$$

where $c_{d_j} = d_j - \frac{1}{l_j}\nabla_j L(\boldsymbol{w} + d_j\boldsymbol{e}_j, w_0, \mathcal{D})$, and l_j is a Lipschitz constant on coordinate j. Importantly, we can perform Equation 7 on all j where $w_j = 0$ in parallel using matrix form.

For the second step, after the parallel single coordinate optimization is done, we pick the top B indices (j's) with the smallest logistic losses $L(\boldsymbol{w} + d_j^*\boldsymbol{e}_j)$ and fine tune on the new support:

$$\boldsymbol{w}_{\text{new}}^j, {w_0}_{\text{new}}^j \in \underset{\boldsymbol{a} \in [-5,5]^p, b}{\operatorname{argmin}} L(\boldsymbol{a}, b, \mathcal{D}) \text{ with } supp(\boldsymbol{a}) = supp(\boldsymbol{w}) \cup \{j\}. \tag{8}$$

This can be done again using a variant of Equation 7 iteratively on all the coordinates in the new support. We get B pairs of $(\boldsymbol{w}_{\text{new}}^j, {w_0}_{\text{new}}^j)$ through this ExpandSuppBy1 procedure, and the collection of these pairs form the set $\mathcal{W}'$ in Line 8 of Algorithm 2. The ExpandSuppBy1 method is computationally efficient because we are doing parallel single coordinate optimization. This gives the fine-tuning procedure a warm start.

3.2 Collect Sparse Diverse Pool

We now collect the sparse diverse pool. In Section 3.1, our goal was to find a sparse model $(\boldsymbol{w}^*, w_0^*)$ with the smallest logistic loss. For high dimensional features or in the presence of highly correlated features, there could exist many sparse models with almost equally good performance [6]. Let us find those and turn them into risk scores. We first predefine a tolerance gap level ϵ (usually set to 0.3). Then, we delete a feature with index j_- in the support $\text{supp}(\boldsymbol{w}^*)$ and add a new feature with index j_+. We select each new index to be j_+ whose logistic loss is within the tolerance gap:

$$\text{Find all } j_+ \text{ s.t. } \min_{a \in [-5,5]} L(\boldsymbol{w}^* - w^*_{j-}\boldsymbol{e}_{j-} + a\boldsymbol{e}_{j+}, w_0, \mathcal{D}) \leq L(\boldsymbol{w}^*, w_0^*, \mathcal{D})(1+\epsilon). \tag{9}$$

We fine-tune the coefficients on each of the new supports and then save the new solution in our pool. Details can be found in Algorithm 5. Swapping one feature at a time is computationally efficient, and our experiments show it produces sufficiently diverse pools over many datasets.

3.3 "Star" Search for Integer Solutions

Algorithm 3 StarRaySearch$(\mathcal{D}, (\boldsymbol{w}, w_0), C, N_m) \rightarrow (\boldsymbol{w}^+, w_0^+), m$

Input: dataset $\mathcal{D}$, a sparse continuous solution $(\boldsymbol{w}, w_0)$, coefficient constraint C, and number of multipliers to try N_m.
Output: a sparse integer solution $(\boldsymbol{w}^+, w_0^+)$ with $\|\boldsymbol{w}^+\|_\infty \leq C$ and multiplier m.

1: Define $m_{\max} \leftarrow C/\max|\boldsymbol{w}|$ as discussed in Section 3.3. If $m_{\max} = 1$, set $m_{\min} \leftarrow 0.5$; if $m_{\max} > 1$, set $m_{\min} \leftarrow 1$.
2: Pick N_m equally spaced multiplier values $m_l \in [m_{\min}, m_{\max}]$ for $l \in [1, ..., N_m]$ and call this set $\mathcal{M} = \{m_l\}_l$.
3: Use each multiplier to scale the good continuous solution $(\boldsymbol{w}, w_0)$, to obtain $(m_l\boldsymbol{w}, m_l w_0)$, which is a good continuous solution to the rescaled dataset $\frac{1}{m_l}\mathcal{D}$.
4: Send each rescaled solution $(m_l\boldsymbol{w}, m_l w_0)$ and its rescaled dataset $\frac{1}{m_l}\mathcal{D}$ to Algorithm 6 AuxiliaryLossRounding$(\frac{1}{m_l}\mathcal{D}, m_l\boldsymbol{w}, m_l w_0)$ for rounding. It returns $(\boldsymbol{w}^{+l}, w_0^{+l}, m_l)$, where $(\boldsymbol{w}^{+l}, w_0^{+l})$ is close to $(m_l\boldsymbol{w}, m_l w_0)$, and where $(\boldsymbol{w}^{+l}, w_0^{+l})$ on $\frac{1}{m_l}\mathcal{D}$ has a small logistic loss.
5: Evaluate the logistic loss to pick the best multiplier $l^* \in \text{argmin}_l\, L(\boldsymbol{w}^{+l}, w_0^{+l}, \frac{1}{m^l}\mathcal{D})$
6: Return $(\boldsymbol{w}^{+l^*}, w_0^{+l^*})$ and m_{l^*}.

The last challenge is how to get an integer solution from a continuous solution. To achieve this, we use a "star" search that searches along each "ray" of the star, extending each continuous solution outward from the origin using many values of a multiplier, as shown in Algorithm 3. The star search provides much more flexibility in finding a good integer solution than simple rounding. The largest multiplier $m_{\max}$ is set to $5/\max(|w^*|)$ which will take one of the coefficients to the boundary of the box constraint at 5. We set the smallest multiplier to be 1.0 and pick N_m (usually 20) equally spaced points from $[m_{\min}, m_{\max}]$. If $m_{\max} = 1$, we set $m_{\min} = 0.5$ to allow shrinkage of the coefficients. We scale the coefficients and datasets with each multiplier and round the coefficients to integers using the sequential rounding technique in Algorithm 6. For each continuous solution (each "ray" of the "star"), we report the integer solution and multiplier with the smallest logistic loss. This process yields our collection of risk scores. Note here that a standard line search along the multiplier would not work because the rounding error is highly non-convex.

We briefly discuss how the sequential rounding technique works. Details of this method can be found in Appendix A. We initialize $\boldsymbol{w}^+ = \boldsymbol{w}$. Then we round the fractional part of $\boldsymbol{w}^+$ one coordinate at a time. At each step, some of the w_j^+'s are integer-valued (so $w_j^+ - w_j$ is nonzero) and we pick the coordinate and rounding operation (either floor or ceil) based on which can minimize the following objective function, where we will round to an integer at coordinate r^*:

$$r^*, v^* \in \underset{r,v}{\text{argmin}} \sum_{i=1}^{n} l_i^2 \left(x_{ir}(v - w_r) + \sum_{j \neq r} x_{ij}(w_j^+ - w_j) \right)^2, \tag{10}$$
$$\text{subject to } r \in \{j \mid w_j^+ \notin \mathbb{Z}\} \text{ and } v \in \{\lfloor w_r^+ \rfloor, \lceil w_r^+ \rceil\},$$

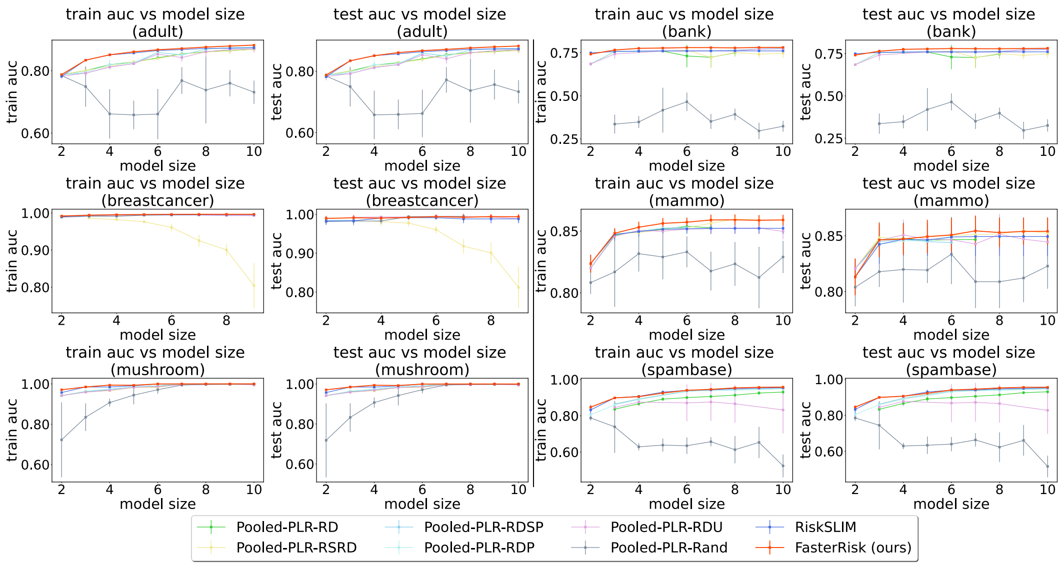

Figure 2: Performance comparison. FasterRisk outperforms baselines due to larger hypothesis space.

where l_i is the Lipschitz constant restricted to the rounding interval[1] and can be computed as $l_i = 1/(1 + \exp(y_i \boldsymbol{x}_i^T \boldsymbol{\gamma}_i))$ with $\gamma_{ij} = \lfloor w_j \rfloor$ if $y_i x_{ij} > 0$ and $\gamma_{ij} = \lceil w_j \rceil$ otherwise. After we select r^* and find value v^*, we update $\boldsymbol{w}^+$ through $w^+_{r^*} = v^*$. We repeat this process until $\boldsymbol{w}^+$ is on the integer lattice: $\boldsymbol{w}^+ \in \mathbb{Z}^p$. The objective function in Equation 10 can be understood as an auxiliary upper bound of the logistic loss. Our algorithm provides an upper bound on the difference between the logistic losses of the continuous solution and the final rounded solution before we start the rounding algorithm (See Theorem 3.1). Additionally, during the sequential rounding procedure, we do not need to perform expensive operations such as logarithms or exponentials as required by the logistic loss function; the bound and auxiliary function require only sums of squares, not logarithms or exponentials. Its derivation and proof are in Appendix C.

Theorem 3.1. *Let $\boldsymbol{w}$ be the real-valued coefficients for the logistic regression model with objective function $L(\boldsymbol{w}) = \sum_{i=1}^n \log(1 + \exp(-y_i \boldsymbol{x}_i^T \boldsymbol{w}))$ (the intercept is incorporated). Let $\boldsymbol{w}^+$ be the integer-valued coefficients returned by the AuxiliaryLossRounding method. Furthermore, let $u_j = w_j - \lfloor w_j \rfloor$. Let $l_i = 1/(1 + \exp(y_i \boldsymbol{x}_i^T \boldsymbol{\gamma}_i))$ with $\gamma_{ij} = \lfloor w_j \rfloor$ if $y_i x_{ij} > 0$ and $\gamma_{ij} = \lceil w_j \rceil$ otherwise. Then, we have an upper bound on the difference between the loss $L(\boldsymbol{w})$ and the loss $L(\boldsymbol{w}^+)$:*

$$L(\boldsymbol{w}^+) - L(\boldsymbol{w}) \leq \sqrt{n \sum_{i=1}^n \sum_{j=1}^p (l_i x_{ij})^2 u_j (1 - u_j)}. \tag{11}$$

Note. *Our method has a higher prediction capacity than RiskSLIM: its search space is much larger.* Compared to RiskSLIM, our use of the multiplier permits a number of solutions that grows exponentially in k as we increase the multiplier. To see this, consider that for each support of k features, since logistic loss is convex, it contains a hypersphere in coefficient space. The volume of that hypersphere is (as usual) $V = \frac{\pi^{k/2}}{\Gamma(\frac{k}{2}+1)} r^k$ where r is the radius of the hypersphere. If we increase the multiplier to 2, the grid becomes finer by a factor of 2, which is equivalent to increasing the radius by a factor of 2. Thus, the volume increases by a factor of 2^k. In general, for maximum multiplier m, the search space is increased by a factor of m^k over RiskSLIM.

4 Experiments

Our experiments focus on three questions: (1) How good is FasterRisk's solution quality compared to baselines? (§4.1) (2) How fast is FasterRisk compared with the state-of-the-art? (§4.2) (3) How

[1]The Lipschitz constant here is much smaller than the one in Section 3.1 due to the interval restriction.

does each of our proposed technique, including sparse beam search, diverse pool, and multipliers, contribute to our solution quality? (see Appendix E)

We compare with RiskSLIM (the current state-of-the-art), as well as algorithms Pooled-PLR-RD, Pooled-PLR-RSRD, Pooled-PRL-RDSP, Pooled-PLR-Rand and Pooled-PRL-RDP. These algorithms were all previously shown to be inferior to RiskSLIM [34]. These methods first find a pool of sparse continuous solutions using different regularizations of ElasticNet (hence the name "Pooled Penalized Logistic Regression" – Pooled-PLR) and then round the coefficients with different techniques. Details are in Appendix D.3. The best solution is chosen from this pool of integer solutions that obeys the sparsity and box constraints and has the smallest logistic loss. For each dataset, we perform 5-fold cross validation and report training and test AUC. Details about datasets, experimental setup, evaluation metrics, loss values, and computing platform/environment can be found in Appendix D. More experimental results appear in Appendix E. Code from [9, 15, 27, 28] is not publicly available.

4.1 Solution Quality

We first evaluate FasterRisk's solution quality. Figure 2 shows the training and test AUC on six datasets (results for training loss appear in Appendix E). FasterRisk (the red line) outperforms all baselines, consistently obtaining the highest AUC scores on both the training and test sets. Notably, our method obtains better results than RiskSLIM, which uses a mathematical solver and is the current state-of-the-art method for scoring systems. This superior performance is due to the use of multipliers, which increases the complexity of the hypothesis space. A more detailed comparison between FasterRisk and RiskSLIM appears in Figure 3.

FasterRisk performs significantly better than the other baselines for two reasons. First, the continuous sparse solutions produced by ElasticNet are low quality for very sparse models. Second, it is difficult to obtain an exact model size by controlling ℓ_1 regularization. For example, Pooled-PLR-RD and Pooled-PLR-RDSP do not have results for model size 10 on the mammo datasets, because such a model size does not exist in the pooled solutions after rounding.

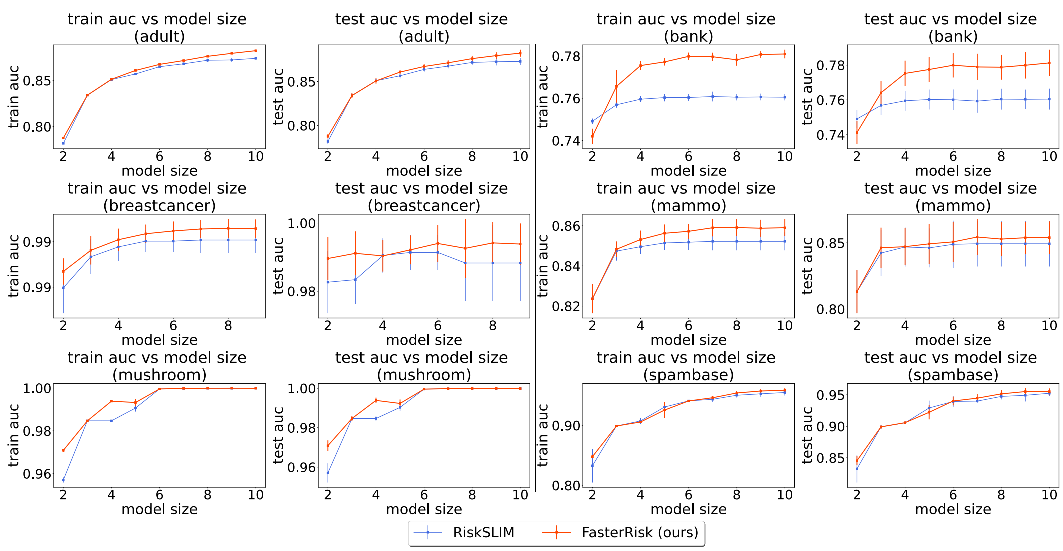

Figure 3: Performance comparison between FasterRisk and RiskSLIM.

4.2 Runtime Comparison

The major drawback of RiskSLIM is its limited scalability. Figure 4 shows that FasterRisk (red bars) is significantly faster than RiskSLIM (blue bars) in general. We ran these experiments with a 900 second (15 minute) timeout. RiskSLIM finishes running on small datasets (mammo and breast cancer), but it times out on the larger datasets, timing out on models larger than 3 features for bank and spambase, larger than 4 features for adult, and larger than 7 features for mushroom. Thus, we see that FasterRisk is both faster and more accurate than RiskSLIM.

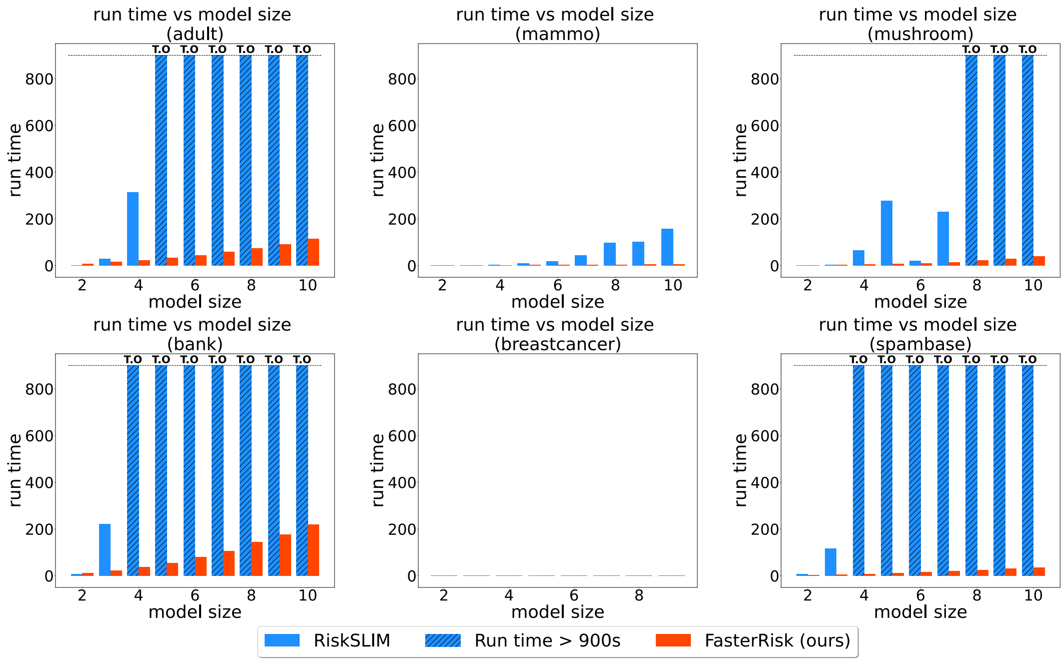

Figure 4: Runtime Comparison. Runtime (in seconds) versus model size for our method FasterRisk (in red) and the RiskSLIM (in blue). The shaded blue bars indicate cases that timed out at 900 seconds. Breastcancer is small and takes approximately 2 seconds for both algorithms.

4.3 Example Scoring Systems

The main benefit of risk scores is their interpretability. We place a few example risk scores in Table 1 to allow the reader to judge for themselves. More risk scores examples can be found in Appendix F.

1.	no high school diploma	-4 points		...
2.	high school diploma only	-2 points	+	...
3.	age 22 to 29	-2 points	+	...
4.	any capital gains	3 points	+	...
5.	married	4 points	+	...
		SCORE	=	

SCORE	<-4	-3	-2	-1	0
RISK	<1.3%	2.4%	4.4%	7.8%	13.6%

SCORE	1	2	3	4	7
RISK	22.5%	35.0%	50.5%	65.0%	92.2%

(a) FasterRisk models for the adult dataset, predicting salary> 50K.

1.	odor=almond	-5 points		...
2.	odor=anise	-5 points	+	...
3.	odor=none	-5 points	+	...
4.	odor=foul	5 points	+	...
5.	gill size=broad	-3 points	+	...
		SCORE	=	

SCORE	-8	-5	-3	≥2
RISK	1.62%	26.4%	73.6%	>99.8%

(b) FasterRisk model for the mushroom dataset, predicting whether a mushroom is poisonous.

Table 1: Example FasterRisk models

5 Conclusion

FasterRisk produces a collection of high-quality risk scores within minutes. Its performance owes to three key ideas: a better algorithm for sparsity and box-constrained continuous models, using a pool of diverse solutions, and the use of the star search, which leverages multipliers and a new sequential rounding technique. FasterRisk is suitable for high-stakes decisions, and permits domain experts a collection of interpretable models to choose from.

References

[1] Izuwa Ahanor, Hugh Medal, and Andrew C. Trapp. Diversitree: Computing diverse sets of near-optimal solutions to mixed-integer optimization problems. *arXiv*, 2022.

[2] Virginia Apgar. A proposal for a new method of evaluation of the newborn infant. *Current Researches in Anesthesia and Analgesia*, 1953(32):260–267, 1953.

[3] James Austin, Roger Ocker, and Avi Bhati. Kentucky pretrial risk assessment instrument validation. *Bureau of Justice Statistics*, 2010.

[4] Lieven Billiet, Sabine Van Huffel, and Vanya Van Belle. Interval coded scoring extensions for larger problems. In *Proceedings of the IEEE Symposium on Computers and Communications*, pages 198–203. IEEE, 2017.

[5] Lieven Billiet, Sabine Van Huffel, and Vanya Van Belle. Interval Coded Scoring: A toolbox for interpretable scoring systems. *PeerJ Computer Science*, 4:e150, 04 2018.

[6] Leo Breiman. Statistical modeling: The two cultures. *Statistical Science*, 16(3):199–231, 2001.

[7] Ernest W Burgess. Factors determining success or failure on parole. Illinois Committee on Indeterminate-Sentence Law and Parole Springfield, IL, 1928.

[8] E. Carrizosa, A. Nogales-Gómez, and D. Romero Morales. Strongly agree or strongly disagree?: Rating features in support vector machines. Technical report, Saïd Business School, University of Oxford, UK, 2013.

[9] Yann Chevaleyre, Frédéerick Koriche, and Jean-Daniel Zucker. Rounding methods for discrete linear classification. In *International Conference on Machine Learning*, pages 651–659. PMLR, 2013.

[10] TJ Cole. Algorithm as 281: scaling and rounding regression coefficients to integers. *Applied statistics*, pages 261–268, 1993.

[11] Lorrie Faith Cranor and Brian A LaMacchia. Spam! *Communications of the ACM*, 41(8):74–83, 1998.

[12] Antoine Dedieu, Hussein Hazimeh, and Rahul Mazumder. Learning sparse classifiers: Continuous and mixed integer optimization perspectives. *arXiv preprint arXiv:2001.06471*, 2020.

[13] Ethan R Elenberg, Rajiv Khanna, Alexandros G Dimakis, and Sahand Negahban. Restricted strong convexity implies weak submodularity. *The Annals of Statistics*, 46(6B):3539–3568, 2018.

[14] Matthias Elter, Rüdiger Schulz-Wendtland, and Thomas Wittenberg. The prediction of breast cancer biopsy outcomes using two cad approaches that both emphasize an intelligible decision process. *Medical physics*, 34(11):4164–4172, 2007.

[15] Sëyda Ertekin and Cynthia Rudin. A bayesian approach to learning scoring systems. *Big Data*, 2015.

[16] Brian F Gage, Amy D Waterman, William Shannon, Michael Boechler, Michael W Rich, and Martha J Radford. Validation of clinical classification schemes for predicting stroke. *The Journal of the American Medical Association*, 285(22):2864–2870, 2001.

[17] Ronald C Kessler, Lenard Adler, Minnie Ames, Olga Demler, Steve Faraone, EVA Hiripi, Mary J Howes, Robert Jin, Kristina Secnik, Thomas Spencer, and et al. The world health organization adult ADHD self-report scale (ASRS): a short screening scale for use in the general population. *Psychological Medicine*, 35(02):245–256, 2005.

[18] Ron Kohavi et al. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In *Kdd*, volume 96, pages 202–207, 1996.

[19] Edward Latessa, Paula Smith, Richard Lemke, Matthew Makarios, and Christopher Lowenkamp. Creation and validation of the ohio risk assessment system: Final report. *Center for Criminal Justice Research, School of Criminal Justice, University of Cincinnati, Cincinnati, OH. Retrieved from http://www. ocjs. ohio. gov/ORAS_FinalReport. pd f*, 2009.

[20] Jiachang Liu, Chudi Zhong, Margo Seltzer, and Cynthia Rudin. Fast sparse classification for generalized linear and additive models. In *Proceedings of Artificial Intelligence and Statistics (AISTATS)*, 2022.

[21] Aurelie Lozano, Grzegorz Swirszcz, and Naoki Abe. Group orthogonal matching pursuit for logistic regression. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 452–460. JMLR Workshop and Conference Proceedings, 2011.

[22] Olvi L Mangasarian, W Nick Street, and William H Wolberg. Breast cancer diagnosis and prognosis via linear programming. *Operations Research*, 43(4):570–577, 1995.

[23] Rui P Moreno, Philipp GH Metnitz, Eduardo Almeida, Barbara Jordan, Peter Bauer, Ricardo Abizanda Campos, Gaetano Iapichino, David Edbrooke, Maurizia Capuzzo, and Jean-

Roger Le Gall. SAPS 3 - from evaluation of the patient to evaluation of the intensive care unit. part 2: Development of a prognostic model for hospital mortality at ICU admission. *Intensive Care Medicine*, 31(10):1345–1355, 2005.

[24] Sérgio Moro, Paulo Cortez, and Paulo Rita. A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems*, 62:22–31, 2014.

[25] Jeffrey Curtis Schlimmer. *Concept acquisition through representational adjustment*. PhD thesis, University of California, Irvine, 1987.

[26] A. J. Six, B. E. Backus, and J. C. Kelder. Chest pain in the emergency room: value of the heart score. *Netherlands Heart Journal*, 16(6):191–196, 2008.

[27] Nataliya Sokolovska, Yann Chevaleyre, Karine Clément, and Jean-Daniel Zucker. The fused lasso penalty for learning interpretable medical scoring systems. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 4504–4511. IEEE, 2017.

[28] Nataliya Sokolovska, Yann Chevaleyre, and Jean-Daniel Zucker. A provable algorithm for learning interpretable scoring systems. In *International Conference on Artificial Intelligence and Statistics*, pages 566–574. PMLR, 2018.

[29] Martin Than, Dylan Flaws, Sharon Sanders, Jenny Doust, Paul Glasziou, Jeffery Kline, Sally Aldous, Richard Troughton, Christopher Reid, and William A Parsonage. Development and validation of the emergency department assessment of chest pain score and 2h accelerated diagnostic protocol. *Emergency Medicine Australasia*, 26(1):34–44, 2014.

[30] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[31] Berk Ustun, Lenard A Adler, Cynthia Rudin, Stephen V Faraone, Thomas J Spencer, Patricia Berglund, Michael J Gruber, and Ronald C Kessler. The world health organization adult attention-deficit / hyperactivity disorder self-report screening scale for dsm-5. *JAMA Psychiatry*, 74(5):520–526, 2017.

[32] Berk Ustun and Cynthia Rudin. Supersparse linear integer models for optimized medical scoring systems. *Machine Learning*, pages 1–43, 2015.

[33] Berk Ustun and Cynthia Rudin. Optimized risk scores. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1125–1134, 2017.

[34] Berk Ustun and Cynthia Rudin. Learning optimized risk scores. *J. Mach. Learn. Res.*, 20:150–1, 2019.

[35] Berk Ustun, Stefano Traca, and Cynthia Rudin. Supersparse linear integer models for predictive scoring systems. In *Workshops at the Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

[36] Caroline Wang, Bin Han, Bhrij Patel, and Cynthia Rudin. In Pursuit of Interpretable, Fair and Accurate Machine Learning for Criminal Recidivism Prediction. *Journal of Quantitative Criminology*, pages 1–63, 2022.

[37] Jiaming Zeng, Berk Ustun, and Cynthia Rudin. Interpretable classification models for recidivism prediction. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 180(3):689–722, 2017.

[38] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See the Limitations section at the end of Section 2
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix C.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] The code and README are included as part of the supplemental material. Data links are included in the Appendix D.1.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We perform 5-fold CV as specified in Section 4. Hyperparameters are already specified (default values) in Algorithm 1 of Section 3.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Section 4.1. Error bars are included for the 5-fold CV.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix D.2
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Appendix D.3 and the References
   (b) Did you mention the license of the assets? [Yes] See Appendix D.3
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] Our code is included as part of the supplementary material.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]