Generalized Inverse Classification

Michael T. Lash[*] Qihang Lin[†] Nick Street[†] Jennifer G. Robinson[‡]
Jeffrey Ohlmann[†]

Abstract

Inverse classification is the process of perturbing an instance in a meaningful way such that it is more likely to conform to a specific class. Historical methods that address such a problem are often framed to leverage only a single classifier, or specific set of classifiers. These works are often accompanied by naive assumptions. In this work we propose generalized inverse classification (GIC), which avoids restricting the classification model that can be used. We incorporate this formulation into a refined framework in which GIC takes place. Under this framework, GIC operates on features that are immediately actionable. Each change incurs an individual cost, either linear or non-linear. Such changes are subjected to occur within a specified level of cumulative change (budget). Furthermore, our framework incorporates the estimation of features that change as a consequence of direct actions taken (indirectly changeable features). To solve such a problem, we propose three real-valued heuristic-based methods and two sensitivity analysis-based comparison methods, each of which is evaluated on two freely available real-world datasets. Our results demonstrate the validity and benefits of our formulation, framework, and methods.

1 Introduction

In typical classification settings, a model is trained and used to make predictions about some event of interest. Depending upon the predictive task, some action may then be taken. In a medical domain, a patient may be monitored more carefully if a prediction yields a high likelihood of some negative outcome. However, in the same setting, we may want to know what actions can be taken to minimize the patient's chances of said adverse event occurring. The process of finding the optimal set of actions, or changes, that can be taken in order to minimize the probability of such events occurring is what we term *inverse classification.*

This example domain further highlights the nature and importance of the problem. Consider, specifically, the problem of mitigating the long-term risk of cardiovascular disease (CVD) of Patient 29 taken from our experiments below. Initially, we use a constructed model to estimate this patient's risk, or probability, of developing CVD, which is found to be 55%. This estimate is based on pertinent factors such as medications, lab measurements (e.g., blood glucose), lifestyle (e.g., diet), and demographics (e.g., age).

Following an initial assessment of risk, we would like to work 'backwards' through our learned model to obtain recommendations that reduce Patient 29's probabililty of CVD. Past methods, however, restrict the set of classifiers that are used to obtain such recommendations, often only affording the use of a single algorithm. Such restrictions are prohibitive in that a particular classifier may have useful properties. These might include high predictive accuracy, such as the random forest used to obtain Patient 29's initial level of risk, or a high degree of explanatory power,which may help the patient better understand why certain recommendations were made. Therefore we propose *generalized inverse classification* (GIC), which permits the use of virtually any classification function, requiring only a simple non-prohibitive assumption (further discussed in Section 3). This is the first contribution of this work.

Our second contribution is to show that the problem can be solved using heuristics. Specifically, we propose three real-valued heuristic-based methods that solve this problem, which we compare to two sensitivity analysis-based baseline methods. We demonstrate the efficacy of these results on two freely available datasets, one of which includes Patient 29, whose risk we lower from 55% to less than 30% (Section 4). Thirdly, we refine an existing inverse classification framework to include non-linear cost-to-change functions, which we then incorporate into our experiments. Section 3 outlines the framework, the *generalized inverse classification problem*, the three heuristic-based methods, and two sensitivity analysis-based methods, while Section 5 concludes the paper.

2 Related Work

Inverse classification is akin to the sub-discipline of sensitivity analysis, which examines the impact of predictive algorithm input on the output. While there are

[*]Department of Computer Science, University of Iowa.
[†]Department of Management Science, University of Iowa.
[‡]Department of Epidemiology, University of Iowa.
{michael-lash, qihang-lin, nick-street, jennifer-g-robinson, jeffrey-ohlmann}@uiowa.edu

many forms of sensitivity analysis[1, 2], local and variable perturbation methods are most similar. Based on this we develop two sensitivity analysis-based methods, related in Section 3, for comparison purposes.

Past works on inverse classification differ with respect to three distinct perspectives: operational data types, algorithmic mechanism, and framework. The **operational data types** which encode the data, on which inverse classification is performed, are either discrete [3, 4, 5], continuous [6, 7, 8, 9], or both [10]. The latter two allow for more fine-grained results leading to greater precision in the recommendations made.

The **algorithmic mechanism** operates on these data types by finding the feasible recommendations that optimize the predicted probability. Such optimization strategies are constructed to be greedy [3, 4, 5, 6] or non-greedy [7, 8, 9, 10].

The **framework** ensures that recommended changes are feasible and implementable. These include: (1) identifying features that can be changed (e.g. age cannot), (2) the difficulty in implementing changes (feature-specific costs) and (3) a restriction on the cumulative change (budget). In [3, 8] there are no constraints imposed. Of those that do impose constrains:

- In [7] constraints are imposed that lead to non-extreme recommendations, but neither (1), (2), or (3) are considered.
- In [6] (2) is imposed, but not (1) or (3).
- In [4, 5] only (1) and (2) are considered.
- A different notion of (1), (2), and (3) are explored in [10] by matching discrete entities to compute features.
- In [9] (1), (2), and (3) are all considered, but does not permit nondifferentiable classifiers.

Real-valued heuristic-based methods are also relevant to this work. These methods include variable neighborhood search (VNS), genetic algorithms [11], and hill-climbing [4, 6]. In this work we elect to focus on genetic algorithms, hill-climbing, and local search, which can be viewed as a simpler form of VNS. As will be shown, by using heuristic-based methods, we can be as general as possible in solving the inverse classification problem.

3 Generalized Inverse Classification

In this section we first briefly discuss GIC. Subsequently, we outline our inverse classification framework. Next, we relate three heuristic-based methods that can be used to solve GIC. Finally, we introduce two sensitivity analysis-based methods that will be compared to our heuristic-based methods.

Under the GIC formulation no assumptions are made about the classification function $f()$ other than $f : \mathbb{R}^p \to \mathbb{R}$. Such a level of generality allow us to obtain optimal solutions for nondifferentiable functions. These functions include popular ensemble techniques such as bagging [12] and boosting [13], as well as C4.5 decision trees. Classifiers such as these are often found to have high predictive power (ensembles) or are more readily interpretable and explainable (e.g., C4.5 decision trees), which is why it is so important methods be developed that incorporate such classifiers.

3.1 Framework Suppose $\{(\mathbf{x}^i, y^i)\}_{i=1,2,\ldots,n}$ is a dataset of n instances where $\mathbf{x}^i \in \mathbb{R}^p$ is a column feature vector of length p and $y^i \in \{-1, 1\}$ is the binary label associated with $\mathbf{x}^i$ for $i = 1, 2, \ldots, n$. Let $f(\mathbf{x})$ be a function that computes the probability of $\mathbf{x}$ being in the positive class (with $y = 1$). Typically, $f(\mathbf{x})$ is based on a certain classification model built on the dataset. Given a new instance, with feature vector $\bar{\mathbf{x}}$, we want to modify some components of $\bar{\mathbf{x}}$, subject to some budget constraints, so that the predicted probability of being positive is minimized.

We further partition the features into three subsets, U, D and I, which represent the sets of unchangeable, directly changeable and indirectly changeable features, respectively. When we optimize the features, we can only determine the value for $\mathbf{x}_D$ and the values of $\mathbf{x}_I$ will depend on $\mathbf{x}_D$ and $\mathbf{x}_U$. Therefore, we model the dependency of $\mathbf{x}_I$ on $\mathbf{x}_D$ and $\mathbf{x}_U$ as $\mathbf{x}_I = H(\mathbf{x}_D, \mathbf{x}_U)$ where the mapping $H : \mathbb{R}^{|D|+|U|} \to \mathbb{R}^{|I|}$ is assumed to be differentiable. Note that the mapping H can be any predictive model constructed using the same training instances. Therefore, we represent $f(\mathbf{x})$ as $f(\mathbf{x}_U, \mathbf{x}_I, \mathbf{x}_D)$ to distinguish these three blocks so that the feature optimization problem can be formulated as

$$\begin{aligned} \min_{\mathbf{x}_D \in \mathbb{R}^{|D|}} \quad & f(\bar{\mathbf{x}}_U, \mathbf{x}_I, \mathbf{x}_D) = f(\bar{\mathbf{x}}_U, \cdots \\ & \qquad H(\mathbf{x}_D, \bar{\mathbf{x}}_U), \mathbf{x}_D) \\ \text{s.t.} \quad & \phi(\mathbf{x}_D - \bar{\mathbf{x}}_D) \leq B \\ & l_i \leq x_i \leq u_i \text{ for } i \in D. \end{aligned} \tag{3.1}$$

Here, we assume the reasonable value of each directly changeable feature in D must be within an interval, denoted by $[l_i, u_i]$ for $i \in D$. If x_i can only be increased (decreased), we can set $l_i = \bar{x}_i$ ($u_i = \bar{x}_i$). In addition, $\phi : \mathbb{R}^{|D|} \to \mathbb{R}$ is a convex cost function that measures the cost for changing $\bar{\mathbf{x}}_D$ to $\mathbf{x}_D$ and B is the total budget we have to support this change. We require $\phi(\mathbf{0}) = 0$.

Here, we provide two examples of $\phi(\mathbf{z})$. The first assumes the cost increases linearly, as $\mathbf{x}$ is deviated from

$\bar{\mathbf{x}}$, which is

$$\phi(\mathbf{z}) = \sum_{i\in D} c_i^+ (z_i)_+ + c_i^- (z_i)_- \tag{3.2}$$

where $(z)_+ = \max\{0, z\}$ and $(z)_- = \max\{0, -z\}$, and c_i^+ and c_i^- denotes the costs for increasing and decreasing the feature x_i by one unit for $i \in D$. If one assumes the costs increase quadratically as $\mathbf{x}$ deviates from $\bar{\mathbf{x}}$, then

$$\phi(\mathbf{z}) = \sum_{i\in D} c_i^+ (z_i)_+^2 + c_i^- (z_i)_-^2 \tag{3.3}$$

Note that the constants c_i^+ and c_i^- in (3.2) and (3.3) can be different. In both cost functions, if decreasing (increasing) x_i is cost-free, we can set $c_i^- = 0$ ($c_i^+ = 0$). In the rest of this paper, we will only focus on the quadratic cost $\phi(\mathbf{z})$ in (3.3).

We define $\mathbf{z} = \mathbf{x}_D - \bar{\mathbf{x}}_D$ in (3.1) and, by changing variables, (3.1) can be equivalently written as

$$\min_{\mathbf{z}\in\Delta_D} \quad g(\mathbf{z}) \tag{3.4}$$

where $g(\mathbf{z}) \equiv f(\bar{\mathbf{x}}_U, H(\bar{\mathbf{x}}_D + \mathbf{z}, \bar{\mathbf{x}}_U), \bar{\mathbf{x}}_D + \mathbf{z})$,

$$\Delta_D \equiv \left\{ \mathbf{z} \in \mathbb{R}^{|D|} \;\middle|\; \begin{array}{c} \phi(\mathbf{z}) \le B, \\ l'_i \le z_i \le u'_i \text{ for } i \in D. \end{array} \right\}, \tag{3.5}$$

$l'_i = l_i - \bar{x}_i$ and $u'_i = u_i - \bar{x}_i$ for $i \in D$. The projection mapping onto the feasible set Δ_D is defined as

$$\mathbf{Proj}_{\Delta_D}(\mathbf{w}) \equiv \arg\min_{\mathbf{z}\in\Delta_D} \frac{1}{2}\|\mathbf{z} - \mathbf{w}\|^2. \tag{3.6}$$

We then define a subroutine for solving (3.6). We first define

$$h_i(w, \lambda) = \begin{cases} \max\{\min\{w/(1 + 2\lambda c_i^+), & \text{if } w \ge 0 \\ \quad \ldots u'_i\}, l'_i\} & \\ \max\{\min\{w/(1 + 2\lambda c_i^-), & \text{if } w < 0 \\ \quad \ldots u'_i\}, l'_i\} & \end{cases} \tag{3.7}$$

for each $i \in D$ and $\lambda \ge 0$. The subroutine is given in Algorithm 1 whose validity can be easily verified by the KKT conditions of (3.4). Note that the bisection search in Algorithm 1 can always succeed because $\sum_{i\in D} c_i^+(h_i(w_i,\lambda))_+^2 + c_i^-(h_i(w_i,\lambda)))_-^2$ monotonically decreases to zero as λ increases to infinity.

Algorithm 1 Projection Mapping **Proj**

```
Input: $\mathbf{w} \in \mathbb{R}^{|D|}$, $\{c_i^+\}_{i\in D}$, $\{c_i^-\}_{i\in D}$, $\{l'_i\}_{i\in D}$ and $\{u'_i\}_{i\in D}$
1: if $\sum_{i\in D} c_i^+(h_i(w_i,0))_+^2 + c_i^-(h_i(w_i,0)))_-^2 \le B$ then
2:     $\lambda \leftarrow 0$
3: else
4:     Apply bisection search to find $\lambda \in (0, +\infty)$ such that
           $\sum_{i\in D} c_i^+(h_i(w_i,\lambda))_+^2 + c_i^-(h_i(w_i,\lambda)))_-^2 = B$
5: end if
6: $z_i \leftarrow h_i(w,\lambda)$ for $i \in D$.
Output: $\mathbf{z}$
```

There are several processes shared among the three algorithms. For simplicity of notations, we assume the features of $\mathbf{x}$ indexed by D are the first $|D|$ features, i.e., $D = \{1, 2, \ldots, |D|\}$. Let $q \sim \mathcal{U}\{D\}$ represent a uniformly distributed random variable over D indicating the indexical position of feature vector $\mathbf{x}_D$ that will be perturbed. Perturbations to feature $\mathbf{x}_q$ occur according to a standard normal distribution

$$b_q \sim \psi(\sigma_q) = \frac{1}{\sigma_q\sqrt{2\pi}} \exp\left(-\frac{b_q^2}{2\sigma_q^2}\right) \tag{3.8}$$

where b_q is random variable representing the perturbation that occurs at indexical position q and σ_q is the standard deviation of feature q obtained from the training data. Let $\mathbf{e}_q \in \mathbb{R}^{|D|}$ be a vector that equals one in the qth coordinate and zero in other places so that the perturbed version of $\mathbf{x}_D$ is denoted by $\mathbf{x}_D + b_q\mathbf{e}_q$. Let $[\mathbf{Z}]_j$ represent the jth row of a matrix $\mathbf{Z}$. Two additional shared parameters include m which we will use to denote the total population size and $MaxIters$ which we will use to denote the number of iterations until an algorithm terminates.

3.2 Heuristic-based methods

We propose three real-valued heuristic-based algorithms to solve the generalized inverse classification problem: hill-climbing + local search (HC+LS), a genetic algorithm (GA), and a genetic algorithm + local search (GA+LS).

3.2.1 Hill-climbing + local search

Our hill-climbing + local search (HC+LS) algorithm is based on that outlined by Mannino and Koushik [6] and is related by Algorithm 3 which calls a local search procedure, outlined in Algorithm 2. In this algorithm, the best current solution, denoted by $\mathbf{x}_D^*$, is perturbed a single feature q at a time in order to find a better solution. There are m single-feature perturbations that occur at each iteration, leading to m perturbed versions of $\mathbf{x}_D^*$, denoted by $\mathbf{x}_D^* + b_{q_j}\mathbf{e}_{q_j}$, for $j = 1, 2, \ldots, m$. We use Algorithm 1 to convert the direction $\mathbf{x}_D^* + b_{q_j}\mathbf{e}_{q_j} - \bar{\mathbf{x}}_D$ into a feasible state and update $\mathbf{x}_D^*$ along the direction that yields the smallest $g(\mathbf{x}_D^* + b_{q_j}\mathbf{e}_{q_j} - \bar{\mathbf{x}}_D)$, where g is defined in (3.4).

Algorithm 2 LS **Local**$(\mathbf{x}_D)$

```
Input: $\mathbf{x}_D \in \mathbb{R}^{|D|}, (\bar{\mathbf{x}}_U, \bar{\mathbf{x}}_D) \in \mathbb{R}^{|U|+|D|}$, $\{c_i^+\}_{i\in D}$, $\{c_i^-\}_{i\in D}$, $\{l_i'\}_{i\in D}$, $\{u_i'\}_{i\in D}$, $B$, and $m$
1: for $j = 1$ to $m$ do
2:     Generate $q_j \sim \mathcal{U}\{D\}$ and $b_{q_j}$ as (3.8);
3:     $\mathbf{z}_j = \mathbf{Proj}_{\Delta_D}(\mathbf{x}_D + b_{q_j}\mathbf{e}_{q_j} - \bar{\mathbf{x}}_D)$;
4: end for
5: if $\min\{g(\mathbf{z}_j); j = 1, \dots, m\} < g(\mathbf{x}_D - \bar{\mathbf{x}}_D)$ then
6:     $\mathbf{x}_D = \mathbf{x}_D + \mathbf{z}_{j'}$ with $j' = \text{argmin}_{j=1,\dots,m} g(\mathbf{z}_j)$
7: end if
Output: $\mathbf{x}_D$
```

We note here that the difference between regular HC and HC+LS is that HC operates on a first improvement basis, whereas HC+LS operates on a best improvement basis.

Algorithm 3 HC+LS **Hill**$(\bar{\mathbf{x}})$

```
Input: $(\bar{\mathbf{x}}_U, \bar{\mathbf{x}}_D) \in \mathbb{R}^{|U|+|D|}$, $\{c_i^+\}_{i\in D}$, $\{c_i^-\}_{i\in D}$, $\{l_i'\}_{i\in D}$, $\{u_i'\}_{i\in D}$, $B$, $m$, and $MaxIters$
1: Initialize $\mathbf{x}_D^* = \bar{\mathbf{x}}_D$;
2: for $iters = 1$ to $MaxIters$ do
3:     $\mathbf{x}_D^* = \mathbf{Local}(\mathbf{x}_D^*)$
4: end for
Output: $\mathbf{x}_D^*$
```

3.2.2 Genetic algorithm

Genetic algorithms are composed of four primary processes: initial population generation, crossover, carryover, and mutation. Our real-valued genetic algorithm (GA) is outlined by Algorithm 4. Prior to outlining such a method, we first relate the four aforementioned components.

At the first iteration of our genetic algorithm, an initial population is generated. For $j = 1, 2, \dots, m$, let $t_j \sim U\{D\}$ be a discrete uniform random variable. We then generate $q_{k,j} \sim \mathcal{U}\{D\}$ and $b_{q_{k,j}}$ as in (3.8) for $k = 1, 2, \dots, t_j$ and define

$$\mathbf{b}_j = \sum_{k=1}^{t_j} b_{q_{k,j}} \mathbf{e}_{q_{k,j}}; j = 1, \dots, m. \tag{3.9}$$

We use $\bar{\mathbf{x}}_D + \mathbf{Proj}_{\Delta_D}(\mathbf{b}_j)$ for $j = 1, \dots, m$ as the initial population and store them as the rows of a $m \times |D|$ matrix $\mathbf{X}_D^{\text{pop}}$, i.e,

$$[\mathbf{X}_D^{\text{pop}}]_j = \bar{\mathbf{x}}_D + \mathbf{Proj}_{\Delta_D}(\mathbf{b}_j); j = 1, \dots, m. \tag{3.10}$$

We note that $\mathbf{b}$ is updated m times, resulting in unique entries in $\mathbf{X}_D^{\text{pop}}$. Here, we apply (3.6) to ensure that all population chromosomes are feasible.

Following this, a simple procedure $\mathbf{Order}(\mathbf{X}_D^{\text{pop}})$ is called. This orders the rows $\mathbf{X}_D^{\text{pop}}$ by objective function value from smallest to largest. Let $\beta \in (0, 1]$ be a user specified parameter that denotes the proportion of the population that will be bred to produce the offspring for the next generation. We make a copy of the first $\lceil m\beta \rceil$ rows of $\mathbf{X}_D^{\text{pop}}$ and store them as a $\lceil m\beta \rceil \times |D|$ matrix $\mathbf{X}_D^{\text{cross}}$ with $[\mathbf{X}_D^{\text{cross}}]_j = [\mathbf{X}_D^{\text{pop}}]_j$ for $j = 1, 2, ..., \lceil m\beta \rceil$. We then randomly shuffle the rows of $\mathbf{X}_D^{\text{cross}}$ using a procedure $\mathbf{Shuffle}(\mathbf{X}_D^{\text{cross}})$.

Let $1 - \gamma$ be the proportion of the population that should be composed of children (γ being the proportion of the population that will be carried over, discussed shortly). We construct a vector of indices $\vartheta \in \mathbb{N}^{\lceil m\beta \rceil}$ as

$$\vartheta = \begin{cases} (1, 2, \dots, \lceil m(1-\gamma) \rceil) & \text{if } (1-\gamma) \le \beta \\ (1, 2, \dots, \lceil m\beta \rceil, & \text{otherwise} \\ \quad r_{\lceil m\beta \rceil + 1}, r_{\lceil m\beta \rceil + 2}, \dots, r_{\lceil m(1-\gamma) \rceil}) & \end{cases}, \tag{3.11}$$

where $r_k \sim \mathcal{U}\{1, 2, \dots, \lceil m\beta \rceil\}$ is a uniformly distributed random index for $k = \lceil m\beta \rceil + 1, \lceil m\beta \rceil + 2, \dots, \lceil m(1-\gamma) \rceil$.

Selected chromosomes are bred using single-point crossover outlined in Michalewicz, 2013 [11], adapted to maintain feasibility via our projection operator. Without loss of generality, we assume $\lceil m(1-\gamma) \rceil$ is an even number and $\lceil m(1-\gamma) \rceil = 2K$ for some integer K. For $k = 1, 2, \dots, K$, we use the vector ϑ defined in (3.11) to create children from the matrix of parent chromosomes $\mathbf{X}_D^{\text{cross}}$ by doing

$$\begin{aligned} \mathbf{x}_D^{2k-1} &= \mathbf{Mut}(([\mathbf{X}_D^{\text{cross}}]^1_{\vartheta_{2k-1}}, ..., [\mathbf{X}_D^{\text{cross}}]^{q_k - 1}_{\vartheta_{2k-1}}, [\mathbf{X}_D^{\text{cross}}]^{q_k}_{\vartheta_{2k}}, \\ &\qquad ..., [\mathbf{X}_D^{\text{cross}}]^{|D|}_{\vartheta_{2k}})) \\ \mathbf{x}_D^{2k} &= \mathbf{Mut}(([\mathbf{X}_D^{\text{cross}}]^1_{\vartheta_{2k}}, ..., [\mathbf{X}_D^{\text{cross}}]^{q_k - 1}_{\vartheta_{2k}}, [\mathbf{X}_D^{\text{cross}}]^{q_k}_{\vartheta_{2k-1}}, \\ &\qquad ..., [\mathbf{X}_D^{\text{cross}}]^{|D|}_{\vartheta_{2k-1}})) \end{aligned} \tag{3.12}$$

where $[\mathbf{Z}]^i_j$ represent the entry in the jth row and the ith column of a matrix $\mathbf{Z}$, $q_k \sim \mathcal{U}\{D\}$ is a selected crossover point, generated for a pair of parents, i.e., the rows ϑ_{2k-1} and ϑ_{2k} of $\mathbf{X}_D^{\text{cross}}$ for $k = 1, 2, \dots, K$. $\mathbf{Mut}(\mathbf{x}_D)$ is the mutation operator defined as

$$\mathbf{Mut}(\mathbf{x}_D) = \mathbf{x}_D + I_v b_q \mathbf{e}_q, \tag{3.13}$$

where $q \sim \mathcal{U}\{D\}$, b_q is generated as (3.8), I_v is a binary random variable which equals zero and one with a probability of v and $1 - v$ respectively (v is a user-specified parameter representing the probability of mutation occurring to allele i by amount b_i defined in (3.8)). Subsequently, the children are ensured feasible by

$$\begin{aligned} \mathbf{x}_D^{2k-1\prime} &= \mathbf{Proj}_{\Delta_D}(\mathbf{x}_D^{2k-1} - \bar{\mathbf{x}}_D) + \bar{\mathbf{x}}_D \\ \mathbf{x}_D^{2k\prime} &= \mathbf{Proj}_{\Delta_D}(\mathbf{x}_D^{2k} - \bar{\mathbf{x}}_D) + \bar{\mathbf{x}}_D. \end{aligned} \tag{3.14}$$

These feasible children are then stored as rows of a $\lceil m(1-\gamma)\rceil \times |D|$ matrix $\mathbf{X}_D^{\text{child}}$.

The carryover procedure uses roulette wheel selection [11] to select chromosomes from the current generation that will survive to the next. Chromosomes that have larger (i.e., better) fitness values (where fitness denotes solution quality) have a higher likelihood of surviving to the next generation.

First, we create an inverted solution vector $\mathcal{P} \in \mathbb{R}^m$. These inverted solutions are transformed from $g(\mathbf{x}_D - \bar{\mathbf{x}}_D)$ by function $\Lambda : \mathbb{R}^{|D|} \to \mathbb{R}$ defined as

$$\Lambda(\mathbf{x}_D) = \omega - g(\mathbf{x}_D - \bar{\mathbf{x}}_D) \tag{3.15}$$

where ω is the worst-case solution possible and is assumed to be positive. Using $\Lambda(\cdot)$ we construct a vector of selection probability

(3.16)
$$\mathscr{P} = \left(\frac{\Lambda([\mathbf{X}_D^{\text{pop}}]_1)}{\sum_{j=1}^m \Lambda([\mathbf{X}_D^{\text{pop}}]_j)}, \ldots, \frac{\Lambda([\mathbf{X}_D^{\text{pop}}]_m)}{\sum_{j=1}^m \Lambda([\mathbf{X}_D^{\text{pop}}]_j)} \right).$$

Intuitively, higher quality solutions have larger probability to be selected, since we have already ordered $\mathbf{X}_D^{\text{pop}}$ by $g(\mathbf{x}_D - \bar{\mathbf{x}}_D)$ so that $\mathscr{P}_j > \mathscr{P}_{j+1}$.

Using (3.16) we select chromosomes from the population matrix $\mathbf{X}_D^{\text{pop}}$ to be carried over to the next generation by

$$\mathbf{x}_D^{\text{carry},k} = [\mathbf{X}_D^{\text{pop}}]_j \text{ with a probability of } \mathscr{P}_j \tag{3.17}$$

for $k = 1, 2, \ldots, \lfloor m\gamma \rfloor$. Those selected children are stored as the $\lfloor m\gamma \rfloor \times |D|$ carryover matrix $\mathbf{X}_D^{\text{carry}}$.

Using (3.10),(3.12), (3.13) and (3.17) we construct our GA as outlined by Algorithm 4. The procedure begins by initializing the best solution to the unperturbed chromosome $\bar{\mathbf{x}}_D$. The algorithm then begins iteration, executing $MaxIters$ times. If it is the first iteration, the initial population is generated. The current population $\mathbf{X}_D^{\text{pop}}$ is then evaluated and if a better solution is found, it is updated. Following this, a simple procedure **Order**($\mathbf{X}_D^{\text{pop}}$) is called. This orders $\mathbf{X}_D^{\text{pop}}$ by objective function value from smallest to largest. Crossover points are then selected in an elitist fashion from this ordered matrix of chromosomes. Selected chromosomes are then randomly shuffled, using procedure **Shuffle**($\mathbf{X}_D^{\text{cross}}$), before crossover is applied to create the offspring chromosomes. Next, the carryover chromosomes are selected. Finally, the children and carryover chromosome matrices are concatenated to form the population for the next generation.

3.2.3 Genetic algorithm + local search The third method is a genetic algorithm + local search

Algorithm 4 GA **Gen**($\bar{\mathbf{x}}$)

```
Input: $(\bar{\mathbf{x}}_U, \bar{\mathbf{x}}_D) \in \mathbb{R}^{|U|+|D|}$, $\{c_i^+\}_{i\in D}$, $\{c_i^-\}_{i\in D}$, $\{l_i'\}_{i\in D}$, $\{u_i'\}_{i\in D}$, $B$, $m$, $MaxIters$, $\alpha$, $\beta$, $\gamma$, $v$ and $\omega$
1: Initialize $\mathbf{x}_D^* = \bar{\mathbf{x}}_D$
2: for $iters = 1$ to $MaxIters$ do
3:   if $iters == 1$ then
4:     Generate an initial population $\mathbf{X}_D^{\text{pop}}$ using (3.10)
5:   end if
6:   if $\min\{g([\mathbf{X}_D^{\text{pop}}]_j - \bar{\mathbf{x}}_D); j = 1, \ldots, m\} < g(\mathbf{x}_D^* - \bar{\mathbf{x}}_D)$ then
7:     $\mathbf{x}_D^* = [\mathbf{X}_D^{\text{pop}}]_{j'}$ with $j' = \text{argmin}_{j=1,\ldots,m} g([\mathbf{X}_D^{\text{pop}}]_j - \bar{\mathbf{x}}_D)$
8:   end if
9:   Order($\mathbf{X}_D^{\text{pop}}$)
10:  $[\mathbf{X}_D^{\text{cross}}]_j = [\mathbf{X}_D^{\text{pop}}]_j$, $j = 1, 2, ..., \lceil m\beta \rceil$
11:  Shuffle($\mathbf{X}_D^{\text{cross}}$)
12:  Obtain $\mathbf{X}_D^{\text{child}}$ from $\mathbf{X}_D^{\text{cross}}$ according to (3.12)
13:  Obtain $\mathbf{X}_D^{\text{carry}}$ from $\mathbf{X}_D^{\text{pop}}$ by (3.17)
14:  Create the new population $\mathbf{X}_D^{\text{pop}} = \begin{pmatrix} \mathbf{X}_D^{\text{child}} \\ \mathbf{X}_D^{\text{carry}} \end{pmatrix}$
15: end for
Output: $\mathbf{x}_D^*$
```

(GA+LS). It is related by Algorithm 5. There are a few important distinctions between the original GA and that with local search applied. First, we reformulate the crossover procedure outlined by (3.12) to be

(3.18)
$$\begin{aligned}\mathbf{x}_D^{2k-1} &= ([\mathbf{X}_D^{\text{cross}}]_{\vartheta_{2k-1}}^1, ..., [\mathbf{X}_D^{\text{cross}}]_{\vartheta_{2k-1}}^{q_k-1}, [\mathbf{X}_D^{\text{cross}}]_{\vartheta_{2k}}^{q_k}, \\ &\qquad ..., [\mathbf{X}_D^{\text{cross}}]_{\vartheta_{2k}}^{|D|}) \\ \mathbf{x}_D^{2k} &= ([\mathbf{X}_D^{\text{cross}}]_{\vartheta_{2k}}^1, ..., [\mathbf{X}_D^{\text{cross}}]_{\vartheta_{2k}}^{q_k-1}, [\mathbf{X}_D^{\text{cross}}]_{\vartheta_{2k-1}}^{q_k}, \\ &\qquad ..., [\mathbf{X}_D^{\text{cross}}]_{\vartheta_{2k-1}}^{|D|})\end{aligned}$$

for $k = 1, 2, \ldots, K$. The reader will note that here the mutation procedure is not applied.

Second, we incorporate the use of the local search (LS) procedure previously outlined in Algorithm 2. Here, we set parameter m equal to ξ, which dictates the extent of the search.

GA+LS is outlined by Algorithm 5. The differences between this method and the original GA are outlined in blue. At line 14 the LS procedure is applied to each of the non-mutated children. The best solution obtained from LS is the child chromosome that is kept for the next generation.

3.3 Sensitivity analysis-based methods As discussed in Section 2, sensitivity analysis is closely related

Algorithm 5 GA+LS **GenLoc**($\bar{\mathbf{x}}$)

```
Input: $(\bar{\mathbf{x}}_U, \bar{\mathbf{x}}_D) \in \mathbb{R}^{|U|+|D|}$, $\{c_i^+\}_{i\in D}$, $\{c_i^-\}_{i\in D}$, $\{l_i'\}_{i\in D}$, $\{u_i'\}_{i\in D}$, $B$, $m$, $MaxIters$, $\alpha$, $\beta$, $\gamma$, $v$ and $\xi$
 1: Initialize $\mathbf{x}_D^* = \bar{\mathbf{x}}_D$
 2: for $iters = 1$ to $MaxIters$ do
 3:   if $iters == 1$ then
 4:     Generate an initial population $\mathbf{X}_D^{\text{pop}}$ using (3.10)
 5:   end if
 6:   if $\min\{g([\mathbf{X}_D^{\text{pop}}]_j - \bar{\mathbf{x}}_D); j = 1,\ldots,m\} < g(\mathbf{x}_D^* - \bar{\mathbf{x}}_D)$ then
 7:     $\mathbf{x}_D^* = [\mathbf{X}_D^{\text{pop}}]_{j'}$ with $j' = \text{argmin}_{j=1,\ldots,m} g([\mathbf{X}_D^{\text{pop}}]_j - \bar{\mathbf{x}}_D)$
 8:   end if
 9:   Order($\mathbf{X}_D^{\text{pop}}$)
10:   $[\mathbf{X}_D^{\text{cross}}]_j = [\mathbf{X}_D^{\text{pop}}]_j$, $j = 1, 2, ..., \lceil m\beta \rceil$
11:   Shuffle($\mathbf{X}_D^{\text{cross}}$)
12:   Obtain $\mathbf{X}_D^{\text{child}}$ from $\mathbf{X}_D^{\text{cross}}$ by (3.18)
13:   $[\mathbf{X}_D^{\text{child}\prime}]_j = \textbf{Local}([\mathbf{X}_D^{\text{child}}]_j), j = 1, ..., \lceil m(1-\gamma) \rceil$
14:   Obtain $\mathbf{X}_D^{\text{carry}}$ from $\mathbf{X}_D^{\text{pop}}$ by (3.17)
15:   Create the new population $\mathbf{X}_D^{\text{pop}} = \begin{pmatrix} \mathbf{X}_D^{\text{child}\prime} \\ \mathbf{X}_D^{\text{carry}} \end{pmatrix}$
16: end for
Output: $\mathbf{x}_D^*$
```

to inverse classification. Therefore, we propose two sensitivity analysis-based algorithms that serve as baselines against which the heuristic-based methods can be compared against. To our knowledge, no past methods addressing this problem have been proposed. Therefore, we craft these ourselves, and believe that they represent a reasonable initial attempt at a solution. Such methods can be viewed as a combination of local and variable perturbation methods of sensitivity analysis.

We refer to the first sensitivity analysis-based method as Local Variable Perturbation–Best Improvement (LVP-BI). This method calls for perturbing a single feature $i \in D$ to the extent of feasibility given by $\min\{\frac{B}{c_i}, u_i'\}$. The single feature perturbation having the greatest objective function improvement is the one that is accepted. If some budget remains following this perturbation, subsequent perturbations are performed (e.g., double feature, triple feature, etc. perturbations).

Our second method, which we refer to as Local Variable Perturbation–First Improvement (LVP-FI), is very similar to that of LVP–BI. Instead of accepting the best perturbation over all $i \in D$ it accepts the first perturbation that leads to a better objective function value, where i is selected at random.

4 Experiments

In this section we first outline our choices regarding the parameters of the inverse classification framework and then apply our methods to two freely available datasets. Our experiments will evaluate the five methods by examining the average likelihood of test instances conforming to a non-ideal class over varying budget constraints.. First, we will explore the capability of each algorithm in reducing the likelihood of test instances conforming to a non-ideal class. Additionally, we will examine the perturbations made to an individual test instance, selected at random, by the top performing algorithm. We wish to emphasize that practical and real-world use of these methods should be undertaken with experts in the domain of use. We further emphasize that inverse classification puts the individual at the center of the process and optimizes over his/her current values. Therefore, if an individual so choses, he/she can adjust expert-specified costs according to their own outlook on what may be more or less difficult to change.

4.1 Experiment Parameters and Evaluation

There are three choices that need to be made regarding the established inverse classification framework: the learning algorithm, the indirectly changeable feature estimator, and the method we will use to set the lower- and upper-bounds that directly changeable features can take.

4.1.1 Objective Function We selected the Random forest classifier [14] to evaluate each of the five methods. We chose this as it is (a) an ensemble classifier and (b) composed of weak-learner decision trees. Both (a) and (b) are separately non-differentiable, and comprehensively help highlight the need for the GIC formulation we have proposed. The returned objective function value will be the proportion of decision trees in the ensemble voting in favor of the class to be minimized. As such $f(\cdot) \in [0, 1]$. We therefore can also parameterize $\omega = 1$, the worst-case objective function value in (3.15).

4.1.2 Indirectly Changeable Feature Estimation The inverse classification framework allows for any smooth model $H(\cdot)$ to be selected to estimate the indirectly changeable features. We elect to use a kernel regression method [15, 16]

$$(4.19) \quad \mathbf{x}_I = \frac{\sum_{i=1}^n k([\mathbf{x}_D^i, \mathbf{x}_U^i], [\mathbf{x}_D, \mathbf{x}_U])\mathbf{x}_I^i}{\sum_{i=1}^n k([\mathbf{x}_D^i, \mathbf{x}_U^i], [\mathbf{x}_D, \mathbf{x}_U])},$$

where $\mathbf{x}^i$ is a training instance and $k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\sigma^2}\right)$ is the Gaussian kernel. We elect to use this function and corresponding Gaussian kernel for its similarity-based estimation properties. We cross-validate this model on each of the indirectly changeable features in order to learn the best σ for each.

4.1.3 Bound-setting method and cost function Lash et al., [9] outline two methods of specifying lower- and upper-bounds for the directly changeable features. Each result in different algorithmic behavior. In our experiments we use the Hard-line bound-setting method. Under this method we specify, for feature $i \in D$, the upper- and lower- bounds such that i can only either increase or decrease. If feature i should increase from its current value of $\mathbf{x}_i$ we set $l_i = \bar{\mathbf{x}}_i$. If feature i should decrease from it's current value of $\mathbf{x}_i$ we set $u_i = \bar{\mathbf{x}}_i$. This allows us to maintain more control over what we know and believe to be the beneficial direction of feature movement. We do note, however, that under different circumstances (e.g., uncertainty) it may be beneficial to allow the optimization to learn the most beneficial direction of feature movement.

In this set of experiments we elect to explore the effects of non-linear costs, related by 3.3. We elect to do so as non-linear costs, to the best of our knowledge, have not been explored in past works.

4.1.4 Evaluating Recommendations To evaluate the success of the inverse classification we use an established procedure originally outlined in [4] and refined in [9]. This process entails initially splitting a dataset $\mathbb{D}$ randomly into two equal parts $\mathbb{D}^{\text{train}}$, where the first is used for training the random forest model upon which inverse classification will take place. The second set $\mathbb{D}^{\text{ho}}$, is the held-out set of data to which inverse classification will be applied.

$\mathbb{D}^{\text{ho}}$ is further partitioned into k distinct subsets which we can denote $\mathbb{D}_i^{\text{ho}}$, $i = 1, ..., k$ ($k = 10$ in our experiments). The process of evaluation entails that we perform inverse classification on $\bar{\mathbf{x}} \in \mathbb{D}_i^{\text{ho}}$ and use $\{\mathbb{D}^{\text{ho}}\} \setminus \{\mathbb{D}_i^{\text{ho}}\}$ to train a separate model to evaluate the success of the inverse classification. Such a process ensures that no information used to perform the inverse classification and obtain recommendations is used in evaluating how successful the process actually was. Additionally, this helps ensure that the classifier used to make the recommendations has not overfit the data.

4.2 Student Performance: Grade-improving recommendations Our first set of experiments are conducted on a UCI Machine Learning Repository dataset called Student Performance [17]. This dataset consists of Portuguese students enrolled in two different classes: a math class and a Portuguese language class. Represented as two disjoint, but overlapping datasets, we elect to use the Portuguese language set as it has the larger number of instances ($n = 649$).

Param	HC + LS	GA	GA + LS
MI	300	300	150
m	15	15	15
α	NA	.30	NA
β	NA	.40	.40
γ	NA	.10	.10
ξ	NA	NA	6
Big O	$MI * m$	$MI * m$	$MI * m * \xi$

Table 1: Benchmark experiment parameters

4.2.1 Data Description Each individual in Student Performance is initially represented by 45 features, including a unique identifier (discarded) and class variable y, which we define to be whether or not a student's final grade was above a C ($y = 0$) or, conversely, less than or equal to a C ($y = 1$). Our GIC methods will attempt to reduce the likelihood of earning a grade of C or worse. We discard the two intermediary grade reports to reflect a long-term goal of earning a higher grade overall and make the problem more realistic. The full set of features and corresponding parameters can be viewed in the Supplemental Material.

The parameters set for the three heuristic-based methods in these experiments are related by Table 1, as is the computational complexity. We arrived at these after a brief exploration of the parameter space, selecting values that were comparable so that performance could be equivalently compared. For GA+LS, we kept $MaxIters$ (abbreviated MI) lower because of the added complexity of the ξ parameter.

4.2.2 Results We first examine the success of reducing the average predicted probability for each of the five methods. These results are reported in Figure 1. We report each over 15 increasing budgetary constraints. Additionally, we include the best result on a randomly selected positively classified instance – Student 57 – obtained using GA.

As we can observe in Figure 1 the two sensitivity analysis-based methods were unsuccessful. The result also shows that the three heuristic-based methods are comparable, with GA and GA+LS declining slightly faster than HC+LS. We include more detailed information about the performance of each method in the Supplemental Materials.

We report the changes made to "Student 57" in Figure 2 for the method most successful in reducing their predicted probability: GA. We report this so that

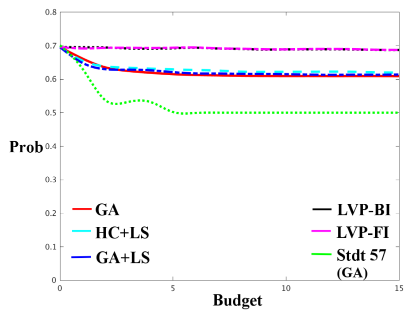

Figure 1: The average predicted probability vs budget for three heuristic- and two sensitivity analysis-based methods. GA result for Student 57.

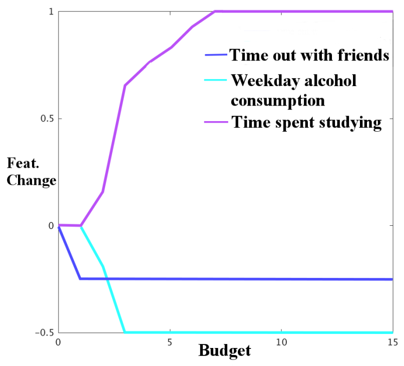

Figure 2: GA recommended changes to Student 57.

the reader may have a better idea of what such recommendations look like. GA recommends the student to increase study time and curb weekday alcohol consumption, as well as to decrease time out with friends.

Cumulatively, the three heuristic methods were, on average, able to reduce the probability from approximately 70% to 62% at a budget level of three. Individually, the best performing method was able to reduce Studet 57's probability from 70% to 50% at a budget level of five.

4.3 Cardiovascular disease mitigating lifestyle recommendations

Our second set of experiments is conducted on a real-world patient dataset, derived from the ARIC study. These data are freely available upon request from BioLINCC.

4.3.1 Data Description These data represent patients, for whom we have known cardiovascular disease (CVD) outcomes over a 10 year period. There are 110 defined features for each patient. Patients who, during the course of the 10 year period have probable myocardial infarction (MI), definite MI, suspect MI, definite fatal coronary heart disease (CHD), possible fatal CHD,

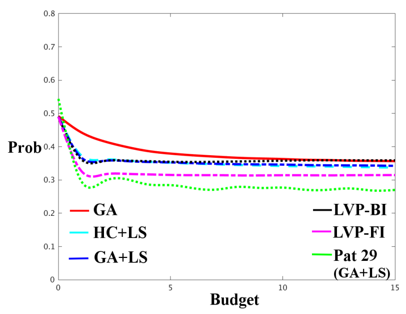

Figure 3: The average predicted probability vs budget for the three heuristic- and two sensitivity analysis-based methods. GA+LS result for Patient 29.

or stroke have $y = 1$ and $y = 0$ otherwise. Patients who had a pre-existing CVD event are excluded from our dataset, giving us a total of $n = 12007$ patients. This set of experiments is meant to more closely reflect a real-world scenario and, as such, is guided by a CVD specialist. The full list of features, their feature designation (e.g., changeable) and parameters (e.g., cost) can be viewed in the Supplemental Materials.

After a brief exploration of the parameter space, we arrived at the same set of parameters as in the previous experiment (Student Performance). We omit the duplicate table and refer to Table 1. Additionally, because of the size of the testing dataset, and the computational complexity associated with the heuristic-based methods, we elected to test on a subset of data. We used all 587 positive test instances and another 587 randomly selected negative test instances, giving us a final evaluative test set size of 1164. Evaluation models were constructed using the full set of data by the procedure outlined in Section 4.1.4.

4.3.2 Results We first examine the success of reducing the average predicted probability using the five outlined methods. These results are reported in Figure 3. We report each over 15 increasing budgetary constraints. Additionally, we include the best result on a randomly selected positively classified instance – Patient 29 – obtained using GA+LS.

The results obtained for the heuristic-based methods are similar to those of Student Performance. There is a striking difference, however, between those and the sensitivity based-method results here. We observe that LVP-FI outperforms all other methods, while LVP-BI is comparable to GA and GA+LS. HC+LS performs the worst. The stark difference in performance of LVP-FI and LVP-BI on this dataset vs. that of student performance may suggest that there are instances in which it is advantageous to use sensitivity analysis-based meth-

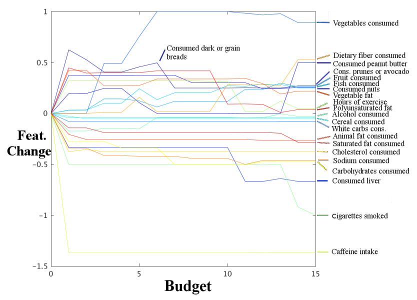

Figure 4: GA+LS recommended changes to Patient 29.

ods over those that are heuristic-based, and vice-versa. We leave such an analysis for future work.

We report the changes made to "Patient 29" in Figure 4 for the method most successful in reducing the patient's predicted probability: GA+LS. Here we observe that the number of feature changes recommended are quite numerous: there are 22 of them. This suggests that it may be beneficial to include sparsity constraints.

Cumulatively, these results show that, on average, risk can be taken from approximately 50% to 30-35%, depending upon the method, at a budgetary level of two. At the individual level, using the best method, Patient 29's risk can be lowered from 55% to less than 30%, also a at a budgetary level of two.

5 Conclusions

In this work we propose and solve *generalized inverse classification* by working backward through the previously un-navigable random forest classifier using five proposed algorithms that we incorporated into a framework, updated to account for non-linear costs, that leads to realistic recommendations. Future work is needed to analyze instances in which one method may outperform another, the performance of other classifiers and constraints limiting the number of features that are changed.

References

[1] S. S. Isukapalli, *Uncertainty Analysis of Transport-transformation Models*. PhD thesis, Citeseer, 1999.

[2] J. Yao, "Sensitivity analysis for data mining," in *Fuzzy Information Processing Society, 2003. NAFIPS 2003. 22nd International Conference of the North American*, pp. 272–277, July 2003.

[3] C. C. Aggarwal, C. Chen, and J. Han, "The inverse classification problem," *Journal of Computer Science and Technology*, vol. 25, no. May, pp. 458–468, 2010.

[4] C. L. Chi, W. N. Street, J. G. Robinson, and M. A. Crawford, "Individualized patient-centered lifestyle recommendations: An expert system for communicating patient specific cardiovascular risk information and prioritizing lifestyle options," *Journal of Biomedical Informatics*, vol. 45, no. 6, pp. 1164–1174, 2012.

[5] C. Yang, W. N. Street, and J. G. Robinson, "10-year CVD risk prediction and minimization via inverse classification," in *Proceedings of the 2nd ACM SIGHIT symposium on International health informatics - IHI '12*, pp. 603–610, 2012.

[6] M. V. Mannino and M. V. Koushik, "The cost minimizing inverse classification problem : A algorithm approach," *Decision Support Systems*, vol. 29, no. 3, pp. 283–300, 2000.

[7] D. Barbella, S. Benzaid, J. Christensen, B. Jackson, X. V. Qin, and D. Musicant, "Understanding support vector machine classifications via a recommender system-like approach," in *Proceedings of the International Conference on Data Mining*, pp. 305–11, 2009.

[8] P. C. Pendharkar, "A potential use of data envelopment analysis for the inverse classification problem," *Omega*, vol. 30, no. 3, pp. 243–248, 2002.

[9] M. T. Lash, Q. Lin, W. N. Street, and J. G. Robinson, "A budget-constrained inverse classification framework for smooth classifiers," *arXiv preprint; arxiv:1605.09068*, 2016.

[10] M. T. Lash and K. Zhao, "Early predictions of movie success: The who, what, and when of profitability," *Journal of Management Information Systems*, vol. 33, no. 3, pp. 874–903, 2016.

[11] Z. Michalewicz, *Genetic algorithms+ data structures= evolution programs*. Springer Science & Business Media, 2013.

[12] L. Breiman, "Bagging Predictors," *Machine Learning*, vol. 24, pp. 123–140, 1996.

[13] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," *Thirteenth International Conference on Machine Learning*, pp. 148–156, 1996.

[14] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[15] E. a. Nadaraya, "On estimating regression," *Theory of Probability & Its Applications*, vol. 9, no. 1, pp. 141–142, 1964.

[16] G. S. Watson, "Smooth regression analysis," *The Indian Journal of Statistics, Series A*, vol. 26, no. 4, pp. 359–372, 1964.

[17] P. Cortez and A. M. G. Silva, "Using data mining to predict secondary school student performance," in *Proceedings of 5th Annual Future Business Technology Conference*, EUROSIS, 2008.