Flexibly Mining Better Subgroups

Hoang-Vu Nguyen[○] Jilles Vreeken[○]

Abstract

In subgroup discovery, perhaps the most crucial task is to discover high-quality one-dimensional subgroups, and refinements of these. For nominal attributes, finding such binary features is relatively straightforward, as we can consider individual attribute values as such. For numerical attributes, the task is more challenging as individual numeric values are not reliable statistics. Instead, we can consider combinations of adjacent values, i.e. bins. Existing binning strategies, however, are not tailored for subgroup discovery. That is, the bins they construct do not necessarily facilitate the discovery of high-quality subgroups, therewith potentially degrading the mining result.

To address this, we introduce FLEXI. In short, we propose to use an optimal binning strategy for finding high-quality binary features for *both* numeric and ordinal attributes. We instantiate FLEXI with various quality measures and show how to achieve efficiency accordingly. Experiments on both synthetic and real-world data sets show that FLEXI outperforms state of the art with up to 25 times improvement in subgroup quality.

1 Introduction

Subgroup discovery aims at finding subsets of the data, called subgroups, that are statistically unusual with respect to the distribution of target variable(s) [5, 7, 23]. As such, it is a branch of supervised with applications in many areas, including spatial analysis [7], marketing campaign management [9], and health care [13].

A crucial part of the subgroup discovery process is the extraction of high quality binary features out of existing attributes. By binary features, we mean features whose values are either true or false. For instance, possible binary features of *Age* attribute are *Age* ≥ 50 and $20 \leq$ *Age* ≤ 30. These features constitute one-dimensional subgroups or one-dimensional refinements of subgroups, which are used by many existing search schemes (e.g. beam search) [2,6,20].

Deriving such features is straightforward for *nominal* attributes, e.g. their individual values can be used directly as binary features [12]. This also is the case for *ordinal* attributes if one is to treat them as nominal; the downside is that their ordinal nature is not used. The task, however, becomes more challenging for *numerical* (e.g. real-valued) attributes. For such an attribute, binary features formed by single values statistically and empirically are not reliable; they tend to have low generality. Thus, one usually switches to combinations of adjacent values, i.e. *bins*.

To this end, we observe three challenges that are in the way of finding high quality bins, i.e. binary features, for subgroup discovery. First, we need a problem formulation tailored to this purpose. Commonly used binning strategies such as equal-width and equal-frequency are oblivious of subgroup quality, impacting quality of the final output. Second, we should not place any restriction on the target; be it univariate or multivariate; nominal, ordinal, or numeric. Existing solutions also do not address this issue. For instance, SD [3] used in [5] requires that the target is univariate and nominal. Likewise, ROC [12] requires a univariate target. Third, the solution should scale well in order to handle large data sets. This means that we need new methods that can handle the first two issues and are efficient.

In this paper, we aim at tackling these challenges. We do so by proposing FLEXI, for flexible subgroup discovery. In short, FLEXI formulates the search of binary features per numeric/ordinal attribute as identifying the features with *maximal average quality*. This formulation meets the generality requirement since it does not make any assumption on the target. We instantiate FLEXI with various quality measures and show how to achieve efficiency accordingly. Extensive experiments on large real-world data sets show that FLEXI outperforms state of the art, providing up to 25 times improvement in terms of subgroup quality. Furthermore, FLEXI scales very well on large data sets.

The road map of this paper is as follows. In Section 2, we present preliminaries. In Section 3, we introduce FLEXI. In Sections 4 and 5, we consider different quality measures and explain how to efficiently optimise their binnings. In Section 6, we review related work. We present the experimental results in Section 7. In Section 8 we round up with a discussion and conclude the paper in Section 9. For readability and succinctness, we postpone the proofs for the theorems to the Appendix.

2 Preliminaries

Let us consider a data set $\mathbf{D}$ of size m with attributes $\mathbf{A} = \{A_1, \ldots, A_n\}$, and targets $\mathbf{T} = \{T_1, \ldots, T_d\}$. Each attribute $A \in \mathbf{A}$ can be nominal, ordinal, or numeric. When A is either nominal or ordinal, its domain $dom(A)$ is the set of its possible values. Each target $T \in \mathbf{T}$ can be either numeric or ordinal. If $T_i \in \mathbf{T}$ is numeric, we assume that $dom(T_i) = [v_i, V_i]$. Otherwise, $dom(T_i)$ is the set of possible values of T_i. The probability function of $\mathbf{T}$ on $\mathbf{D}$ is

[○] Max Planck Institute for Informatics and Saarland University, Saarbrücken, Germany. {hnguyen, jilles}@mpi-inf.mpg.de

denoted as $p(\mathbf{T})$.

A subgroup S on $\mathbf{D}$ has the form $b_1 \wedge \ldots \wedge b_k$ ($k \in [1, n]$) where (1) each b_j ($j \in [1, k]$) is a condition imposed on some attribute $A \in \mathbf{A}$ and (2) no two conditions share the same attribute. For each numeric attribute A, each of its conditions b has the form $A \in (l, u]$ where $l \in \mathbb{R} \cup \{-\infty\}$, $u \in \mathbb{R} \cup \{+\infty\}$, and $l < u$. If A is ordinal, b also has the form $A \in (l, u]$ where $l, u \in dom(A)$ and $l < u$. If A is categoric, b instead has the form $A = a$ where $a \in dom(A)$.

We let $\mathcal{S}$ be the set of all subgroups on $\mathbf{D}$. The subset of $\mathbf{D}$ covered by S is denoted as $\mathbf{D}_S$. We write $p_S(\mathbf{T})$ as the probability function of $\mathbf{T}$ on $\mathbf{D}_S$. Overall, subgroup discovery is concerned with detecting S having high exception in its target distribution. The level of exception can be expressed through the divergence between $p_S(\mathbf{T})$ and $p(\mathbf{T})$. To achieve high generality – besides the divergence score – the support $s = |\mathbf{D}_S|$ of S should not be too small.

To quantify quality of subgroups, we need quality measure $\phi : \mathcal{S} \to \mathbb{R}$ which assigns a score to each subgroup; the higher the score the better. Typically, ϕ needs to capture both unusualness of target distribution and subgroup support. In this paper, we will study five such quality measures.

3 Mining Binary Features

FLEXI mines binary features for an attribute A of either numeric or ordinal values. When the features serve as one-dimensional subgroups on the first level of the search lattice, the entire realisations of A are used. For one-dimensional refinements, only those realisations covered by the subgroup in consideration are used [13, 20]. For readability, we keep our discussion to the first case. The presentation can straightforwardly be adapted to the second case by switching from the context of the entire data set $\mathbf{D}$ to its subset covered by the subgroup to be refined. Below we also use *bins* and *binary features* interchangeably.

In a nutshell, FLEXI aims at finding binary features with maximal average quality. More specifically, it searches for the binning dsc of A such that the average quality of the bins formed by dsc is maximal. Formally, let $\mathcal{F}$ be the set of possible binnings on A. For each $g \in \mathcal{F}$, we let $\{b_g^1, \ldots, b_g^{|g|}\}$ be the set of bins formed by g where $|g|$ is its number of bins. Each bin $b_g^i = (l_g^i, u_g^i]$ where $l_g^1 = -\infty$, $u_g^{|g|} = +\infty$, and $l_g^i = u_g^{i-1}$ for $i \in [2, |g|]$. FLEXI solves for

$$dsc = \arg\max_{g \in \mathcal{F}} \frac{1}{|g|} \sum_{i=1}^{|g|} \phi(b_g^i) \quad .$$

Another alternative would be to consider the *sum* of subgroup quality. We discuss this option shortly afterwards. Now, we present FLEXI, our solution to the above problem.

At first, we note that $|\mathcal{F}| = O(2^m)$, i.e. the search space is exponential in m making an exhaustive enumeration infeasible. Fortunately, it is structured. In particular, for each $\lambda \in [1, m]$ let dsc_λ be the optimal solution over all binnings producing λ bins on A. Let $\{b_{dsc}^1, \ldots, b_{dsc}^\lambda\}$ be its bins. We observe that for a fixed value of λ,

$$(3.1) \qquad \sum_{i=1}^{\lambda} \phi(b_{dsc}^i) = \phi(b_{dsc}^\lambda) + \sum_{i=1}^{\lambda-1} \phi(b_{dsc}^i) \quad ,$$

must be maximal. On the other hand, as dsc_λ is optimal w.r.t. λ, $\{b_{dsc}^1, \ldots, b_{dsc}^{\lambda-1}\}$ must be the optimal way to partition values $A \leq l_{dsc}^\lambda$ into $\lambda - 1$ bins. Otherwise, we could have chosen a better way to do so. This consequently would produce another binning for all values of A such that (1) this binning has λ bins and (2) it has a total quality higher than that of dsc_λ. The existence of such a binning contradicts our assumption on dsc_λ.

Hence, for each λ its optimal binning dsc_λ exhibits optimal substructure. This motivates us to build a *dynamic programming* algorithm to solve our problem.

Algorithmic approach. The pseudo-code of FLEXI is given as Algorithm 1. In short, it first forms bins $\{c_1, \ldots, c_\beta\}$ where $\beta \ll m$. Each value $qual[\lambda][i]$ where $\lambda \in [1, \beta]$ and $i \in [\lambda, \beta]$ stands for the total quality of bins obtained by optimally merging (discretising) initial bins $c_1, \ldots, c_i$ into λ bins. $b[\lambda][i]$ contains the resulting bins. Our goal is to efficiently compute $qual[1 \ldots \beta][\beta]$ and $b[1 \ldots \beta][\beta]$. To do so, from Lines 4 to 6 we first compute $qual[1][1 \ldots \beta]$ and $b[1][1 \ldots \beta]$. Then from Lines 7 to 14, we incrementally compute relevant elements of arrays $qual$ and b, using the recursive relation described in Equation (3.1). This is standard dynamic programming. Finally, we return the optimal binning after normalising by the number of bins (Lines 15 and 16). There are two important points to note here.

First, we form initial bins $\{c_1, \ldots, c_\beta\}$ of A. Ideally, one would start with $O(m)$ bins. However, the quality score $\phi(c)$ of bin c is not reliable as well as not meaningful when its support $|c| = O(1)$. Thus, by pre-partitioning A in to β bins, we ensure that there is sufficient data in each bin for a statistically reliable assessment of divergence. Choosing a suitable value for β represents a trade-off between accuracy and efficiency. We empirically study its effect in Section 7.

Second, to ensure efficiency we need an efficient strategy to pre-compute $\phi(\bigcup_{k=j}^{i} c_k)$ (used in Lines 5, 9, and 10) for all $1 \leq j \leq i \leq \beta$. In the next section, we explain how to do this for different quality measures and analyse the complexity of FLEXI accordingly.

Alternative setting. An intuitive alternate formulation of the problem is to maximise the *total* quality of 1-D subgroups formed on A. Formally, we have $dsc = \arg\max_{g \in \mathcal{F}} \sum_{i=1}^{|g|} \phi(b_g^i)$, which can also be solved by dynamic programming (see the online Appendix for details). We compare to this setting in the experiments. We find that our standard setting, maximising the average score, leads to much better results.

Algorithm 1 FLEXI

```
Create initial disjoint bins $\{c_1, \dots, c_\beta\}$ of $A$
Create a double array $qual[1 \dots \beta][1 \dots \beta]$
Create an array $b[1 \dots \beta][1 \dots \beta]$ to store bins
for $i = 1 \to \beta$ do
    $b[1][i] = \bigcup_{k=1}^{i} c_k$ and $qual[1][i] = \phi(b[1][i])$
end for
for $\lambda = 2 \to \beta$ do
    for $i = \lambda \to \beta$ do
        $pos = \arg \max_{1 \le j \le i-1} qual[\lambda - 1][j] + \phi(\bigcup_{k=j+1}^{i} c_k)$
        $qual[\lambda][i] = qual[\lambda - 1][pos] + \phi(\bigcup_{k=pos+1}^{i} c_k)$
        Copy all bins in $b[\lambda - 1][pos]$ to $b[\lambda][i]$
        Add $\bigcup_{k=pos+1}^{i} c_k$ to $b[\lambda][i]$
    end for
end for
$\lambda^* = \arg \max_{1 \le \lambda \le \beta} \frac{1}{\lambda} qual[\lambda][\beta]$
Return $b[\lambda^*][\beta]$
```

	Univariate			Multivariate		
Measure	Nom.	Ord.	Num.	Nom.	Ord.	Num.
WRAcc	✓	✓	✗	✗	✗	✗
z-score	✗	✗	✓	✗	✗	✗
kl	✓	✓	✗	✓	✓	✗
hd	✓	✓	✗	✓	✓	✗
qr	✗	✓	✓	✗	✓	✓

Table 1: Characteristics of quality measures considered in this paper, i.e. their applicability to univariate and multivariate attributes of resp. nominal, ordinal, and numeric values.

4 Quality Measures

FLEXI works with any quality measure. In this section we show how to achieve efficiency, i.e. efficiently pre-compute $\phi(\bigcup_{k=j}^{i} c_k)$ for all $1 \le j \le i \le \beta$, with various measures handling different types of targets. More specifically, we look at five measures: *WRAcc* [4, 5, 20], *z-score* [12], a measure based on Kullback-Leibler divergence (*kl*) [20, 21], a measure based on Hellinger distance (*hd*) [10], and a measure based on quadratic measure of divergence (*qr*) [14]. We show characteristics of all measures in Table 1 and provide their details below. To simplify our analysis, we assume that each bin c_i ($i \in [1, \beta]$) contains $\frac{m}{\beta}$ objects.

4.1 *WRAcc* **measure** Weighted Relative Accuracy (*WRAcc*) is a suited measure when $\mathbf{D}$ has a single binary target T. That is, T assumes either a positive or a negative nominal value. Let m_+ be the number of objects in $\mathbf{D}$ having positive target, i.e. positive label. Consider a subgroup S having $s = |\mathbf{D}_S|$ objects; s_+ of which have positive label.

Algorithm 2 PRE-COMPUTATION WITH *WRAcc*

```
1:  Create an integer array $countPos[1 \dots \beta]$
2:  for $i = 1 \to \beta$ do
3:      $countPos[i]$ = # of objects in $\mathbf{D}_{c_i}$ with positive label
4:      Compute $WRAcc(c_i)$ based on $countPos[i]$
5:  end for
6:  for $i = 2 \to \beta$ do
7:      $\theta = countPos[i]$
8:      for $j = i - 1 \to 1$ do
9:          $\theta = \theta + countPos[j]$
10:         Set # of objects with positive label in $\bigcup_{k=j}^{i} c_k$ to $\theta$
            and hence compute $WRAcc(\bigcup_{k=j}^{i} c_k)$
11:     end for
12: end for
```

The *WRAcc* score of S is defined as

$$WRAcc(S) = \frac{s}{m}\left(\frac{s_+}{s} - \frac{m_+}{m}\right) \quad .$$

Algorithm 2 shows how to pre-compute $WRAcc(\bigcup_{k=j}^{i} c_k)$ for all $1 \le j \le i \le \beta$. The first for loop (Lines 2 to 5) is to count the number of positively labeled objects of c_i ($i \in [1, \beta]$) and hence compute its *WRAcc* score. This step takes $O(m)$. The nested loop (Lines 6 to 12) is to incrementally count the number of positively labeled objects of $\bigcup_{k=j}^{i} c_k$ and hence compute its *WRAcc* score. This step takes $O(\beta^2)$. Thus, Algorithm 2 takes $O(m + \beta^2)$ time.

FLEXI with *WRAcc* measure (FLEXI$_w$) hence takes $O(m + \beta^2 + \beta^3) = O(m + \beta^3)$ time.

4.2 *z-score* **measure** The *z-score* measure is suited when $\mathbf{D}$ has a single numeric target T. Let μ_0 and σ_0 be the mean and standard deviation of T in $\mathbf{D}$. Consider a subgroup S and let μ and σ be the mean and standard deviation of T in S. The quality of S w.r.t. *z-score* is defined as

$$z\text{-}score(S) = \frac{\sqrt{s}}{\sigma_0}(\mu - \mu_0) \quad ,$$

where $s = |\mathbf{D}_S|$. To pre-compute $z\text{-}score(\bigcup_{k=j}^{i} c_k)$ for all $1 \le j \le i \le \beta$, we can re-use Algorithm 2 with a few modifications. The new algorithm is in Algorithm 3. It also takes $O(m + \beta^2)$.

Hence, FLEXI with *z-score* measure (FLEXI$_z$) has the same complexity as FLEXI$_w$.

4.3 *kl* **measure** Kullback-Leibler divergence (*kl*) is suited for $\mathbf{D}$ with univariate/multivariate nominal and/or ordinal targets. W.l.o.g., assume that we have multivariate target $\mathbf{T} = \{T_1, \dots, T_d\}$. The *kl* score of each subgroup S is defined as

$$kl(S) = \frac{s}{m} \sum_{t_1, \dots, t_d} p_S(t_1, \dots, t_d) \times \log \frac{p_S(t_1, \dots, t_d)}{p(t_1, \dots, t_d)} \quad ,$$

Algorithm 3 PRE-COMPUTATION WITH *z-score*

```
Create an integer array $binMean[1 \ldots \beta]$
for $i = 1 \to \beta$ do
    $binMean[i]$ = target mean in $c_i$
    Compute $z\text{-}score(c_i)$ based on $binMean[i]$
end for
for $i = 2 \to \beta$ do
    $\theta = |c_i|$
    $\mu = binMean[i]$
    for $j = i - 1 \to 1$ do
        $\mu = \theta \times \mu + |c_j| \times binMean[j]$
        $\theta = \theta + |c_j|$
        Set target mean in $\bigcup_{k=j}^{i} c_k$ to $\mu$ and hence compute
        $z\text{-}score(\bigcup_{k=j}^{i} c_k)$
    end for
end for
```

where $s = |\mathbf{D}_S|$. A straightforward computation of $kl(\bigcup_{k=j}^{i} c_k)$ for every $1 \leq j \leq i \leq \beta$ is done by considering only $(t_1, \ldots, t_d)$ that appears in the data covered by S. This is because $p_S(t_1, \ldots, t_d) \times \log \frac{p_S(t_1,\ldots,t_d)}{p(t_1,\ldots,t_d)} = 0$ for $(t_1, \ldots, t_d)$ not in S. As $p_S(t_1, \ldots, t_d)$ and $p(t_1, \ldots, t_d)$ can be efficiently calculated using hash tables, computing $kl(\bigcup_{k=j}^{i} c_k)$ takes $O((i - j + 1) \times d \times \frac{m}{\beta})$. The pre-computation hence in total takes

$$\sum_{i=1}^{\beta} \sum_{j=1}^{i} O((i - j + 1) \times d \times \tfrac{m}{\beta}) \quad ,$$

which can be simplified to $O(m\beta^2 d)$.

Thus, FLEXI with kl (FLEXI$_k$) takes $O(m\beta^2 d + \beta^3) = O(m\beta^2 d)$ as $\beta \ll m$ time.

4.4 hd **measure** Similarly to kl measure, Hellinger distance (hd) is suited for $\mathbf{D}$ with univariate/multivariate nominal and/or ordinal targets. The hd score of a subgroup S is defined as

$$hd(S) = \left(-\tfrac{s}{m} \log \tfrac{s}{m} - \tfrac{m-s}{m} \log \tfrac{m-s}{m}\right) \times \sum_{t_1,\ldots,t_d} \left(\sqrt{p_S(t_1, \ldots, t_d)} - \sqrt{p(t_1, \ldots, t_d)}\right)^2 \quad .$$

where $s = |\mathbf{D}_S|$. The pre-computation is done similarly to Section 4.3. However, we here need to consider $(t_1, \ldots, t_d)$ that appears in $\mathbf{D}$, not just in S. Thus, for (i, j) where $j \leq i$, computing $hd(\bigcup_{k=j}^{i} c_k)$ takes $O(md)$. Hence, the cost of the pre-computation is identical to that of FLEXI$_k$.

In other words, FLEXI with hd measure (FLEXI$_h$) has the same complexity as FLEXI$_k$.

4.5 qr **measure** To handle univariate/multivariate numeric and/or ordinal targets, we propose qr, a measure based on ID – a quadratic measure of divergence [14]. We pick ID as it is applicable to both univariate and multivariate data. In addition, its computation on empirical data is in closed form formula, i.e. it is highly suited to exploratory data analysis. Originally, ID is used for numeric data. Our qr measure improves over this by adapting ID to ordinal data. This enables qr to handle multivariate numeric targets, as well as multivariate targets whose types are a mixed of numeric and ordinal. As shown in Table 1, no previous measure is able to achieve this. By making qr work with FLEXI, we can further demonstrate the flexibility and generality of our solution. The details are as follows.

Consider a subgroup S with $s = |\mathbf{D}_S|$ objects. W.l.o.g., assume that there are multiple targets. The qr score of S is

$$qr(S) = f(s) \times ID(p_S(\mathbf{T}) \,||\, p(\mathbf{T})) \quad ,$$

where $f(s)$ is either $\frac{s}{m}$ (following [5, 21]) or $\left(\frac{s}{m} \log \frac{s}{m} - \frac{m-s}{m} \log \frac{m-s}{m}\right)$ (following [2, 10]). When all targets are numeric, we have $ID(p_S(\mathbf{T}) \,||\, p(\mathbf{T})) =$

$$\int_{v_1}^{V_1} \cdots \int_{v_d}^{V_d} \left(P_S(t_1, \ldots, t_d) - P(t_1, \ldots, t_d)\right)^2 dt_1 \cdots dt_d \quad ,$$

where $P_S(.)$ and $P(.)$ are the cdfs of $p_S(.)$ and $p(.)$, respectively. We extend to ordinal targets by replacing $\int_{v_i}^{V_i} dt_i$ with $\sum\limits_{t_i \in dom(T_i)}$ for each ordinal T_i.

Similarly to ID, our qr measure also permits computation on empirical data in closed form. More specifically, let the empirical data of $\mathbf{D}$ be $\{\mathbf{D}^1, \ldots, \mathbf{D}^m\}$. Similarly, let the empirical data of $\mathbf{D}_S$ be $\{\mathbf{D}_S^1, \ldots, \mathbf{D}_S^s\}$ where $s = |\mathbf{D}_S|$. We write $\mathbf{D}_i^1$ and $\mathbf{D}_{S,i}^1$ as the projections of $\mathbf{D}^1$ and respectively $\mathbf{D}_S^1$ on T_i. We have the following.

THEOREM 4.1. *Empirically,* $qr(p_S(\mathbf{T}) \,||\, p(\mathbf{T})) =$

$$f(s) \times \left(\frac{1}{s^2} \sum_{i=1}^{s} \sum_{j=1}^{s} \prod_{k=1}^{d} h_k(\mathbf{D}_{S,i}^k, \mathbf{D}_{S,j}^k) - \frac{2}{sm} \sum_{i=1}^{s} \sum_{j=1}^{m} \prod_{k=1}^{d} h_k(\mathbf{D}_{S,i}^k, \mathbf{D}_j^k) + \frac{1}{m^2} \sum_{i=1}^{m} \sum_{j=1}^{m} \prod_{k=1}^{d} h_k(\mathbf{D}_i^k, \mathbf{D}_j^k) \right) \quad ,$$

where $h_k(x, y) = (V_k - \max(x, y))$ *if* T_k *is numeric, and* $h_k(x, y) = \sum\limits_{t \in dom(T_k)} \mathbb{I}(t \geq \max(x, y))$ *if* T_k *is ordinal. Here,* $\mathbb{I}(.)$ *is an indicator function.*

Proof. We postpone the proof to Appendix B.

Following Theorem 4.1, to obtain $qr(p_S(\mathbf{T}) \,||\, p(\mathbf{T}))$ we need to compute three terms – referred to as $S.e_1$, $S.e_2$, and $S.e_3$ – where

$$qr(p_S(\mathbf{T}) \,||\, p(\mathbf{T})) = f(s) \times \left(\tfrac{1}{s^2} S.e_1 - \tfrac{2}{sm} S.e_2 + \tfrac{1}{m^2} S.e_3\right) \quad .$$

Note that $e = S.e_3$ is independent of S and thus needs to be computed only once for all subgroups. We now prove a property of qr which is important for efficiently pre-computing $qr(\bigcup_{k=j}^{i} c_k)$ for all $1 \leq j \leq i \leq \beta$.

LEMMA 4.1. *Let S and R be two consecutive non-overlapping bins of attribute A, i.e. $\mathbf{D}_S \cap \mathbf{D}_R = \emptyset$. Let $Y = S \cup R$, $s = |\mathbf{D}_S|$, and $r = |\mathbf{D}_R|$. It holds that $Y.e_1 = S.e_1 + R.e_1 + 2int(S, R)$ and $Y.e_2 = S.e_2 + R.e_2$ where $int(S, R) = \sum\limits_{i=1}^{s} \sum\limits_{j=1}^{r} \prod\limits_{k=1}^{d} h_k(\mathbf{D}_{S,i}^k, \mathbf{D}_{R,j}^k)$.*

Proof. We postpone the proof to Appendix B.

Lemma 4.1 tells us that terms e_1 and e_2 of a bin made up by joining two adjacent non-overlapping bins S and R can be obtained from the terms of S and R, and $int(S, R)$. Note that int is symmetric. Further, we prove that it is additive – a property that is also important for the pre-computation.

LEMMA 4.2. *Let $R_1, \ldots, R_l$, and S be non-overlapping bins of A such that R_i is adjacent to R_{i+1} for $i \in [1, l-1]$, and R_l is adjacent to S. It holds that*

$$int\left(S, \bigcup_{i=1}^{l} R_i\right) = \sum_{i=1}^{l} int(S, R_i) \quad .$$

Proof. We postpone the proof to Appendix B.

Algorithm 4 summarises how to compute $qr(\bigcup_{k=j}^{i} c_k)$ for all $1 \leq j \leq i \leq \beta$. The details are as follows.

- First, we compute terms e_1 and e_2, and $qr(c_i)$ for each $i \in [1, \beta]$ (Line 1): This step takes $O(m \times \frac{m}{\beta} \times d)$ for each c_i, i.e. its total cost is $O(m^2 d)$.
- Second, we compute $int(c_j, c_i)$ for each $j \in [1, \beta - 1]$ and $i \in [j+1, \beta]$ (Line 2): This step takes $O(\frac{m^2}{\beta^2} d)$ for each pair (j, i), i.e. its total cost is $O(m^2 d)$.
- Third, we compute $int(\bigcup_{k=j}^{i-1} c_k, c_i)$ for each $i \in [2, \beta]$ and $j \in [1, i-1]$ (Lines 3 to 9): We use the fact that $int(\bigcup_{k=j}^{i-1} c_k, c_i) = \sum_{k=j}^{i-1} int(c_k, c_i)$ (see Lemma 4.2). This step takes $O(\beta^2)$.
- Fourth, we compute terms e_1 and e_2, and $qr(\bigcup_{k=j}^{i} c_k)$ for each $i \in [2, \beta]$ and $j \in [1, i-1]$ (Lines 10 to 14): From Lemma 4.1, terms e_1 and e_2 of $\bigcup_{k=j}^{i} c_k$ can be computed based on the terms of $\bigcup_{k=j}^{i-1} c_k$, c_i, and $int(\bigcup_{k=j}^{i-1} c_k, c_i)$. This step takes $O(\beta^2)$.

Overall, Algorithm 4 takes $O(m^2 d)$. Thus, FLEXI with qr (FLEXI$_q$) takes $O(m^2 d + \beta^3) = O(m^2 d)$ as $\beta \ll m$.

Algorithm 4 PRE-COMPUTATION WITH qr

```
1:  Compute terms $e_1$ and $e_2$, and $qr(c_i)$ for $c_i$ ($i \in [1, \beta]$)
2:  Compute $int(c_j, c_i)$ for every $j \in [1, \beta - 1]$ and $i \in [j+1, \beta]$
3:  for $i = 2 \to \beta$ do
4:      $\theta = 0$
5:      for $j = i - 1 \to 1$ do
6:          $\theta = \theta + int(c_j, c_i)$
7:          Set $int(\bigcup_{k=j}^{i-1} c_k, c_i)$ to $\theta$
8:      end for
9:  end for
10: for $i = 2 \to \beta$ do
11:     for $j = 1 \to i - 1$ do
12:         Compute terms $e_1$ and $e_2$, and $qr(\bigcup_{k=j}^{i} c_k)$ for $\bigcup_{k=j}^{i} c_k$ using the terms of $\bigcup_{k=j}^{i-1} c_k$, $c_i$, and $int(\bigcup_{k=j}^{i-1} c_k, c_i)$
13:     end for
14: end for
```

4.6 Remarks As β is typically small (from 5 to 40), FLEXI$_w$, FLEXI$_z$, FLEXI$_k$, and FLEXI$_h$ all scale linearly in m. On the other hand, FLEXI$_q$ scales quadratic in m regardless which value β takes. In Section 5, we propose a method to boost the efficiency of FLEXI$_q$.

5 Improving Scalability

The complexity of FLEXI$_q$ is quadratic in m, which may become a disadvantage on large data. We thus propose a solution to alleviate the issue. Again, we keep our discussion to the case of one-dimensional subgroups. The case of refinements straightforwardly follows.

We observe that the performance bottleneck is the pre-computations of $qr(c_i)$ ($i \in [1, \beta]$) and $int(c_j, c_i)$ ($j \in [1, \beta - 1]$ and $i \in [j+1, \beta]$). In fact, keys to these quantities are the distributions $p_{c_i}(\mathbf{T})$ in bins c_i ($i \in [1, \beta]$). In our computation the data of $\mathbf{D}_{c_i}$ projected onto $\mathbf{T}$, denoted as $\mathbf{D}_{c_i,\mathbf{T}}$, is considered to be i.i.d. samples of the (unknown) pdf $p_{c_i}(\mathbf{T})$. By definition, i.i.d. samples are obtained by randomly sampling from an infinite population or by randomly sampling with replacement from a finite population [19]. In both cases, the distribution of i.i.d. samples are assumed to be identical to the distribution of the population. This is especially true when the sample size is very large [18]. Thus, when m is very large the size of $\mathbf{D}_{c_i,\mathbf{T}}$ – which is $\frac{m}{\beta}$ – is also large. This makes the empirical distribution $\hat{p}_{c_i}(\mathbf{T})$ formed by $\mathbf{D}_{c_i,\mathbf{T}}$ approach the true distribution $p_{c_i}(\mathbf{T})$.

Assume now that we randomly draw with replacement $\epsilon \times \frac{m}{\beta}$ samples $\mathbf{d}_{c_i,\mathbf{T}}$ from $\mathbf{D}_{c_i,\mathbf{T}}$ where $\epsilon \in (0, 1)$. As mentioned above, $\mathbf{d}_{c_i,\mathbf{T}}$ contains i.i.d. samples of $\hat{p}_{c_i}(\mathbf{T}) \approx p_{c_i}(\mathbf{T})$. As with any set of i.i.d. samples with a reasonable size, we can assume that the distribution of $\mathbf{T}$ in $\mathbf{d}_{c_i,\mathbf{T}}$ is identical to $p_{c_i}(\mathbf{T})$.

Based on this line of reasoning, when m is large we propose to randomly sub-sample with replacement the data in each bin c_i ($i \in [1, \beta]$) for our computation. The important point here is to identify how large ϵ should be, i.e. how many samples we should use. We will show that a low value of ϵ already suffices, e.g. $\epsilon = 0.1$. If we sub-sample the bins while not sub-sampling $\mathbf{D}$ (in the same way) for computing quality scores, the complexity of FLEXI$_q$ is $O(\epsilon m^2 d)$. If we sub-sample $\mathbf{D}$ as well, its complexity is then $O(\epsilon^2 m^2 d)$.

6 Related Work

Traditionally, subgroup discovery focuses on nominal attributes [4, 6, 7, 9]. More recent work [2, 11, 13, 20] considers numeric attributes, employing equal-width or equal-frequency binning to create binary features. These strategies however do not optimise quality of the features generated, which consequently affects the final output quality.

To alleviate this, Grosskreutz and Rüping [5] employ SD [3]. It requires that the target is univariate and nominal. Further, it finds the bins optimising the divergence between $p_b(\mathbf{T})$ and $p_{b'}(\mathbf{T})$ where b and b' are two arbitrary consecutive bins. That is, only local distributions of the target (within individual bins) are compared to each other. The goal of subgroup discovery in turn is to assess the divergence between $p_b(\mathbf{T})$ and $p(\mathbf{T})$ [6]. While SD improves over naïve binning methods, it does not directly optimise subgroup quality.

Mampaey et al. [12] introduce ROC, which searches for the binary feature with highest quality on each numeric/ordinal attribute. It does so by analysing the coverage space, reminiscent of ROC spaces, of the *univariate* target. ROC and FLEXI are different in many aspects. First, ROC is suitable for univariate targets only. Second, it is designed for mining one-dimensional refinements of subgroups, and is not well-suited to find univariate subgroups. Third, it requires ϕ to be convex. FLEXI, on the other hand, works with any type of quality measure, can discover both high-quality refinements as well as one-dimensional subgroups, and works for both univariate and multivariate targets.

Besides the binning methods discussed above, there exist also other techniques applicable to – albeit not yet studied in – subgroup discovery. For instance, UD [8] mines bins per numeric attribute that best approximate its true distribution. On the other hand, multivariate binning techniques (e.g. IPD [14]) focus on optimising the divergence between local distributions in individual bins. Overall, these methods do not optimise subgroup quality.

Regarding quality measure ϕ, majority of existing ones focus on univariate targets [4–7, 9, 11–13, 22]. Van Leeuwen and Knobbe [20, 21] propose a measure based on Kullback-Leibler divergence for multivariate nominal/ordinal targets. Their measure is reminiscent of *kl* measure in Section 4.3; yet, they assume the targets are statistically independent while *kl* takes into account interaction of targets. Also for multivariate nominal/ordinal targets, Duivesteijn et al. [2] introduce a measure based on a Bayesian network. Measures for multivariate numeric targets appear mainly in exceptional model mining (EMM) [1, 10]. Consequently, such measures are model-based. Our *qr* measure in turn is purely non-parametric. Recently, we introduced a non-parametric measure for multivariate numeric targets [16]. Unlike both this measure and those of EMM, *qr* can handle multivariate targets whose types are a mixed of numeric and ordinal.

		Attributes			
Data	**Rows**	Nom.	Ord.	Num.	Total
Adult	48 842	7	1	6	14
Bike	17 379	5	3	7	15
Cover	581 012	44	3	7	54
Gesture	9 900	1	0	32	33
Letter	20 000	1	0	16	16
Bank	45 211	11	2	8	21
Naval	11 934	0	0	18	18
Network	53 413	1	9	14	24
SatImage	6 435	1	0	36	37
Drive	58 509	1	0	48	49
Turkiye	5 820	0	32	1	33
Year	515 345	1	0	90	91

Table 2: Characteristics of real-world data sets. We give the number of rows, the number of resp. nominal, ordinal, and numeric attributes, and the total number of attributes.

7 Experiments

In this section, we empirically evaluate FLEXI through beam search – a common search scheme of subgroup discovery [2, 10, 20]. We aim at examining if FLEXI is able to efficiently and effectively discover subgroups of high quality. For a comprehensive assessment, we test with all five quality measures discussed above. As performance metric, we use the average quality of top 50 subgroups. We also study the parameter setting of FLEXI; this includes the effect of our scalability improvement for *qr* measure (see Section 5). We implemented FLEXI in Java, and make our code available for research purposes.[1] All experiments were performed single-threaded on an Intel(R) Core(TM) i7-4600U CPU with 16GB RAM. We report wall-clock running times.

We compare FLEXI to SUM which finds bins optimising the sum of quality instead of average quality, EF for equal-frequency binning, and EW for equal-width binning. As further baselines, we test with state of the art *supervised* discretisation SD [3], *unsupervised univariate* discreti-

[1] http://eda.mmci.uni-saarland.de/flexi/

sation UD [8], and *unsupervised multivariate* discretisation IPD [14]. For measures that handle univariate targets only ($WRAcc$ and $z\text{-}score$), we test with UD and exclude IPD. For the other three measures, we use IPD instead. Finally, we include ROC [12], state of the art method on mining binary features for subgroup discovery. For all competitors, we optimise their parameters and report the best results. For FLEXI, by default we set the number of initial bins $\beta = 20$; and when subsampling is used, we set the subsampling rate $\epsilon = 0.1$. We form initial bins $\{c_1, \ldots, c_\beta\}$ by applying equal-frequency binning; this procedure has also been used in [14, 15, 17].

We experiment with 12 real-world data sets drawn from the UCI Machine Learning Repository. Their details are in Table 2. To show that FLEXI methods are suited to subgroup discovery on large-scale data, 9 data sets we pick have more than 10 000 records. For brevity, in the following we present the results on 6 data sets with largest sizes: Adult, Cover, Bank, Network, Drive, and Year. For conciseness, we keep our discussion to FLEXI$_w$, FLEXI$_k$, and FLEXI$_q$.

7.1 Quality results with $WRAcc$ As $WRAcc$ requires univariate binary target, we follow [13] to convert nominal (but non-binary) targets to binary. The results are in Tables 3. Here, we display the absolute as well as relative average quality (for other measures we show relative quality only). For the relative quality, the scores of FLEXI$_w$ are the bases (100%). Going over the results, we see that FLEXI$_w$ gives the best average quality in all data sets. Its performance gain over the competitors is up to 300%. Note that by optimising average subgroup quality, instead of total quality as SUM does, FLEXI$_w$ mines better binary features and hence achieves better performance than SUM. EF, EW, SD, and UD form binary features oblivious of subgroup quality and perform less well. ROC, on the other hand, performs better, but as it forms one feature per attribute at each level of the search it makes the search more sensitive to local optima.

7.2 Quality results with kl We recall that kl is suited to univariate/multivariate nominal and/or ordinal targets. For Adult and Bank, we use all nominal attributes as targets. For Cover, we randomly select 27 nominal attributes as targets. For Network, we combine nominal and ordinal attributes to create the targets. Drive and Year both have one nominal attribute and no ordinal one. Thus, for each of them we use the nominal attribute as univariate target.

The results are in Table 4. FLEXI$_k$ achieves the best performance in all data sets. It yields up to 25 times quality improvement compared to competing methods. Note that SD and ROC both require univariate targets and hence are not applicable to Adult, Cover, Bank, and Network. FLEXI$_k$ in turn is suited to both univariate and multivariate targets.

7.3 Quality results with qr We recall that qr is suited to univariate/multivariate numeric and/or ordinal targets. In this experiment, we focus on multivariate targets; hence, SD and ROC are inapplicable. Regarding the setup, for Adult we combine the ordinal attribute and two randomly selected numeric attributes to form targets. For Cover, we pick three ordinal attributes as targets. For Bank, we combine the two ordinal attributes and two randomly selected numeric attributes to create targets. For Network, we randomly sample five ordinal attributes and five numeric attributes to form targets. For Drive and Year, we randomly pick half of the numeric attributes as targets.

To avoid runtimes of more than 5 hours on Cover, Network, Drive, and Year, for *all* methods we subsample with $\epsilon = 0.1$. Note that with EF, EW, and IPD, we need to compute subgroup quality after the bins have been formed, which in total is quadratic to the data size m. That is, for efficiency subsampling is necessary. For the final subgroups, we use their actual quality for evaluation. Every reported quality measurement is the average of 10 independent runs; standard deviation is small and hence skipped.

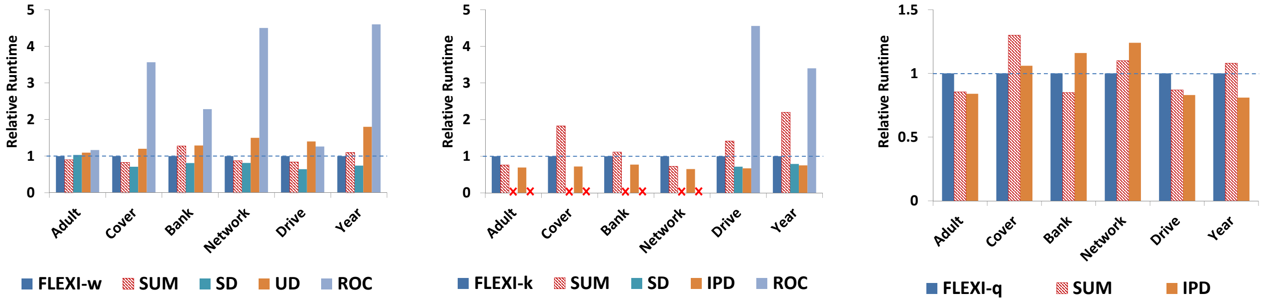
The results are in Table 5. We see that FLEXI$_q$ outperforms all competitors with large margins, improving quality up to 14 times.

7.4 Efficiency results We here compare the efficiency of methods that have an advanced way to form binary features; that is, for fairness we skip EF and EW. The relative runtime of all remaining methods are shown in Figures 1(a), 1(b), and 1(c). The results of our methods in each case are the bases. We observe that we overall are faster than ROC. This could be attributed to the fact that we form initial bins before mining actual features. ROC in turn uses the original set of cut points and hence has a larger search space per attribute. We can also see that our methods have comparable runtime to SUM. While in theory SUM is more efficient than our method, it may unnecessarily form too many binary features per attribute, which potentially incurs higher runtime for the whole subgroup discovery process.

7.5 Parameter setting FLEXI has two input parameters: the number of initial bins β and the subsampling rate ϵ. To assess the sensitivity to β, we vary it from 5 to 40 with step size being 5. For sensitivity to ϵ, we vary it from 0.05 to 0.2 with step size being 0.05. The default setting is $\beta = 20$ and $\epsilon = 0.1$. The results are in Figures 2(a) and 2(b). For β, we show representative outcome of FLEXI$_w$ and FLEXI$_k$ on Adult and Bank. For ϵ, we show outcome of FLEXI$_q$ on Network and Drive. We can see that our methods are robust with regard to how we set the parameters.

Data	FLEXI$_w$	SUM	EF	EW	SD	UD	ROC
Adult	**0.08 (100)**	0.07 (88)	0.07 (88)	0.07 (88)	0.07 (88)	0.06 (75)	0.07 (88)
Cover	**0.12 (100)**	0.11 (92)	0.04 (33)	0.08 (66)	0.04 (33)	0.05 (42)	0.04 (33)
Bank	**0.04 (100)**	0.03 (75)	0.02 (50)	0.03 (75)	0.02 (50)	0.02 (50)	0.02 (50)
Network	**0.18 (100)**	0.13 (72)	0.10 (56)	0.12 (67)	0.14 (78)	0.12 (67)	0.14 (78)
Drive	**0.11 (100)**	0.08 (73)	0.03 (27)	0.08 (73)	0.05 (45)	0.06 (55)	0.05 (45)
Year	**0.12 (100)**	0.08 (67)	0.06 (50)	0.06 (50)	0.07 (58)	0.06 (50)	0.07 (58)

Table 3: [Higher is better] Average quality, measured by $WRAcc$, of top 50 subgroups. We give both the absolute scores, as well as the relative results (in brackets) compared to FLEXI$_w$.



(a) Runtime with $WRAcc$

(b) Runtime with kl

(c) Runtime with qr

Figure 1: [Lower is better] Relative runtime with $WRAcc$, kl, and qr. In each case the runtime of FLEXI is the base. SD and ROC are not applicable to Adult, Cover, Bank, and Network, which is marked by ✗.

Data	FLEXI$_k$	SUM	EF	EW	SD	IPD	ROC
Adult	**100**	38	37	31	*n/a*	4	*n/a*
Cover	**100**	43	64	75	*n/a*	45	*n/a*
Bank	**100**	46	62	33	*n/a*	6	*n/a*
Network	**100**	55	68	55	*n/a*	21	*n/a*
Drive	**100**	42	64	85	89	42	62
Year	**100**	43	45	42	40	42	74

Table 4: [Higher is better] Average quality, measured by kl, of top 50 subgroups. The results are relative and the quality of FLEXI$_k$ on each data set is the base (100%).

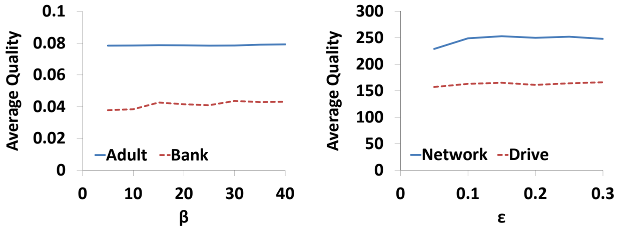
Data	FLEXI$_q$	SUM	EF	EW	IPD
Adult	**100**	18	7	8	23
Cover	**100**	60	41	39	53
Bank	**100**	31	47	59	66
Network	**100**	48	69	64	56
Drive	**100**	62	41	59	66
Year	**100**	26	27	21	55

Table 5: [Higher is better] Average quality, measured by qr, of top 50 subgroups. The results are relative and the quality of FLEXI$_q$ on each data set is the base (100%).

8 Discussion

The experiments on different quality measures and real-world data sets show that FLEXI found subgroups of higher quality than existing methods. In terms of efficiency, it is on par with SUM and faster than ROC– the state of the art for mining binary features for subgroup discovery. The good performance of FLEXI is attributable to 1) taking subgroup quality into account in binary feature mining, (2) finding optimal binary features by dynamic programming, and (3) using subsampling to handle very large data sets.

Yet, there is room for alternative methods as well as further improvements. For instance, in addition to beam search it is also interesting to apply FLEXI to other search paradigms, e.g. MDL-based search [20]. Along this line, we can also formulate our search problem as mining binary features with high quality that together effectively compress the data. Besides the already demonstrated efficiency of our method, it can be trivially sped up by parallelisation, e.g. with MapReduce. This direction is applicable to subgroup discovery in general, and a potential solution to apply this paradigm on large real-world scenarios.



(a) Quality vs. β

(b) Quality vs. ϵ

Figure 2: Sensitivity to β and ϵ. For β, we show the results of FLEXI_w and FLEXI_k on Adult and Bank. For ϵ, we show the results of FLEXI_q on Network and Drive.

9 Conclusion

We studied the problem of mining binary features for subgroup discovery. This is challenging as one needs a formulation that allows us to identify features leading to the detection of high quality subgroup. Second, the solution should place no restrictions on the target. Third, it should permit efficient computation. To address these issues, we proposed FLEXI. In short, FLEXI aims at identifying binary features per attribute with maximal average quality. The formulation of FLEXI is abstract from the targets and hence suited to any type of targets. We instantiated FLEXI with five different measures and showed how to make it efficient in every case. Extensive experiments on various real-world data sets verified that compared to existing methods, FLEXI is able to efficiently detect subgroups with considerably higher quality.

Acknowledgements

The authors are supported by the Cluster of Excellence "Multimodal Computing and Interaction" within the Excellence Initiative of the German Federal Government.

References

[1] W. Duivesteijn, A. Feelders, and A. J. Knobbe. Different slopes for different folks: mining for exceptional regression models with cook's distance. In *KDD*, pages 868–876, 2012.

[2] W. Duivesteijn, A. J. Knobbe, A. Feelders, and M. van Leeuwen. Subgroup discovery meets bayesian networks – an exceptional model mining approach. In *ICDM*, pages 158–167, 2010.

[3] U. M. Fayyad and K. B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *IJCAI*, pages 1022–1029, 1993.

[4] H. Grosskreutz and D. Paurat. Fast and memory-efficient discovery of the top-k relevant subgroups in a reduced candidate space. In *ECML/PKDD (1)*, pages 533–548, 2011.

[5] H. Grosskreutz and S. Rüping. On subgroup discovery in numerical domains. *Data Min. Knowl. Discov.*, 19(2):210–226, 2009.

[6] H. Grosskreutz, S. Rüping, and S. Wrobel. Tight optimistic estimates for fast subgroup discovery. In *ECML/PKDD (1)*, pages 440–456, 2008.

[7] W. Klösgen. *Advances in knowledge discovery and data mining, Chapter Explora: a multipattern and multistrategy discovery assistant*. MIT Press, Cambridge, 1996.

[8] P. Kontkanen and P. Myllymäki. MDL histogram density estimation. In *AISTATS*, pages 219–226, 2007.

[9] N. Lavrac, B. Kavsek, P. A. Flach, and L. Todorovski. Subgroup discovery with CN2-SD. *JMLR*, 5:153–188, 2004.

[10] D. Leman, A. Feelders, and A. J. Knobbe. Exceptional model mining. In *ECML/PKDD*, pages 1–16, 2008.

[11] F. Lemmerich, M. Becker, and F. Puppe. Difference-based estimates for generalization-aware subgroup discovery. In *ECML/PKDD (3)*, pages 288–303, 2013.

[12] M. Mampaey, S. Nijssen, A. Feelders, R. M. Konijn, and A. J. Knobbe. Efficient algorithms for finding optimal binary features in numeric and nominal labeled data. *Knowl. Inf. Syst.*, 42(2):465–492, 2015.

[13] M. Meeng, W. Duivesteijn, and A. J. Knobbe. ROCsearch - an ROC-guided search strategy for subgroup discovery. In *SDM*, pages 704–712, 2014.

[14] H. V. Nguyen, E. Müller, J. Vreeken, and K. Böhm. Unsupervised interaction-preserving discretization of multivariate data. *Data Min. Knowl. Discov.*, 28(5-6):1366–1397, 2014.

[15] H. V. Nguyen, E. Müller, J. Vreeken, P. Efros, and K. Böhm. Multivariate maximal correlation analysis. In *ICML*, pages 775–783, 2014.

[16] H. V. Nguyen and J. Vreeken. Non-parametric jensen-shannon divergence. In *ECML/PKDD*, pages 173–189, 2015.

[17] D. N. Reshef, Y. A. Reshef, H. K. Finucane, S. R. Grossman, G. McVean, P. J. Turnbaugh, E. S. Lander, M. Mitzenmacher, and P. C. Sabeti. Detecting novel associations in large data sets. *Science*, 334(6062):1518–1524, 2011.

[18] D. W. Scott. *Multivariate Density Estimation: Theory, Practice, and Visualization*. John Wiley & Sons Inc, New York, 1992.

[19] S. K. Thompson. *Sampling*. Wiley, 3rd edition, 2012.

[20] M. van Leeuwen and A. J. Knobbe. Non-redundant subgroup discovery in large and complex data. In *ECML/PKDD (3)*, pages 459–474, 2011.

[21] M. van Leeuwen and A. J. Knobbe. Diverse subgroup set discovery. *Data Min. Knowl. Discov.*, 25(2):208–242, 2012.

[22] M. van Leeuwen and A. Ukkonen. Discovering skylines of subgroup sets. In *ECML/PKDD (3)*, pages 272–287, 2013.

[23] S. Wrobel. An algorithm for multi-relational discovery of subgroups. In *PKDD*, pages 78–87, 1997.

Algorithm 5 SUM

```
Create initial disjoint bins $\{c_1, \ldots, c_\beta\}$ of $A$
Create a double array $qual[1 \ldots \beta]$
Create an array $b[1 \ldots \beta]$ whose each entry stores bins
Set $qual[1] = \phi(c_1)$ and $b[1] = c_1$
for $i = 2 \rightarrow \beta$ do
    $pos = \arg\max_{1 \leq j \leq i-1} qual[j] + \phi(\bigcup_{k=j+1}^{i} c_k)$
    $qual[i] = qual[pos] + \phi(\bigcup_{k=pos+1}^{i} c_k)$
    Copy all bins in $b[pos]$ to $b[i]$
    Add $\bigcup_{k=pos+1}^{i} c_k$ to $b[i]$
end for
Return $b[\beta]$
```

A Alternative Setting

Here we show that the alternate problem formulation can also be solved by dynamic programing. More specifically, let dsc be the optimal solution and $\{b_{dsc}^1, \ldots, b_{dsc}^{|dsc|}\}$ be its bins. It holds that

$$\sum_{i=1}^{|dsc|} \phi(b_{dsc}^i) = \phi(b_{dsc}^{|dsc|}) + \sum_{i=1}^{|dsc|-1} \phi(b_{dsc}^i).$$

As dsc is optimal, $\{b_{dsc}^1, \ldots, b_{dsc}^{|dsc|-1}\}$ must be the optimal binning for values $A \leq l_{dsc}^{|dsc|}$. Otherwise, we could have chosen a different binning for such values that improves the total quality. This would yield another binning for all values of A that has a total quality higher than that of dsc, which contradicts the assumption on dsc. Hence, the optimal binning dsc also exhibits optimal substructure, permitting the use of dynamic programming. The detailed solution is in Algorithm 5.

B Proofs

Proof. [Theorem 4.1] W.l.o.g., we assume that $T_1, \ldots, T_l$ are numeric and $T_{l+1}, \ldots, T_d$ are ordinal. We have

$$P(t_1, \ldots, t_d) = \int_{v_1}^{V_1} \cdots \int_{v_l}^{V_l} \sum_{t_{l+1} \in dom(T_{l+1})} \cdots \sum_{t_d \in dom(T_d)} \mathbf{I}(x_1 \leq t_1) \times \cdots \times \mathbf{I}(x_d \leq t_d) \times p(x_1, \ldots, x_d) dx_1 \cdots dx_d.$$

Similarly, we have

$$P_S(t_1, \ldots, t_d) = \int_{v_1}^{V_1} \cdots \int_{v_l}^{V_l} \sum_{t_{l+1} \in dom(T_{l+1})} \cdots \sum_{t_d \in dom(T_d)} \mathbf{I}(x_1 \leq t_1) \times \cdots \times \mathbf{I}(x_d \leq t_d) \times p_S(x_1, \ldots, x_d) dx_1 \cdots dx_d.$$

Using empirical data, we have

$$P(t_1, \ldots, t_d) = \frac{1}{m} \sum_{i=1}^{m} \prod_{k=1}^{d} \mathbf{I}(\mathbf{D}_k^i \leq t_i), \quad \text{and}$$

$$P_S(t_1, \ldots, t_d) = \frac{1}{s} \sum_{i=1}^{s} \prod_{k=1}^{d} \mathbf{I}(\mathbf{D}_{S,k}^i \leq t_i).$$

Hence, we have

$$ID(p_S(\mathbf{T}) \mid\mid p(\mathbf{T})) = \int_{v_1}^{V_1} \cdots \int_{v_l}^{V_l} \sum_{t_{l+1} \in dom(T_{l+1})} \cdots \sum_{t_d \in dom(T_d)} \left(\frac{1}{s} \sum_{i=1}^{s} \prod_{k=1}^{d} \mathbf{I}(\mathbf{D}_{S,k}^i \leq t_i) - \frac{1}{m} \sum_{i=1}^{m} \prod_{k=1}^{d} \mathbf{I}(\mathbf{D}_k^i \leq t_i) \right)^2 dt_1 \cdots dt_l.$$

Expanding the above term and bringing the integrals inside the sums, we have

$$\begin{aligned} &ID(p_S(\mathbf{T}) \mid\mid p(\mathbf{T})) = \\ &\frac{1}{s^2} \sum_{i=1}^{s} \sum_{j=1}^{s} \left(\prod_{k=1}^{l} \int_{v_i}^{V_i} \mathbf{I}(\max(\mathbf{D}_{S,k}^i, \mathbf{D}_{S,k}^j) \leq t_k) dt_k \right) \times \\ &\qquad \left(\prod_{k=l+1}^{d} \sum_{t_k \in dom(T_k)} \mathbf{I}(\max(\mathbf{D}_{S,k}^i, \mathbf{D}_{S,k}^j) \leq t_k) \right) \\ &- \frac{2}{sm} \sum_{i=1}^{s} \sum_{j=1}^{m} \left(\prod_{k=1}^{l} \int_{v_i}^{V_i} \mathbf{I}(\max(\mathbf{D}_{S,k}^i, \mathbf{D}_k^j) \leq t_k) dt_k \right) \times \\ &\qquad \left(\prod_{k=l+1}^{d} \sum_{t_k \in dom(T_k)} \mathbf{I}(\max(\mathbf{D}_{S,k}^i, \mathbf{D}_k^j) \leq t_k) \right) \\ &+ \frac{1}{m^2} \sum_{i=1}^{m} \sum_{j=1}^{m} \left(\prod_{k=1}^{l} \int_{v_i}^{V_i} \mathbf{I}(\max(\mathbf{D}_k^i, \mathbf{D}_k^j) \leq t_k) dt_k \right) \times \\ &\qquad \left(\prod_{k=l+1}^{d} \sum_{t_k \in dom(T_k)} \mathbf{I}(\max(\mathbf{D}_k^i, \mathbf{D}_k^j) \leq t_k) \right) \end{aligned}$$

by which we arrive at the final result.

Proof. [Lemma 4.1] Empirically, we have that

$$\begin{aligned} &div(p_{S \cup R}(\mathbf{T}) \mid\mid p(\mathbf{T})) \\ &= \frac{1}{(s+r)^2} \sum_{i=1}^{s+r} \sum_{j=1}^{s+r} \prod_{k=1}^{d} h_k(\mathbf{D}_{S \cup R,i}^k, \mathbf{D}_{S \cup R,j}^k) \\ &- \frac{2}{(s+r)m} \sum_{i=1}^{s+r} \sum_{j=1}^{m} \prod_{k=1}^{d} h_k(\mathbf{D}_{S \cup R,i}^k, \mathbf{D}_j^k) \\ &+ \frac{1}{m^2} \sum_{i=1}^{m} \sum_{j=1}^{m} \prod_{k=1}^{d} h_k(\mathbf{D}_i^k, \mathbf{D}_j^k). \end{aligned}$$

We can see that the first term is equal to $\frac{1}{(s+r)^2}S.e_1 + \frac{1}{(s+r)^2}R.e_1 + \frac{2}{(s+r)^2}int(S,R)$ where $int(S,R) = \sum_{i=1}^{s}\sum_{j=1}^{r}\prod_{k=1}^{d} h_k(\mathbf{D}^k_{S,i}, \mathbf{D}^k_{R,j})$. The second term is equal to $\frac{2}{(s+r)m}S.e_2 + \frac{2}{(s+r)m}R.e_2$. The third term is in fact e.

Proof. [Lemma 4.2] By definition, we have that

$$\begin{aligned} & int\left(S, \bigcup_{i=1}^{l} R_i\right) \\ & = \sum_{q=1}^{s} \sum_{j=1}^{s_1+\ldots+s_l} \prod_{k=1}^{d} h_k(\mathbf{D}^k_{S,q}, \mathbf{D}^k_{\bigcup_{i=1}^{l} R_i, j}) \\ & = \sum_{i=1}^{l}\sum_{q=1}^{s}\sum_{j=1}^{s_i}\prod_{k=1}^{d} h_k(\mathbf{D}^k_{S,q}, \mathbf{D}^k_{R_i,j}) \\ & = \sum_{i=1}^{l} int(S, R_i). \end{aligned}$$

C Additional Experimental Results

Quality results on all quality measures are in Tables 6, 7, 8, 9, and 10. Note that we show absolute values. As Naval has neither categorical nor ordinal attributes, it is not applicable to *WRAcc*, *kl*, and *hd*.

Additional efficiency results are in Figures 3(a), 3(b), and 3(c). Interestingly, on *qr* measure, FLEXI$_q$ is even faster than EW on 3 data sets. Our explanation is similar to the case of SUM; that is, EW may form unnecessarily many binary features than required per attribute which prolongs the runtime.

Data	**FLEXI$_w$**	**SUM**	**EF**	**EW**	**SD**	**UD**	**ROC**
Adult	**0.08**	0.07	0.07	0.07	0.07	0.06	0.07
Bike	**0.06**	0.04	0.04	0.04	0.06	0.04	0.05
Cover	**0.12**	0.11	0.04	0.08	0.04	0.05	0.04
Gesture	**0.10**	0.08	0.03	0.09	0.07	0.04	0.04
Letter	**0.08**	0.05	0.02	0.03	0.05	0.04	0.04
Bank	**0.04**	0.03	0.02	0.03	0.02	0.02	0.02
Network	**0.18**	0.13	0.10	0.12	0.14	0.12	0.14
SatImage	**0.15**	0.11	0.03	0.05	0.09	0.04	0.05
Drive	**0.11**	0.08	0.03	0.08	0.05	0.06	0.05
Turkiye	**0.11**	**0.11**	0.10	0.10	0.10	0.10	0.10
Year	**0.12**	0.08	0.06	0.06	0.07	0.06	0.07
Average	**0.10**	0.08	0.05	0.07	0.07	0.06	0.06

Table 6: [Higher is better] Average quality, measured by $WRAcc$, of top 50 subgroups. Best values are in **bold**.

Data	**FLEXI$_z$**	**SUM**	**EF**	**EW**	**UD**	**ROC**
Adult	**89.44**	82.14	82.14	86.04	79.62	82.14
Bike	**68.61**	50.44	57.54	50.24	56.25	61.50
Cover	**434.97**	328.43	356.44	249.49	288.29	384.48
Gesture	38.09	31.38	35.32	33.55	31.42	**44.01**
Letter	**47.11**	41.90	43.82	39.97	40.77	44.17
Bank	**78.76**	69.54	72.45	71.39	66.40	72.45
Naval	28.20	23.60	22.92	22.50	22.61	**32.25**
Network	135.09	129.38	133.60	114.78	110.45	**145.91**
SatImage	**50.28**	35.23	39.32	41.94	39.42	44.16
Drive	**120.33**	86.64	69.57	46.93	44.43	40.80
Turkiye	**14.56**	9.53	9.53	9.54	7.10	12.37
Year	**88.57**	57.59	47.93	53.50	50.40	60.31
Average	**99.50**	78.82	80.88	68.32	69.76	85.38

Table 7: [Higher is better] Average quality, measured by $z\text{-}score$, of top 50 subgroups. Best values are in **bold**.

Data	**FLEXI$_k$**	**SUM**	**EF**	**EW**	**SD**	**IPD**	**ROC**
Adult	**0.52**	0.20	0.19	0.16	*n/a*	0.02	*n/a*
Bike	**0.50**	0.26	0.34	0.35	*n/a*	0.05	*n/a*
Cover	**0.53**	0.23	0.34	0.40	*n/a*	0.24	*n/a*
Gesture	**0.53**	0.22	0.33	0.33	0.50	0.31	0.33
Letter	**0.52**	0.43	0.43	0.47	0.43	0.06	0.43
Bank	**0.52**	0.24	0.32	0.17	*n/a*	0.03	*n/a*
Network	**0.53**	0.29	0.36	0.29	*n/a*	0.11	*n/a*
SatImage	**0.53**	0.28	0.37	0.48	0.45	0.26	0.37
Drive	**0.53**	0.22	0.34	0.45	0.47	0.22	0.33
Turkiye	**0.53**	0.50	0.50	0.50	*n/a*	0.15	*n/a*
Year	**0.53**	0.23	0.24	0.22	0.21	0.22	0.39
Average	**0.52**	0.28	0.34	0.35	0.19	0.16	0.17

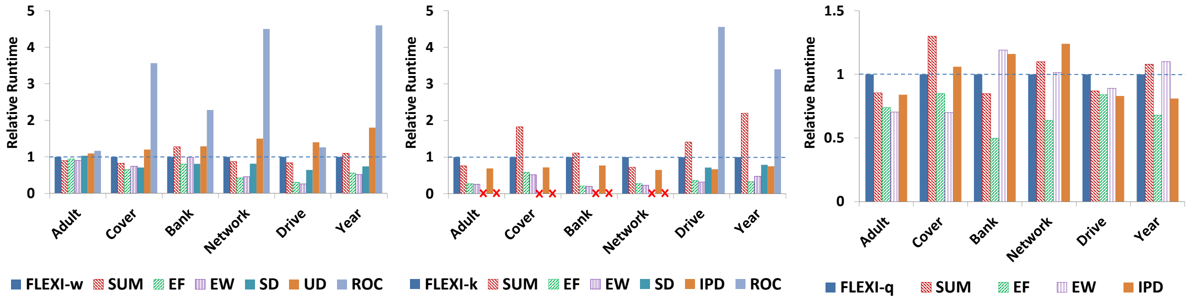
Table 8: [Higher is better] Average quality, measured by kl, of top 50 subgroups. Best values are in **bold**.

Data	FLEXI$_h$	SUM	EF	EW	SD	IPD	ROC
Adult	**0.29**	0.26	0.26	0.26	*n/a*	0.22	*n/a*
Bike	**0.27**	0.10	0.14	0.22	*n/a*	0.25	*n/a*
Cover	**0.30**	**0.30**	0.22	0.21	*n/a*	0.27	*n/a*
Gesture	0.29	0.08	0.14	**0.30**	0.27	**0.30**	0.14
Letter	**0.29**	0.21	0.21	0.25	0.24	0.25	0.28
Bank	**0.29**	0.13	0.16	0.23	*n/a*	0.26	*n/a*
Network	**0.29**	0.22	0.22	0.21	*n/a*	0.25	*n/a*
SatImage	**0.29**	0.11	0.16	0.24	0.23	0.23	0.17
Drive	0.29	0.08	0.14	0.28	0.29	**0.30**	0.14
Turkiye	**0.29**	0.26	0.26	0.26	*n/a*	0.26	*n/a*
Year	**0.29**	0.25	0.12	0.14	0.14	0.22	0.15
Average	**0.29**	0.18	0.18	0.24	0.11	0.26	0.08

Table 9: [Higher is better] Average quality, measured by hd, of top 50 subgroups. Best values are in **bold**.

Data	FLEXI$_q$	SUM	EF	EW	IPD
Adult	**110.35**	20.1	8.19	8.58	25.38
Bike	**1.77**	0.49	0.61	0.69	0.75
Cover	**185.72**	110.51	76.58	71.95	98.52
Gesture	**3.25**	0.82	1.13	2.58	2.86
Letter	**0.59**	0.35	0.36	0.41	0.44
Bank	**41.71**	13.02	19.60	24.54	27.63
Naval	**0.57**	0.18	0.21	0.26	0.28
Network	**25.72**	12.37	17.63	16.34	14.34
SatImage	**3.57**	1.23	2.20	1.94	2.11
Drive	**6.37**	3.94	2.64	3.76	4.22
Turkiye	**1.03**	0.85	0.77	0.83	0.83
Year	**271.98**	69.41	73.07	55.95	149.43
Average	**54.39**	19.44	16.92	15.65	27.23

Table 10: [Higher is better] Average quality, measured by qr, of top 50 subgroups. Best values are in **bold**.



(a) Runtime with $WRAcc$ (b) Runtime with kl (c) Runtime with qr

Figure 3: [Lower is better] Relative runtime with $WRAcc$, kl, and qr. The runtime of our methods in each case is the base. SD and ROC are not applicable to Adult, Cover, Bank, and Network, which is marked by ✗.