Journal of Intelligent Information Systems
https://doi.org/10.1007/s10844-022-00750-4

Efficient approximate top-k mutual information based feature selection

Md Abdus Salam[1] · Senjuti Basu Roy[2] · Gautam Das[3]

Received: 28 March 2022 / Revised: 12 August 2022 / Accepted: 20 September 2022

Abstract
Feature selection is an important step in the data science pipeline, and it is critical to develop efficient algorithms for this step. *Mutual Information* (MI) is one of the important measures used for feature selection, where attributes are sorted according to descending score of MI, and top-k attributes are retained. The goal of this work is to develop a new measure *Attribute Average Conflict* to effectively approximate top-*k* attributes, without actually calculating MI. Our proposed method is based on using the database concept of *approximate functional dependency* to quantify MI rank of attributes which to our knowledge has not been studied before. We demonstrate the effectiveness of our proposed measure with a Monte-Carlo simulation. We also perform extensive experiments using high dimensional synthetic and real datasets with millions of records. Our results show that our proposed method demonstrates *perfect* accuracy in selecting the top-*k* attributes, yet is significantly more efficient than state-of-art baselines, including exact methods for computing Mutual Information based feature selection, as well as adaptive random- sampling based approaches. We also investigate the upper and lower bounds of the proposed new measure and show that tighter bounds can be derived by using marginal frequency of attributes in specific arrangements. The bounds on the proposed measure can be used to select top-*k* attributes without full scan of the dataset in a single pass. We perform experimental evaluation on real datasets to show the accuracy and effectiveness of this approach.

Senjuti Basu Roy and Gautam Das These authors contributed equally to this work.

✉ Md Abdus Salam
mdsalam@uta.edu

Senjuti Basu Roy
senjutib@njit.edu

Gautam Das
gdas@uta.edu

[1] Department of Computer Science & Engineering, University of Texas at Arlington, Arlington, TX, USA

[2] Department Computer Science, New Jersey Institute of Technology, Newark, NJ, USA

[3] Department of Computer Science & Engineering, University of Texas at Arlington, Arlington, TX, USA

Published online: 18 October 2022

Keywords Feature selection · Top-k attribute selection · Mutual information · Data science

1 Introduction

Feature or attribute selection is considered to be a time-consuming yet highly essential step inside a data science pipeline, where the goal is to select a subset of features or attributes that exhibit high correlation with the class label to be predicted. Indeed, an effective small number of features play pivotal role in reducing computation time, and facilitates an enhanced understanding and improved efficacy of the underlying model. One of the important feature selection techniques is the filtering based method (Guyon & Elisseeff, 2003; Jović et al., 2015), which leverages scoring functions involving statistical property of the data to select features. *Mutual Information* (MI) (Vergara & Estévez, 2014; Hoque et al., 2014) is one such popular measure and has been extensively studied in recent works (Chen & Wang, 2021; Wang & Ding, 2019; Salam et al., 2019), due to its information theoretic interpretation, and ability in quantifying the predictive power of the attributes in a model agnostic fashion.

Our study unfolds in a classical data exploration setting inside data science pipeline, where given to us is a large database with millions of records designed over a hundreds of attributes (or features) and a class label Z. Given an attribute X and the class label Z, MI between X and Z measures the reduction in uncertainty for Z, given a known value of X. Our goal is to select top-k attributes with k-highest MI with the class label Z (Section 3). The aim of this work is to perform MI based feature selection without actually calculating the MI. The motivation is to avoid some computational overhead for the exact MI calculation. The novelty of the proposed approach is to connect Functional Dependency and Approximate Functional Dependency from database literature to compute the MI based ranking of attributes. This work makes several contributions in this regard.

Contribution (1) - Connecting Functional Dependency with MI Approximation. We connect the database concept of Functional Dependency (FD) to estimate MI-based feature ranking. An attribute will not have exact one-to-one mapping with target variable in a real dataset and hence perfect FD merely exists. We investigate how this imperfection can be effectively quantified and correlated with MI. **Contribution (2)- A New measure to approximate MI.** We investigate *Approximate Functional Dependency* (AFD) from literature (Kivinen & Mannila, 1995; Dalkilic & Roberston, 2000; Giannella & Robertson, 2004; Mandros et al., 2020; Liu et al., 2010; Lee, 1987) and find deficiency of one popular measure $G3 - error$ (Kivinen & Mannila, 1995) to approximate MI based feature ranking. We propose a new measure *Attribute Average Conflict* (Section 4) that effectively approximates MI, while being much faster computationally. **Contribution (3)- Experimental Evaluations** We perform extensive evaluations (Section 5) using multiple synthetic and real world datasets (with millions of records and hundreds of attributes) and compare our solutions with multiple methods, including state-of-the-art solution (Chen & Wang, 2021). Our experimental results convincingly corroborate the superiority of our proposed approach as, we *always* achieve the exact top-k attributes considering precision and ndcg-score, while being $2x$ faster than exact MI calculation. Adaptive uniform sampling (Chen & Wang, 2021) ends up consuming the entire dataset and turns out to be considerably slower ($4 - 7x$) than us. We are also $1.3x$ faster than uniform random

sampling with better precision and ndcg-score. Similar observation holds for real world data that shows that even though our proposed method brings some approximation in the top-k order, the produced attributes still have highly comparable MI with the exact calculation, while being faster than the exact calculation and other comparable methods. **Contribution (4)- Efficient algorithm using bounds on proposed measure** We investigate the upper and lower bounds of the proposed measure *Attribute Average Conflict* (*Aac*) (Section 6) by first exploring the maximum and minimum value of *Aac* for an attribute. First we show that a loose upper and lower bound can be derived by using only the domain size information of attributes. Then we derive tighter bounds by considering a setting where we have prior information of attribute (X) and target (Z) value frequency. This setup is possible for relational databases where the data dictionary, or metadata can be used to get the marginal frequency of attributes and target variable. Our proposed approach is able to use marginal frequency of attribute and target variable to find the possible arrangement that yields the maximum and minimum value of *Aac*. We show that finding the minimum value of *Aac* is NP-Complete and use a greedy approach that works well in practice. We develop the criteria for minimum number of records needed to scan using the established upper and lower bounds. Then we develop an efficient algorithm that can select the top-k attributes using the proposed bounds of the new measure in a single pass without scanning the full dataset. We show the accuracy and effectiveness of this improved algorithm by experimental evaluation on real datasets. Our experiment illustrates that for a real world dataset with 299K records and 28 attributes, our proposed bound based algorithm scans only 65% of the records in a single pass to find the top-10 attributes with perfect accuracy.

2 Related works

Feature selection Feature selection is an important topic for machine learning and data mining discipline, where a subset of input variables is computed to efficiently describe the input data while reducing effects from noise or irrelevant variables ensuring good prediction results (Chandrashekar & Sahin, 2014). A plethora of work exists for various feature selection techniques. A review of feature selection methods with application can be found in Guyon and Elisseeff (2003) and Jović et al. (2015). Various feature selection methods can be broadly categorized under - Wrapper, Filter, and Embedded methods. The performance of the predictor is used as the feature selection criterion in wrapper methods (Chandrashekar and Sahin, 2014). In case of Filter methods, features are ranked using some criteria and highly ranked features are selected and applied to the predictor. Embedded methods include variable selection as part of the training process without splitting the data into training and test sets (Chandrashekar & Sahin, 2014). Variable ranking techniques are used as the principal criteria for selection of variable in Filter methods. Ranking methods are simple and effective for Filter techniques. Variables are scored through a suitable ranking criterion and only those variables are retained that are at least equal to a predefined threshold. As the ranking methods are applied before classification to filter out the less relevant variables, hence they are identified as Filter methods. Two popular criteria for ranking methods are Correlation and Mutual Information (MI). Correlation only detects linear dependencies between variable and target, whereas MI is able to

capture non-linear relationship as well (Chandrashekar and Sahin, 2014). MI based feature selection is a Filter method where features are ranked according to MI score with respect to the class variable. Advantage of MI is that it does not rely on learning algorithm and hence avoids overfitting. One of the drawbacks of this approach is that the selected subset might not be optimal and the variables in the subset can be highly correlated (Chandrashekar & Sahin, 2014). Vergara and Estévez (2014) presents a review of feature selection methods based on MI. Main focus of these works are on effectively using MI to get more accurate top-k features while addressing the issue of relevancy and redundancy, but our aim is to devise a faster method without actually calculating MI . MI has been used for feature engineering tasks as well (Salam et al., 2019). Recent works have proposed sampling based techniques for selecting top-k features using empirical MI (Wang & Ding, 2019; Chen & Wang, 2021). Chen and Wang (2021) focuses on improving speed of the calculation compared to Wang and Ding (2019), but they still compute MI on samples. We develop a new measure for faster computation of the top-k features without calculating MI which differentiates our work from these works.

Approximate functional dependency (AFD) AFD concept is related to FD which has been widely used by researchers and practitioners in database community for schema design. An initial work on approximately inferring functional dependencies using sampling is provided in Kivinen and Mannila (1995), where some important definitions on error measurement for AFD is introduced that stand as a reference for later works. Some alternative techniques on inferring full and approximate functional dependency have been proposed in later works (Novelli & Cicchetti, 2001; Lopes et al., 2000; Huhtala et al., 1999). A notion of information dependency has been introduced in Dalkilic and Roberston (2000) which is used to explore AFD. (Dalkilic & Roberston, 2000) provides new definition for measuring AFD and shows how their measure compares with the one provided by the earlier work of Kivinen and Mannila (1995). Methods for quantifying approximation degree of AFD have been proposed in Giannella and Robertson (2004) based on prior work of Dalkilic and Roberston (2000). A recent work proposed method for discovering dependencies using reliable MI (Mandros et al., 2020). Main focus of these line of works are on devising fast and comprehensive methods for finding AFD in the dataset. The main difference between these works and ours is that we take the idea of AFD and devise a new measure to compute MI based top-k features.

3 Preliminaries and definitions

Example 1 We describe a toy example in Table 1 with 10 records and 2 attributes (predictors) - *Ethnicity* and *Age* <30 and one Boolean class label *PlaysBasketball*. The *recordId* column contains the unique identifier for the record. For the simplicity of exposition, we consider binary predictors and class labels. Table 2 lists the measures for this example which will be discussed later in this section.

Table 1 Running Example 1

record Id	Ethnicity	Age <30	Plays BasketBall
r1	Black	Yes	Yes
r2	Black	Yes	Yes
r3	Black	Yes	Yes
r4	Black	Yes	No
r5	Black	Yes	No
r6	Black	Yes	No
r7	Black	No	No
r8	Black	No	No
r9	Asian	No	No
r10	Asian	No	No

Table 2 Measures for Example 1

	Ethnicity	Age < 30
G3-error	3	3
Aac	2.4	1.8
MI	0.1916	0.2812

3.1 Notations and prior definitions

Attributes, records, and target variable/class label A given dataset $\mathcal{D}$ is comprised of a set $\mathcal{A}$ of m categorical attributes $\{X_1,X_2,...,X_m\}$ or features and n records, as well as an additional class label/target variable (column) Z. The target variable Z contains the class label of an instance. Using Example 1, $n = 10$, $m = 2$, $\mathcal{A} = \{Ethnicity, Age < 30\}$, $X_1 = Ethnicity$, $X_2 = Age < 30$, $Z = PlaysBasketball$.

Mutual Information (MI) (Cover & Thomas, 1991) Mutual Information (MI) captures information theoretic "correlation" Li (1990) between two random variables that quantifies the amount of information obtained about one through the other. When X and Z are discrete [1],

$MI(X,Z)$ is defined as follows:

$$MI(X,Z) = \sum_{x \in X} \sum_{z \in Z} p(x,z) \log \frac{p(x,z)}{p(x)p(z)} \tag{1}$$

where $p(x,z)$ is the joint probability distribution function of X and Z, and $p(x)$ and $p(z)$ are the marginal probability distribution functions of X and Z respectively. We use $MI(X)$

[1] We consider the numeric variables are appropriately discretized, when needed

for brevity instead of $MI(X,Z)$. Using entropy of Z denoted as $H(Z)$ and conditional entropy between Z and X denoted as $H(Z|X)$ (Cover and Thomas, 1991), $MI(Z,X)$ is defined as

$$MI(Z, X) = H(Z) - H(Z \mid X) \tag{2}$$

MI is symmetric, that is $MI(X,Z) = MI(Z,X)$ (Cover & Thomas, 1991). In Example 1, $MI(Ethnicity,PlaysBasketball) = 0.1916$, and $MI(Age < 30,PlaysBasketball) = 0.2812$. Next, we define 3 key terms from Database literature that have been used for schema design.

Functional Dependency (FD) (Elmasri & Navathe, 2011) A functional dependency between X and Z, denoted by $X \rightarrow Z$, is a constraint on the possible tuples that can form a relation state r over $\mathcal{A}$. The constraint is that for any two tuples t_1 and t_2 in r that have $t_1[X] = t_2[X]$, they must also have $t_1[Z] = t_2[Z]$. In Example 1, *Ethnicity* = *Black* is associated with *PlaysBasketBall* = *Yes* in r1, r2, r3, but the same value for *Ethnicity* is associated with *PlaysBasketBall* = *No* in r4. Similarly, *Age* < 30 = *Yes* is associated with different values of *PlaysBasketBall* in different tuples. Hence, there is no FD from either *Ethnicity* or *Age* < 30 to *PlaysBasketball*.

Approximate Functional Dependency (AFD) (Huhtala et al., 1999) Given an error threshold ϵ, $0 \leq \epsilon \leq 1$, $X \rightarrow Z$ is an Approximate Function Dependency (AFD) if and only if $e(X \rightarrow Z)$ is at most ϵ. Here $e(X \rightarrow Z) = \min\{|s| \mid s \subseteq r \text{ and } X \rightarrow Z \text{ holds in } r \setminus s\} / |r|$. This definition is based on the minimum number of tuples ($|s|$) needed to be removed from relation r for $X \rightarrow Z$ to hold in r (Huhtala et al., 1999). In Example 1, each value in *Ethnicity* has one-to-one mapping with *PlaysBasketball* for records r4 through r10, but not for all the records as r1, r2, r3 violate this rule. We say AFD holds from *Ethnicity* to *PlaysBasketball*. Similarly, AFD holds from *Age* < 30 to *PlaysBasketball*.

G3-error (Kivinen & Mannila, 1995) The number of tuples need to be deleted from relation r to achieve FD is defined as G3-error (Kivinen & Mannila, 1995).

$$G3 - error(X \rightarrow Z, r) = |r| - max\{|s| \mid s \subseteq r, s \vDash X \rightarrow Z\}$$

In Example 1, $G3 - error(Ethnicity \rightarrow PlaysBasketball, r) = 3$, $G3 - error(Age < 30 \rightarrow PlaysBasketball, r) = 3$.

3.2 Problem statement

Our aim is to select top-k attributes from m given attributes ($k \leq m$) in decreasing order of MI wrt binary target Z , such that the MI of k-th attribute will not be less than the MI of any of the $m - k$ attributes. The problem statement is formally defined below.

Given a dataset $\mathcal{D}$ containing a set $\mathcal{A}$ of m attributes $\{X_1, .., X_m\}$ and a target variable $Z \in \{0, 1\}$, select top-k attributes $\{X_1, .., X_k\}$ such that $MI(X_i, Z) \geq MI(X_{i+1}, Z) \forall i \in \{1, k-1\}$ and $MI(X_k, Z) \geq MI(X_j, Z) \forall j \in \{k+1, m\}$.

4 Developing a new measure

In this section we discuss various steps in MI exact calculation, connecting the AFD measure for MI approximation, and finally propose our new measure to efficiently and effectively approximate MI.

4.1 MI calculation steps

Taking a close look at equation (1), we observe that there are three components to calculate $MI(X,Z)$ - joint probability distribution $p(x,z)$, marginal probability distribution of distinct values of X and Z - $p(x)$ and $p(z)$ where $x \in X$ and $z \in Z$. We note that $p(z)$ can be computed once and reused later. We need to compute the co-occurrence count of distinct (x,z) values along with the count of x values for each distinct x. In each of these steps one logarithm function is applied along with one division and two multiplication. We investigated the opportunity to reduce the number of operations in each step (i.e. for each (x,z) value combination). Our motivation is that, if we can avoid applying the logarithm function and approximate it with some other basic arithmetic operation, we may be able to speedup the process of MI calculation. During our investigation we observed that for two attributes X_1 and X_2, if most of the values of X_1 can determine a unique Z value, but most of the values of X_2 cannot do so, then $MI(X_1,Z)$ tends to be larger than $MI(X_2,Z)$. This intuitively aligns with the database concept of Functional Dependency (FD), and we proceed with investigating this connection.

4.2 Proposed measure: Attribute average conflict

There has been previous works in database and data mining community regarding various measures for Approximate Function Dependency (AFD) (Kivinen & Mannila, 1995; Dalkilic & Roberston, 2000; Giannella & Robertson, 2004). We define a new measure *Attribute Average Conflict* in such a way that it captures the degree of AFD as well as the weight of attribute value frequency that influences MI score of that attribute. As the MI formula (1) captures co-occurrence of (X,Z) by computing the joint distribution of (X,Z) in the numerator and has the marginal of X and Z in the denominator, by considering attribute value frequency in defining our new measure, we incorporate the effect of the frequency distribution present in MI calculation. In prior works on AFD, the term $G3 - error$ has been defined (see Section 3), which quantifies the number of changes required to attain FD, so smaller value of $G3 - error$ should indicate larger MI. But $G3 - error$ does not consider the weight of attribute value frequency and hence cannot approximate the MI based ranking of attributes correctly in many cases. We will explain one such scenario later in this section. Next we define some key terms which will be important ingredients of our final proposed measure *Attribute Average Conflict*.

Attribute Value Conflict (AV-conflict) For a specific attribute value $x_v \in X$, the minimum number of tuples where Z value need to be changed to establish one-to-one relationship with corresponding x_v is defined as $AV - conflict$. In Example 1, $AV - conflict(Ethnicity = Black) = 3$.

Attribute Value Average Conflict (AV-average-conflict) Multiplying $AV - conflict$ by the probability of that specific attribute value in the dataset yields $AV - average - conflict$ for that attribute.

$$AV - average - conflict(x_v) = AV - conflict(x_v) \times \frac{n_{x_v}}{n} \tag{3}$$

Here n_{x_v} denotes the number of records where X has value v. In Example 1, $AV - average - conflict(Ethnicity = Black) = 3 \times 0.8 = 2.4$;

Attribute Average Conflict (Aac) The sum of AV-average-conflict for all values of an attribute is the *Attribute Average Conflict* (*Aac*) for that attribute.

$$Aac(X) = \sum_{x_v \in X} AV - average - conflict(x_v) \tag{4}$$

In Example 1, $Aac(Ethnicity) = 2.4 + 0 = 2.4$; $Aac(Age < 30) = 1.8 + 0 = 1.8$. $G3 - error$ is not able to capture the weight of co-occurrence of attribute and target value in the dataset, and hence the score remains same for *Ethnicity* and $Age < 30$ although the MI is different. On the other hand, *Aac* decreases when MI increases. *Aac* enables us to overcome the deficit of $G3 - error$ in capturing MI relationship between attributes. We investigated this case for different highly skewed data distributions, and found that *Aac* holds inverse relationship with *MI*, that is, for $\{X_1, X_2\} \in \mathcal{A}$, $MI(X_1) > MI(X_2) \Rightarrow Aac(X_1) < Aac(X_2)$ and vice versa.

4.3 Implications of using proposed measure

Using proposed measure *Aac*, we can skip computing the logarithm function and reduce the number of arithmetic operations as discussed in Section 4.1. *Aac* provides us with the answer to the **Top-K()** query faster than the exact MI based method.

As illustrated in Algorithm 1, *Aac* for each attribute $X_i \in \mathcal{A}$ is calculated, and attributes are sorted in ascending order of *Aac*. The first k attributes in this sorted list will be the top-k attributes based on highest MI. Asymptotically, both our method and MI-based feature selection method runs in $\mathcal{O}(mn)$ times, but empirically, our proposed method is faster than MI-based method as we avoid the step of computing logarithm and division. We perform extensive experiments and present our finding in Section 5.

inputs: set of attributes $\mathcal{A}$, target variable Z, k
output: Top-k attributes
S = Compute $Aac(X_i)$ for $X_i \in \mathcal{A}$
Sort S according to ascending score of $Aac(X_i)$
top-k $\leftarrow$ first k attributes from S
Return top-k

Algorithm 1 Algorithm `topK-Aac`.

4.4 Monte-Carlo simulation

We conduct a Monte-Carlo simulation (Mooney, 1997) that demonstrates for any two attributes X_1 and X_2 and a target variable Z, how $Aac(X_1)$ and $Aac(X_2)$ relate to $MI(X_1)$ and $MI(X_2)$ considering all probable attribute value combinations of X_1, X_2, Z.

4.4.1 Model setup

Let $\mathscr{L}$ denote the number of combinations satisfying all possible combinations of X_1, X_2, and Z. Given two binary attributes X_1, X_2 and a binary target variable Z, $\mathscr{L} = 8$, as there are 8 possible attribute value combinations involving these three. Let n_i denote the fraction of the respective attribute value combinations in n records, where n_1 corresponds to $X_1 = 0, X_2 = 0, Z = 0$, n_2 corresponds to $X_1 = 0, X_2 = 0, Z = 1$, to n_8 corresponding to $X_1 = 1, X_2 = 1, Z = 1$. Let, t_1,t_2 denote the fraction of records that map a specific value of an attribute to $Z = 0$,$Z = 1$ respectively. If $t_1 = 0$, or $t_2 = 0$, then $AV - average - conflict(X_1 = 0) = 0$. If $t_1 \leq t_2$, then $AV - average - conflict(X_1 = 0) = t_1 \times \frac{t_1 + t_2}{n} = t_1 \times (t_1 + t_2)$ (here, $n = 1$), otherwise $AV - average - conflict(X_1 = 0) = t_2 \times (t_1 + t_2)$. $Aac(X_1)$ can be derived by summing these values. Similarly, $G3 - error$ could also be calculated.

4.4.2 Generating all possible probability distributions

We assign probability values to each n_i such that $\sum n_i = 1$. We deliberately assign zero values for some fractions of n to mimic the scenario that not all attribute value combinations appear in real datasets. This process runs in a loop, where we systematically vary fraction of zero values (from 10% to 90% to demonstrate sparsity/skewness). For each run, with a specific zero fraction value $y\%$, we consider $\lceil \mathscr{L} \times y\% \rceil$ of attribute value combinations to be 0 that are uniform randomly chosen. For the remaining combinations, we uniform randomly assign real numbers between [0,1] such that the non-zero combinations add up to 1. For each run, $MI(X_1) > MI(X_2) \Rightarrow Aac(X_1) < Aac(X_2)$, or vice versa. If that happens, then we count it as *support*, otherwise count it as *contradiction*. We repeat each run 100,000 times and calculate the percentage of *support* and *contradiction* of *Aac* and $G3 - Error$.

4.4.3 Simulation results

Table 3 illustrates the simulation results for binary attributes. Here, $y = 0.6$ means that 60% of the possible attribute value combinations are assigned 0 probability and the probability assignment for rest of the 40% combinations add up to 1. We observe that *Aac* performs better than $G3 - error$ for y from 0.1 to 0.6, and shows similar performance for 0.7 to 0.9 ; For smaller y, *Aac* performs significantly better than $G3 - error$. For binary attributes the lowest *support* for *Aac* is 85.4% whereas the highest *support* is 100%. Table 4 illustrates the simulation results for various domain sizes where y is fixed at 0.3. Consistently, *Aac* outperforms $G3 - Error$ for domain size < 10, whereas $G3 - Error$ is slightly better than *Aac* for domain size ≥ 10. This exercise is another evidence that demonstrates the effectiveness of our proposed measure. We conduct experimental evaluation on real world datasets that support this observation as well.

Table 3 Simulation results for binary values

y	G3-error support %	G3-error contradiction %	Aac support %	Aac contradiction %
0.1	66.568	33.432	85.493	14.507
0.2	64.1	35.9	85.069	14.931
0.3	62.427	37.573	86.755	13.245
0.4	66.652	33.348	93.836	6.164
0.5	66.56	33.44	93.88	6.12
0.6	81.01	18.99	100	0
0.7	100	0	100	0
0.8	100	0	100	0
0.9	100	0	100	0

Table 4 Simulation results varying domain size; $y = 0.3$

Domain size	G3-error support %	G3-error contradiction %	Aac support %	Aac contradiction %
2	62.427	37.573	86.755	13.245
3	81.892	18.108	90.397	9.603
4	88.172	11.828	92.054	7.946
5	90.231	9.769	92.694	7.306
6	91.495	8.505	93.187	6.813
7	92.261	7.739	93.338	6.662
8	92.769	7.231	93.373	6.627
9	93.224	6.776	93.359	6.641
10	93.535	6.465	93.44	6.56
15	94.414	5.586	93.429	6.571
20	94.88	5.12	93.212	6.788
25	95.052	4.948	93.171	6.829

5 Experiments

All the experiments are conducted on a 8-core 3.06GHZ machine with 16 GB RAM. We use Python 3 to implement the algorithms in the experiments and the numbers are presented as the average of 5 runs. The code and the data could be found in https://github.com/linsal26/aac-feature-selection. The goal of our experimental evaluation is to effectively answer the following questions.

- How does our proposed method compare with the baselines both qualitatively and efficiency wise by varying n, m, k, and skew λ in data distribution.
- How does our proposed method compare with the baselines both qualitatively and efficiency wise considering real datasets.

5.1 Experimental setup

Datasets

- Synthetic data: We generate highly skewed binary datasets for experimental evaluation. Here a dataset is considered highly skewed if a small number of attribute values perfectly correlate with target value and all the attributes show some degree of this skewness. For example, in a dataset of 1 Million records, if $X_1 = 1$ appers in 1 record, $X_2 = 1$ appears in 2 records,..., $X_{50} = 1$ appears in 50 records and all of these records correlate with $Z = 1$, then we consider this as a case of highly skewed dataset. We use a fixed probability distribution for target Z ($p(Z = 0) = 0.000005$, $p(Z = 1) = 0.999995$) and vary the distribution of attributes X_i to generate highly skewed dataset. We use λ to denote the skew. $\lambda = 0.999999$ indicates that $p(X_1 = 0) = 0.999999$, $p(X_2 = 0) = 0.999998$ and so on. We decrease the probability of 0-value of an attribute by 0.000001 from the previous attribute and continue in this fashion for all the attributes. An example distribution for $\lambda = 0.999999$ is provided in Table 5. Here the column $p(X = 0)$ and $p(X = 1)$ indicate the probability of 0 and 1 for the attribute respectively.
- Real world data: We report our findings on three real world datasets from UCI [2], Keel [3] and OpenML[4] repository. The Datasets are summarized in Table 6.
  - Census Income (Dua & Graff, 2017): US Census dataset containing 40 attributes including integer and categorical types out of which 28 categorical attributes are used. The target has binary class label indicating whether a person has income $> 50K$ or not.

Table 5 Synthetic Dataset

	$p(X = 0)$	$p(X = 1)$
X_1	0.999999	0.000001
X_2	0.999998	0.000002
X_3	0.999997	0.000003
...	...	...
X_{50}	0.99995	0.00005

Table 6 Real Datasets

Dataset	*n*	*m*
Census Income	299,285	28
Kick Car	72,983	17
Connect-4	67,557	42

[2] https://archive.ics.uci.edu/ml/datasets.php

[3] https://sci2s.ugr.es/keel/datasets.php

[4] https://www.openml.org/home

- Kick Car: This dataset contains attributes for various cars from auction and the target class is either kick (bad buy) or not. There are 33 attributes in the original dataset out of which 17 categorical attributes are used in this experiment. Data source - https://www.openml.org/d/41162
- Connect-4 (Dua & Graff, 2017): This dataset contains all legal positions in the game of connect-4 for a 6x7 grid, in which neither player has won yet, and in which the next move is not forced. Original dataset has 42 categorical attributes all of which are used in the experiment. The target has 3 labels in original data - win, loss and draw, which is converted to 2 labels - won (1), not won (0). Data source - https://sci2s.ugr.es/keel/dataset.php?cod=193

Implemented baselines Our proposed measure in Section 4 *Attribute Average Conflict*, *Aac* is compared against the baseline, and 3 other implemented algorithms as follows.

- *MI-based method, MI.* We implement MI-based method for feature selection that computes exact MI score for each attribute in the dataset and provides top-k attributes based on the highest MI score. This is used as the baseline method to compare against for both performance and runtime improvement.
- *Uniform random sampling, Urs*. We implement a uniform random sampling based method that computes MI of attributes on sampled data after taking uniform random sample, and returns top-k attributes based on this MI. We use 65% of the data as the sample size to achieve high precision for the highly skewed synthetic data.
- *Adaptive sampling based method, Swope.* We implement the *Swope* method proposed in (Chen & Wang, 2021) to find top-k attributes using MI. The *Swope* algorithm uses random sampling at its core and adaptively expands the sample size until certain bounds are met.
- *G3-error based method, G3.* The $G3-error$ based method is implemented by using $G3-error$ score instead of *Aac* in Algorithm 1.

Performance measures For computing the accuracy of the proposed approach, we present precision (Han et al., 2011), ndcg-score (Baeza-Yates et al., 1999), and the total MI ($\sum_{i=1}^{k} MI(X_i)$). Efficiency is presented as speedup. Given two algorithms A and B (where B is the baseline), $Speedup(A)$ wrt B is computed as, $\frac{RunningTime(B)}{RunningTime(A)}$. For example, if running time of *MI* and *Aac* is 5s, and 2s respectively, then speedup of *Aac* wrt *MI* is $\frac{5}{2} = 2.5$.

Default parameters Unless otherwise stated, n, m, k, skew parameter λ is set to $n = 10^6$, $m = 50$, $k = 10$, $\lambda = 0.999999$. For *Swope* algorithm, we set $\epsilon = 0.5$ (Chen & Wang, 2021) , and for *Urs* sample size $= 0.65n$. Unless otherwise stated, speed up is presented wrt baseline *MI*.

5.2 Summary of results

Our first and foremost observation is, our proposed method *Aac* and *Swope* achieve perfect precision and ndcg-score considering all settings for the highly skewed synthetic data. However, *Aac* is $4x - 6x$ faster than *Swope*. *Urs* has lower precision for

some k, and ndcg-score is not perfect. $G3$ displays worst precision and ndcg-score. Considering speedup, *Aac* is $2x$ faster than *MI*, $1.3x$ faster than *Urs*, and has similar speedup compared to $G3$. These observations conclusively corroborate the superiority of *Aac* compared to all the baselines. Figures 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 and 12 illustrate the experimental result for synthetic data.

Similar observation holds in the real data experiments. The experimental results are illustrated in Figs. 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23 and 24. *Aac* does not achieve perfect precision and ndcg-score, but achieves highly comparable total-mi score compared to exact baseline *MI*. *Swope* shows perfect precision, ndcg-score and total-mi for all the datasets in the experiment, but is much slower compared to

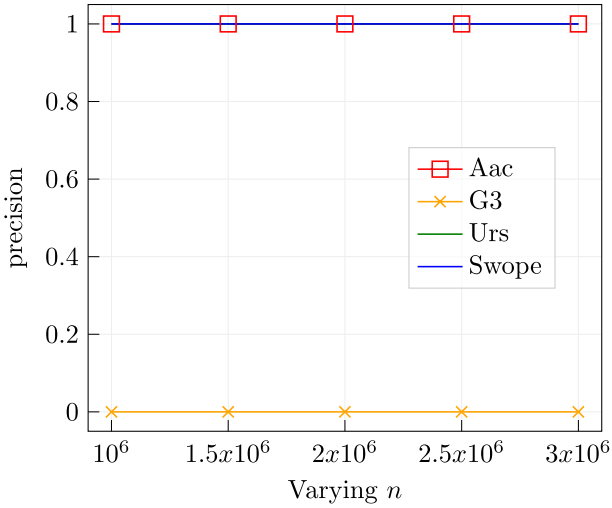


Fig. 1 Precision; m = 50, k = 10

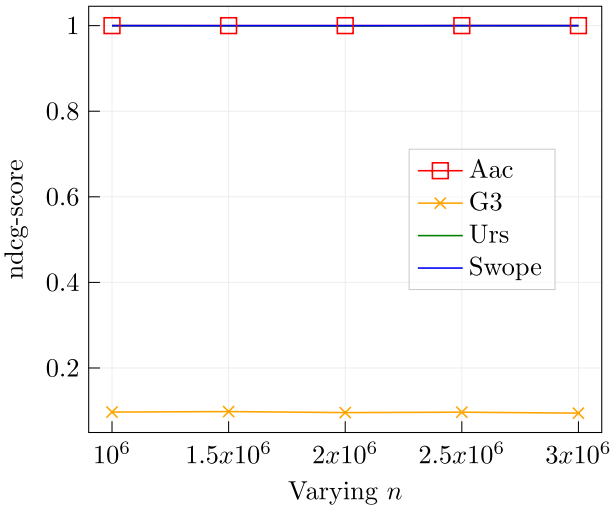


Fig. 2 Ndcg-score; m = 50, k = 10

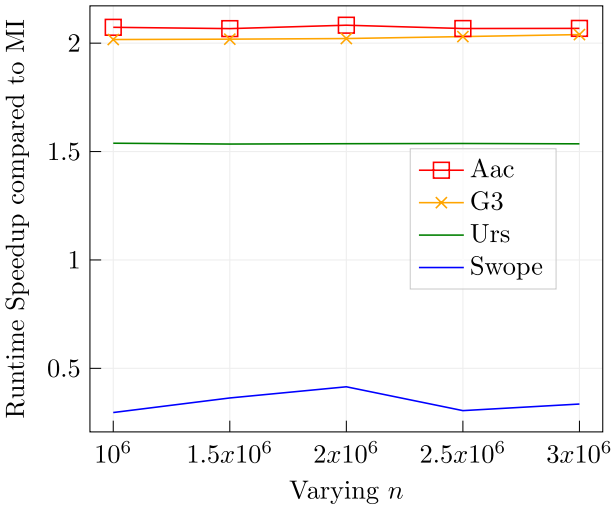


Fig. 3 Speedup; m = 50, k = 10

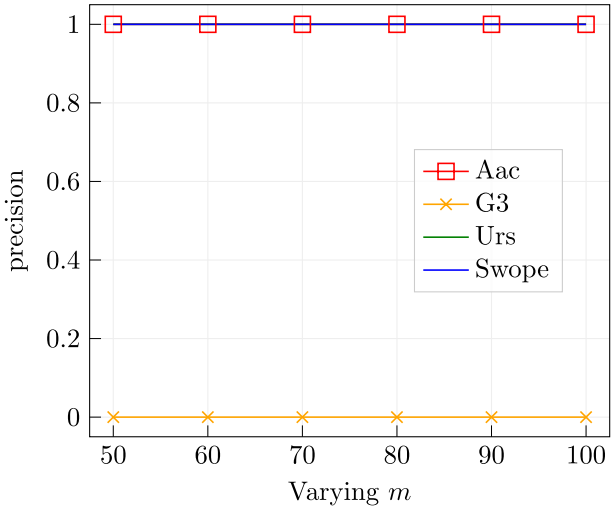


Fig. 4 Precision; n = 10^6, k = 10

Aac. This is because *Swope* uses exact equation for computing *MI* whereas *Aac* is approximating *MI* based ranking. *Urs* has better precision and ndcg-score, but slower than *Aac*. Total MI of *Aac* stays close to baseline *MI*. *G3* shows inferior precision, ndcg-score and Total MI compared to *Aac* for datasets Census Income and Connect-4 , where most of the attributes (> 50%) have small domain size (< 10). But for dataset

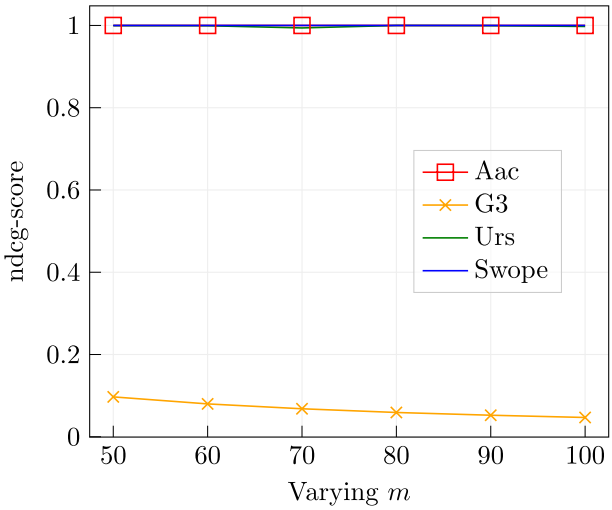


Fig. 5 Ndcg-score; n = 10^6, k = 10

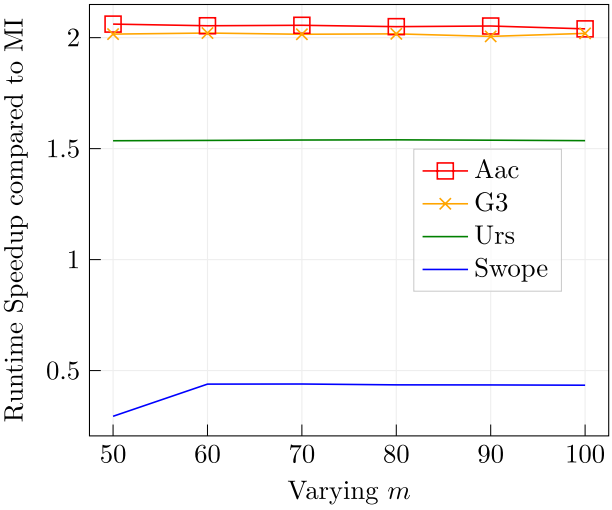


Fig. 6 Speedup; n = 10^6, k = 10

where most of the attributes (> 50%) have larger domain size (≥ 10), *G*3 achieves slightly better precision and ndcg-score than *Aac* which is observed for Kick car dataset. For speedup comparison, on average *Aac* is 2*x* faster than *MI*, 1.3*x* faster than *Urs* and 6.7*x* faster than *Swope* across all 3 datasets. Thus, *Aac* turns out to be the unanimous choice considering both accuracy and efficiency.

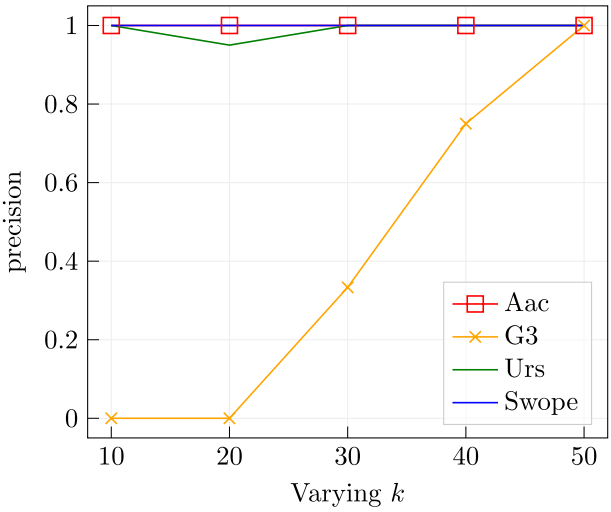


Fig. 7 Precision; n = 10^6, m = 50

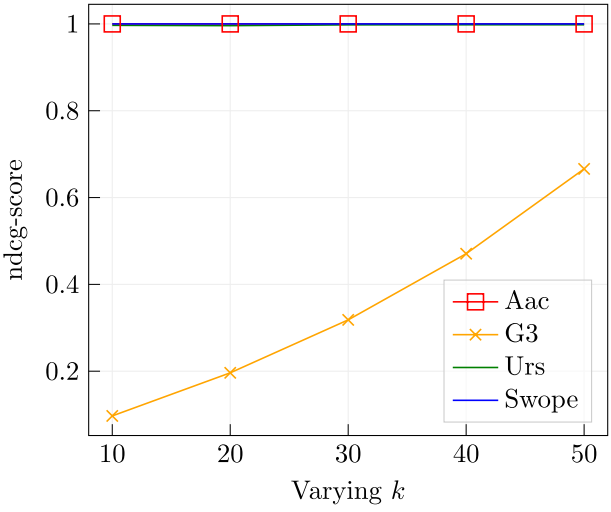


Fig. 8 Ndcg-score; n = 10^6, m = 50

6 Upper and lower bounds of new measure

We investigate the upper and lower bounds of the proposed new measure *Aac* by first exploring the maximum and minimum value of *Aac* for an attribute. A loose upper and lower bound can be derived by using only the domain size information of attributes. Tighter bounds can be achieved by considering a setting where we have prior information of attribute and target value frequency. This setup is possible for relational

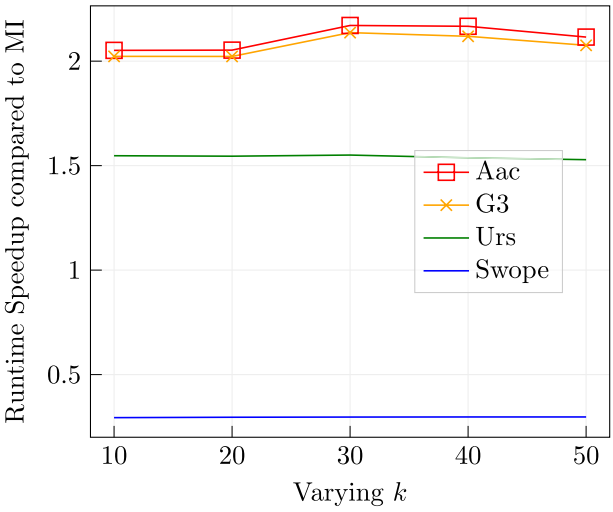


Fig. 9 Speedup; n = 10^6, m = 50

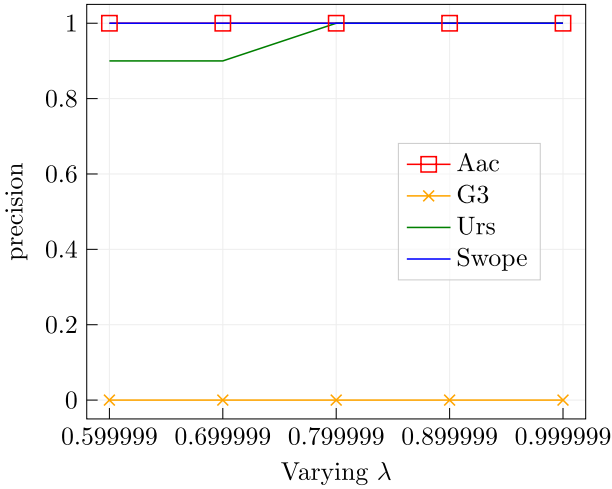


Fig. 10 Precision; n = 10^6, m = 50,k = 10

databases where the data dictionary, or metadata can be used to get the marginal frequencies of attributes and target variable. Table 7 summarizes the notations used to derive the bounds. We first investigate when maximum and minimum value of *Aac* can be achieved for an attribute *X*. At the beginning when no records have been read, *Aac*(*X*,0,1,*n*) denotes the *Aac* of attribute *X* for *n* unseen records. Theorems 1 and 2 provide the minimum and maximum values for *Aac*(*X*,0,1,*n*).

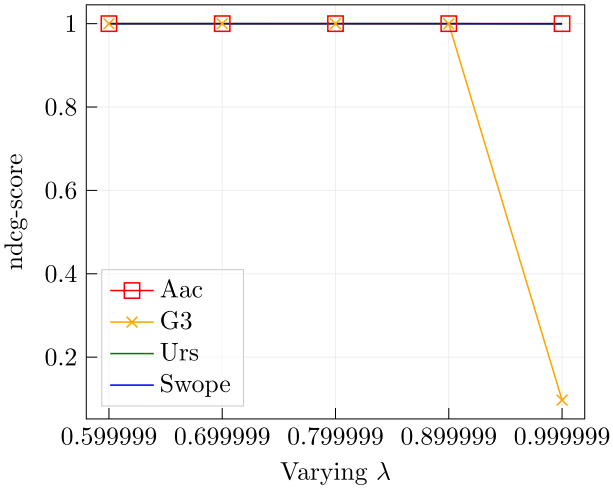


Fig. 11 Ndcg-score; n = 10^6,m = 50,k = 10

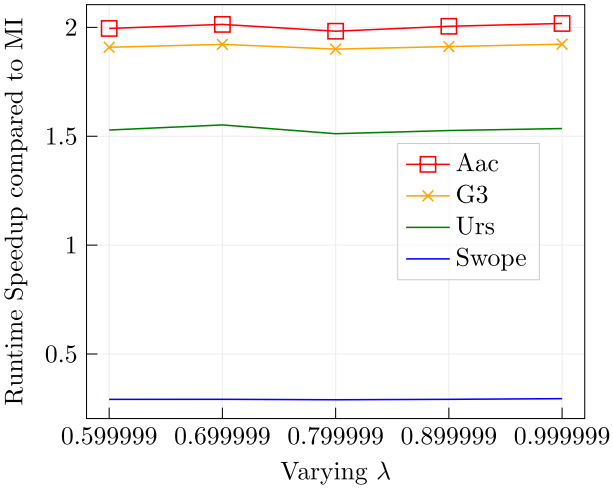


Fig. 12 Speedup; n = 10^6, m = 50,k = 10

Theorem 1 $Aac(X,0,1,n)_{min} = 0$

Proof if $X \rightarrow Z$, then there is a one-to-one mapping between distinct values of X with Z. Hence $AV - conflict(x_i) = 0\ \forall x_i \in X$, putting this value in equation (3), $AV - average - conflict(x_i) = 0$, and using this in equation (4), $Aac(X) = 0$.

Lemma 1 For $|Dom(X)| = 1$, $Aac(X,0,1,n)_{max}$ is achieved if $f_{x_i z_0} = \frac{n}{2}$

Proof In this case, we consider only one value for attribute X ($X = i$), denoted as x_i. So, $f_{x_i} = n$. If f_{x_i} has equal splits with $Z = 0$ and $Z = 1$, then $f_{x_i z_0} = \frac{n}{2}$. So, $Aac(X, 0, 1, n) = AV - average - conflict(x_i) = \frac{1}{2} f_{x_i} \times \frac{n}{n} = \frac{1}{2} f_{x_i}$. For any $0 < \epsilon < \frac{1}{2}$, other

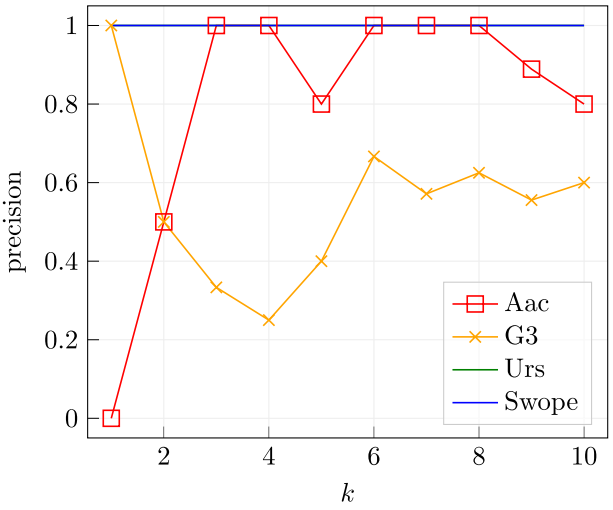


Fig. 13 Precision:Census Income

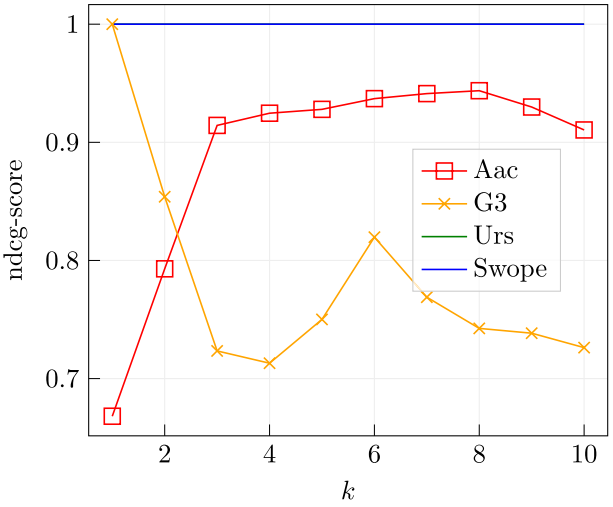


Fig. 14 Ndcg-score:Census Income

possible splits of f_{x_i} with Z are $(\frac{1}{2}+\epsilon)f_{x_i}$ and $(\frac{1}{2}-\epsilon)f_{x_i}$. For any such split, $Aac(X,0,1,n) = \left(\frac{1}{2}-\epsilon\right)f_{x_i} \times \frac{n}{n} = (\frac{1}{2}-\epsilon)f_{x_i} < \frac{1}{2}f_{x_i}$. Hence, $Aac(X,0,1,n)_{max}$ is acheived if $f_{x_i z_0} = \frac{n}{2}$.

Theorem 2 $Aac(X,0,1,n)_{max} = \frac{n}{2}$

Proof For $|Dom(X)| = 1$, $Aac(X,0,1,n)_{max} = \frac{1}{2}f_{x_i} = \frac{n}{2}$. For $|Dom(X)| = 2$ (let, $X = \{0,1\}$) , $Aac(X,0,1,n)_{max}$ is achieved if $AV - average - conflict(x_i)$ is maximum for each $x_i \in X$. Let, $AV - average - conflict(x_i)_{max}$ denote this maximum value for x_i.

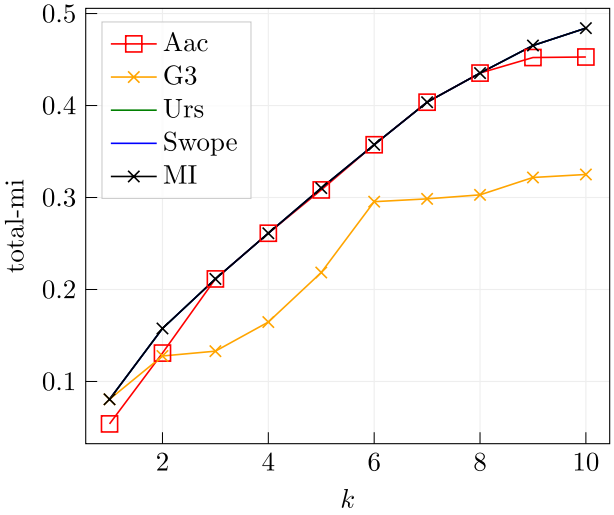


Fig. 15 Total-MI:Census Income

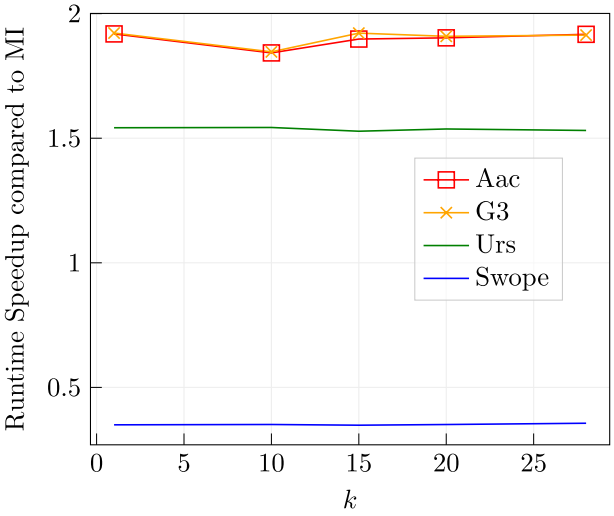


Fig. 16 Speedup:Census Income

Let, $f_{x_0} = n_1$, $f_{x_1} = n_2$. From Lemma 1, $AV - average - conflict(x_0)_{max} = \frac{n_1}{2} \times \frac{n_1}{n} = \frac{n_1^2}{2n}$,and $AV - average - conflict(x_1)_{max} = \frac{n_2}{2} \times \frac{n_2}{n} = \frac{n_2^2}{2n}$. So, $Aac(X, 0, 1, n)_{max} = \frac{n_1^2+n_2^2}{2n}$. Let, $\frac{n_1^2+n_2^2}{2n} > \frac{n}{2}$. There can be 2 cases. Case-(a) $n_1 = n_2 = \frac{n}{2}$. Then $\frac{n_1^2+n_2^2}{2n} = \frac{n^2/4+n^2/4}{2n} = \frac{n}{4} < \frac{n}{2}$ which is a contradiction. Case-(b) $n_1 = \frac{n}{2} + \epsilon$, $n_2 = \frac{n}{2} - \epsilon$, where $1 < \epsilon < \frac{n}{2}$. Then $\frac{n_1^2+n_2^2}{2n} = \frac{(\frac{n}{2}+\epsilon)^2}{2n} + \frac{(\frac{n}{2}-\epsilon)^2}{2n} = \frac{n^2-2(\frac{n^2}{4}-\epsilon^2)}{2n} = \frac{n}{2} - \frac{\frac{n^2}{4}-\epsilon^2}{n} < \frac{n}{2}$ which is a contradiction.

Fig. 17 Precision:Kick Car

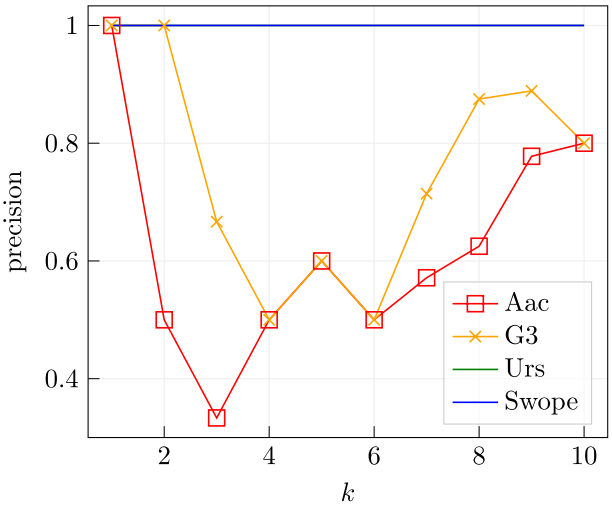
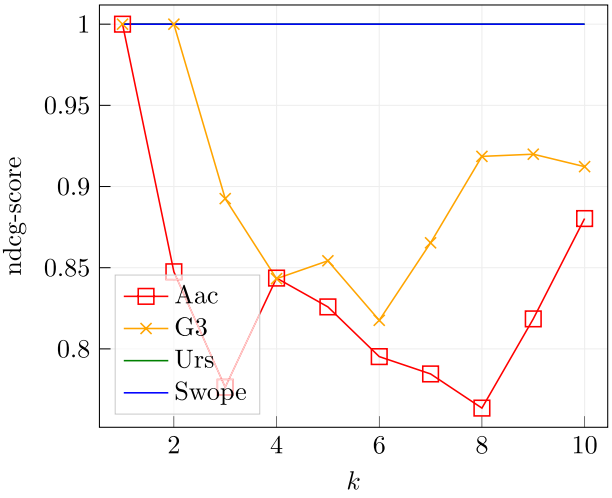


Fig. 18 Ndcg-score:Kick Car



Similarly, it can be shown for any $|Dom(X)| > 1$, $Aac(X,0,1,n)_{max}$ will be less than $Aac(X,0,1,n)_{max}$ for $|Dom(X)| = 1$. Hence $Aac(X, 0, 1, n)_{max} = \frac{n}{2}$.

Using the maximum and minimum values of Aac, we can define upper and lower bounds of Aac for an attribute X.

$$\begin{aligned} UB - Aac(X, l, 1, n) &= Aac(X, l, 1, l) + Aac(X, l, l + 1, n)_{max} \\ &= Aac(X, l, 1, l) + \tfrac{n-l}{2} \end{aligned} \tag{5}$$

Fig. 19 Total-MI: Kick Car

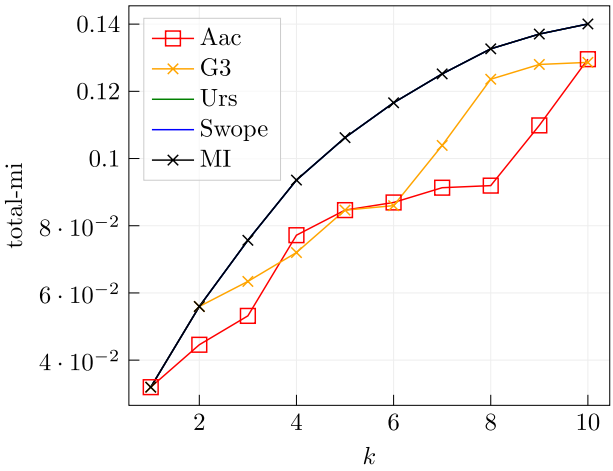
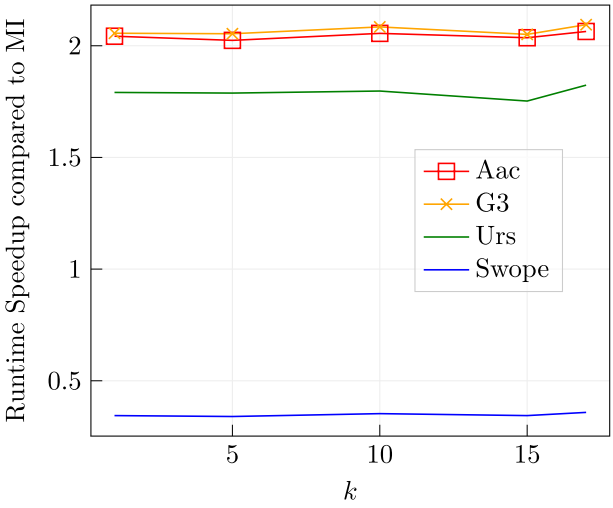


Fig. 20 Speedup:Kick Car



$$\begin{aligned} LB - Aac(X, l, 1, n) &= Aac(X, l, 1, l) + Aac(X, l, l+1, n)_{min} \\ &= Aac(X, l, 1, l) \end{aligned} \tag{6}$$

For two attributes X and Y, we can find whether $Aac(X) > Aac(Y)$ by reading l records, if the following relationship holds

$$\begin{aligned} LB - Aac(X, l, 1, n) &> UB - Aac(Y, l, 1, n) \\ \Rightarrow Aac(X, l, 1, l) &> Aac(Y, l, 1, l) + \tfrac{n-l}{2} \\ \Rightarrow \tfrac{n-l}{2} &< Aac(X, l, 1, l) - Aac(Y, l, 1, l) \end{aligned}$$

Hence, the minimum number of records l that need to be processed to determine the relationship between $Aac(X)$ and $Aac(Y)$ can be specified by equation (7)

Fig. 21 Precision: Connect-4

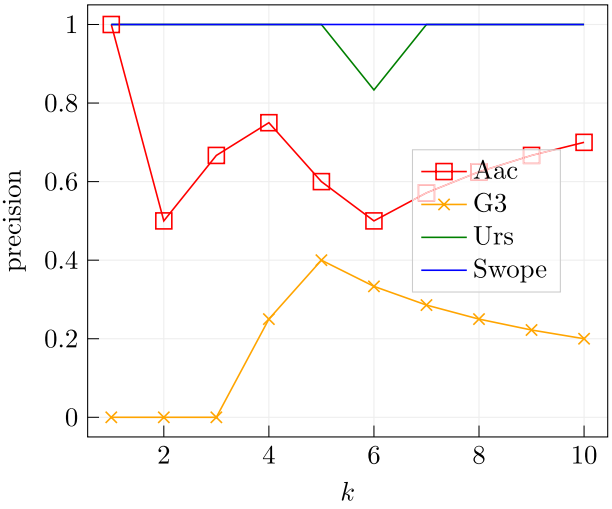
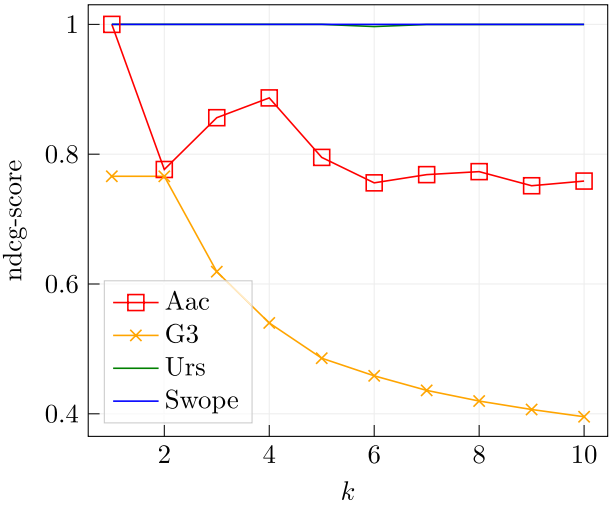


Fig. 22 Ndcg-score:Connect-4



$$l > n - 2[Aac(X, l, 1, l) - Aac(Y, l, 1, l)] \tag{7}$$

The upper and lower bounds in equations (5) and (6) are loose as they only consider the extreme cases for maximum and minimum values of *Aac* for unseen records. In realistic scenario, when dealing with a dataset with thousands of records, after reading a few hundred records, it is quite unlikely that there will be only one value for all the unseen records of an attribute X if $|Dom(X)| > 2$. Similarly, one-to-one mapping from X to Z for large number of unseen records will also be rare. Hence, the upper and lower bounds derived in (5) and (6) might not work well in practice, resulting in a large value for l close to n in equation (7). Next, we derive tighter upper and lower bounds to address this issue.

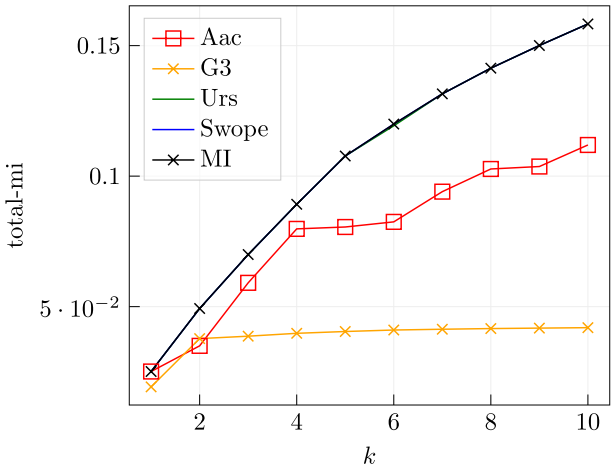


Fig. 23 Total-MI: Connect-4

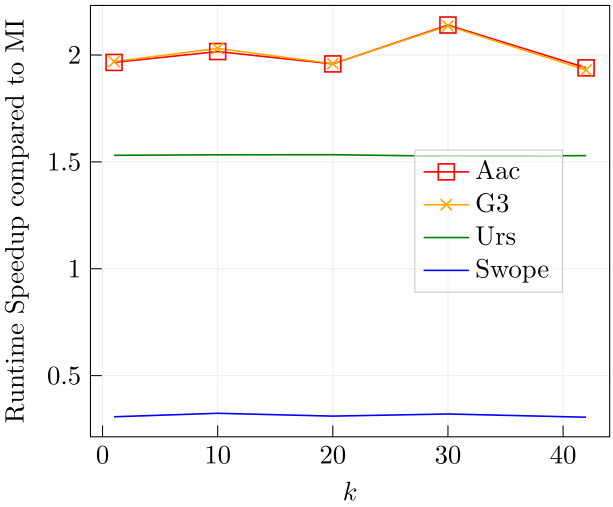


Fig. 24 Speedup: Connect-4

6.1 Tighter upper and lower bounds

In a relational database, the data dictionary contains metadata about the marginal frequency of attributes. We can use this information to devise possible arrangements of attribute-target pairs that can yield maximum and minimum value of *Aac* for an attribute. Our investigation finds that we do not need to have the joint distribution of attribute-target value pair for such arrangements, rather the marginal frequencies are sufficient for predicting the relevant joint distribution to find maximum and minimum value of *Aac*. This suits well with the relational database model as joint distribution of attribute value pairs are not stored in the metadata by default. Our proposed arrangement helps to derive tighter upper and lower bounds as we are not limited to using only the domain size information for $n - l$ unread records.

Table 7 Notations used for deriving bounds

Symbol	Explanation
n	Total number of records
l	Number of records already processed
$n-l$	Number of unseen records
$\|Dom(X)\|$	Number of distinct values for attribute X
f_{x_i}	Frequency of $X=i$
$f_{x_i z_0}$	Joint frequency of $(X=i, Z=0)$
S_{F_X}	Set of distinct value frequency of attribute X. For example, $S_{F_X}=\{f_{x_1}, f_{x_2}, ..., f_{x_s}\}$ where $\|Dom(X)\|=s$
Z	Target binary variable with values {0,1}
S_{F_Z}	Set of distinct value frequency of target Z. Here, $S_{F_Z}=\{f_{z_0}, f_{z_1}\}$ as $Z \in \{0, 1\}$
c_{max}	$max(f_{z_0}, f_{z_1})$
c_{min}	$min(f_{z_0}, f_{z_1})$
$Aac(X)_{max}/Aac(X)_{min}$	Maximum/Minimum value of Aac for attribute X
$Aac(X,l,s,e)$	$Aac(X)$ after processing l records starting from s-th and ending in e-th position
$Aac(X,l,l+1,n)_{max}$	Maximum attainable value of $Aac(X)$ for $n-l$ records
$Aac(X,l,l+1,n)_{min}$	Minimum attainable value of $Aac(X)$ for $n-l$ records
$UB-Aac(X,l,1,n)$	Upper Bound of $Aac(X)$ after processing l records
$LB-Aac(X,l,1,n)$	Lower Bound of $Aac(X)$ after processing l records

Let us consider attributes X, Y and target Z where $|Dom(X)|=s$, $|Dom(Y)|=t$, $|Dom(Z)|=2$; that is , X (i.e. Y) can take s (i.e. t) distinct values where s(i.e. t) ≥ 2, and $Z \in \{0,1\}$. Let f_{x_i}, f_{y_j} denote frequency of $X=i$, $Y=j$ respectively, so $\sum_{i=1}^{s} f_{x_i}=n$, $\sum_{i=1}^{t} f_{y_j}=n$. Let, f_{z_0}, f_{z_1} denote the frequency of $Z=0$ and $Z=1$ respectively; $c_{max}=max(f_{z_0}, f_{z_1})$, $c_{min}=max(f_{z_0}, f_{z_1})$.

Lemma 2 if $f_{x_1} > f_{x_2}$, then $AV-averge-conflict(x_1)_{max} > AV-averge-conflict(x_2)_{max}$

Proof From Lemma 1, $_{AV-averge-conflict(x_1)_{max}}=\frac{f_{x_1}}{2}\times\frac{f_{x_1}}{n}$, and $_{AV-averge-conflict(x_2)_{max}}=\frac{f_{x_2}}{2}\times\frac{f_{x_2}}{n}$. As $f_{x_1} > f_{x_2}$, hence $AV-averge-conflict(x_1)_{max} > AV-averge-conflict(x_2)_{max}$.

Using Lemma 1 and 2 we can derive the following equation for $Aac(X,l,l+1,n)_{max}$

$$Aac(X,l,l+1,n)_{max}=\begin{cases} c_{min}\times\frac{f_{x_1}}{n-l} & \text{if } c_{min}\leq\frac{f_{x_1}}{2} \\ \frac{\sum_{i=1}^{b-1}f_{x_i}^2}{2(n-l)}+(c_{min}-\frac{\sum_{i=1}^{b-1}f_{x_i}}{2})\times\frac{f_{x_b}}{n-l} & \text{otherwise} \end{cases} \quad (8)$$

where, $f_{x_1}\geq f_{x_2}\geq ...\geq f_{x_b}$, and $\sum_{i=1}^{b} f_{x_i}\geq 2c_{min}$.

Proof From Lemma 2, we can see that larger attribute value frequencies will contribute more to the Aac of an attribute. Hence, we arrange f_{x_i}'s in descending order and try to split each frequencies evenly with two Z values until we cover all the c_{min} values. Let, $c_{min}=f_{z_0}$. If $c_{min}\leq\frac{f_{x_1}}{2}$, then all other f_{x_i} $(2\leq i\leq s)$, will have one-to-one mapping with $Z=1$. The AV

$-$ *average* $-$ *conflict*(x_1) will be the maximum *Aac* for the attribute *X*. Figure 25 illustrates this case.

Otherwise, we can find one f_{x_b} such that $\sum_{i=1}^{b} \frac{f_{x_i}}{2} \geq c_{min} \Rightarrow \sum_{i=1}^{b} f_{x_i} \geq 2c_{min}$. Then $f_{x_1}, f_{x_2}, ..., f_{x_{b-1}}$ will have equal split with *Z* values for maximum $AV - average - conflict$, and $c_{min} - \frac{\sum_{i=1}^{b-1} f_{x_i}}{2}$ will be the minimum split value with *Z* for f_{x_b}. Figure 26 illustrates this scenario. Hence, $Aac(X, l, l+1, n)_{max} = \frac{\sum_{i=1}^{b-1} f_{x_i}^2}{2(n-l)} + (c_{min} - \frac{\sum_{i=1}^{b-1} f_{x_i}}{2}) \times \frac{f_{x_b}}{n-l}$.

Examples 2 and 3 illustrates the two cases for equation (8) for $n = 100$, $Aac(X,0,1,n)_{max} = Aac(X)_{max}$.

Example 2 Let, $S_{F_X} = \{40, 30, 20, 10\}$, $S_{F_Z} = \{15, 85\}$. Here, $c_{min} = 15, f_{x_1} = 40, f_{x_2} = 30, f_{x_3} = 20, f_{x_4} = 10$. $c_{min} < \frac{f_{x_1}}{2}$. So, $Aac(X)_{max} = c_{min} \times \frac{f_{x_1}}{n} = 15 \times \frac{40}{100} = 6$.

Example 3 Let, $S_{F_X} = \{40, 30, 20, 10\}$, $S_{F_Z} = \{30, 70\}$. Here, $c_{min} = 30, f_{x_1} = 40, f_{x_2} = 30, f_{x_3} = 20, f_{x_4} = 10$. $c_{min} > \frac{f_{x_1}}{2}$, and $f_{x_b} = 30$. Hence, $Aac(X)_{max} = \frac{\sum_{i=1}^{b-1} f_{x_i}^2}{2n} + (c_{min} - \frac{\sum_{i=1}^{b-1} f_{x_i}}{2}) \times \frac{f_{x_b}}{n} = \frac{40^2}{2 \times 100} + (30 - \frac{40}{2}) \times \frac{30}{100} = 11$

From Theorem 1 as $Aac(X)_{min} = 0$, we need to find an assignment of f_{x_i}'s with c_{min} and c_{max} such that there is a one-to-one mapping from each x_i to *Z* value. This is equivalent to finding an arrangement that requires checking all the f_{x_i} to see if there exists an *a* st $\sum_{i=1}^{a} f_{x_i} = c_{max}$. But the marginal frequency of attribute and target values may have such a distribution that a one-to-one mapping from *X* to *Z* is not possible. In that case, we want to find an arrangement that will ensure minimum $Aac(X) > 0$. To satisfy this

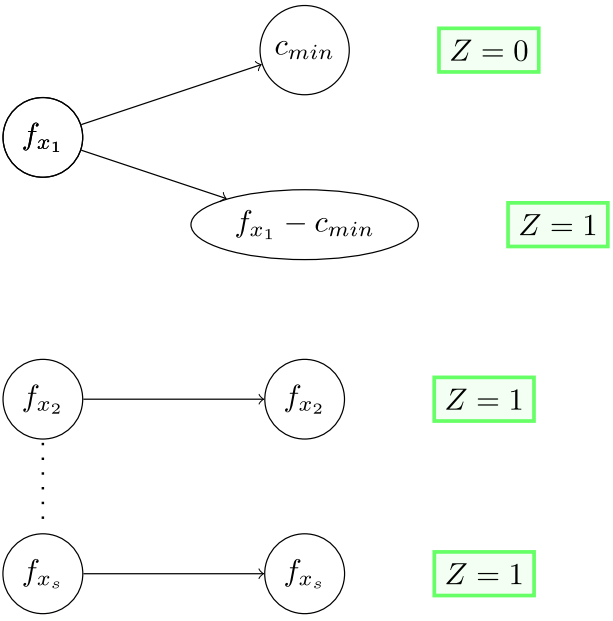


Fig. 25 Arrangement for $Aac(X)_{max}$ when $c_{min} \leq \frac{f_{x_1}}{2}$

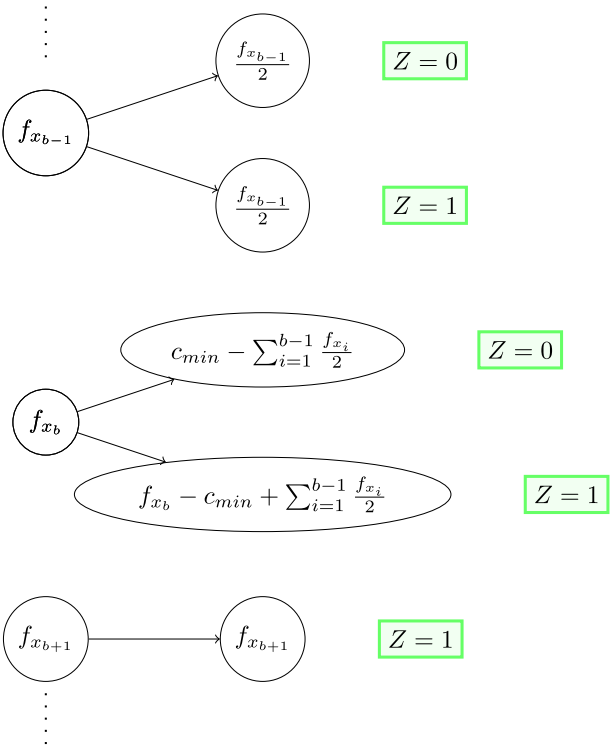


Fig. 26 Arrangement for $Aac(X)_{max}$, $c_{min} > \frac{f_{x_1}}{2}$

criteria, we need to find an a such that $\sum_{i=1}^{a} f_{x_i} > c_{max}$, and the $AV - average - conflict(x_a)$ will be the $Aac(X)_{min}$.

Theorem 3 Finding the arrangement for $Aac(X)_{min} = 0$ using marginal frequency of attribute and target variable is NP-Complete

Proof Given an integer $a \in \{1,2,..s\}$, an instance of f_{x_i} and c_{max}, it can be checked in polynomial time if $\sum_{i=1}^{a} f_{x_i} = c_{max}$. We can reduce the subset sum problem to finding such an a to satisfy $\sum_{i=1}^{a} f_{x_i} = c_{max}$. Subset-sum is NP-Complete (Garey & Johnson, 1979) which completes the proof.

A simple 1/2 -approximation algorithm (Caprara et al., 2000) for Subset-sum problem can be obtained by ordering the inputs in descending order, and then putting the next-largest input into the subset as long as it fits there. Following this approach, we can derive equation (9) for finding $Aac(X,l,l+1,n)_{min}$

$$Aac(X,l,l+1,n)_{min} = \begin{cases} (\sum_{i=1}^{a} f_{x_i} - c_{max}) \times \frac{f_{x_a}}{n-l} & \text{if } \sum_{i=1}^{a} f_{x_i} - c_{max} \leq \frac{f_{x_a}}{2} \\ (c_{max} - \sum_{i=1}^{a-1} f_{x_i}) \times \frac{f_{x_a}}{n-l} & \text{otherwise} \end{cases} \quad (9)$$

where, $f_{x_1} \geq f_{x_2} \geq ... \geq f_{x_a}$, and $\sum_{i=1}^{a} f_{x_i} \geq c_{max}$.

Proof Let, $c_{max} = f_{z_1}$. For $Aac(X,l,l+1,n)_{min} > 0$ to hold, $\sum_{i=1}^{a} f_{x_i} > c_{max}$. In this scenario, $f_{x_1}, f_{x_2}, ..f_{x_{a-1}}$ will have one-to-one mapping with $Z = 1$, hence the $AV - average - conflict$ for each of these f_{x_i} where $i \in \{1,2,..,a-1\}$ will be 0. Only f_{x_a} will have mapping with $Z = 1$ and $Z = 0$ values. All f_{x_i} where $i \in \{a+1,..s\}$ will have one-to-one mapping with $Z = 0$ and hence the $AV - average - conflict$ will be 0 for all such f_{x_i}. Figures 27 and 28 illustrates the scenario for such arrangements when $\sum_{i=1}^{a} f_{x_i} - c_{max} \leq \frac{f_{x_a}}{2}$ and $\sum_{i=1}^{a} f_{x_i} - c_{max} > \frac{f_{x_a}}{2}$ respectively. Hence, $AV - average - conflict(x_a)$ will be equal to $Aac(X,l,l+1,n)$.

Examples 4 and 5 illustrates the two cases of equation (9) for $n = 100$, $Aac(X,0,1,n)_{min} = Aac(X)_{min}$.

Example 4 Let, $S_{F_X} = \{40, 30, 20, 10\}$, $S_{F_Z} = \{85, 15\}$. Here, $c_{max} = 85, f_{x_1} = 40, f_{x_2} = 30, f_{x_3} = 20, f_{x_4} = 10$. $f_{x_1} + f_{x_2} + f_{x_3} = 90 > c_{max}$, so, $f_{x_a} = 20$.Now, $\sum_{i=1}^{a} f_{x_i} - c_{max} = 5 < \frac{f_{x_a}}{2}$. Hence,

$$Aac(X)_{min} = (\sum_{i=1}^{a} f_{x_i} - c_{max}) \times \frac{f_{x_a}}{n} = 5 \times \frac{20}{100} = 1$$

Example 5 Let, $S_{F_X} = \{40, 30, 20, 10\}$, $S_{F_Z} = \{75, 25\}$. Here, $c_{max} = 75, f_{x_1} = 40, f_{x_2} = 30, f_{x_3} = 20, f_{x_4} = 10$. $f_{x_1} + f_{x_2} + f_{x_3} = 90 > c_{max}$, so, $f_{x_a} = 20$.Now, $\sum_{i=1}^{a} f_{x_i} - c_{max} = 15 > \frac{f_{x_a}}{2}$. Hence,

$$Aac(X)_{min} = (c_{max} - \sum_{i=1}^{a-1} f_{x_i}) \times \frac{f_{x_a}}{n} = (75 - 70) \times \frac{20}{100} = 1$$

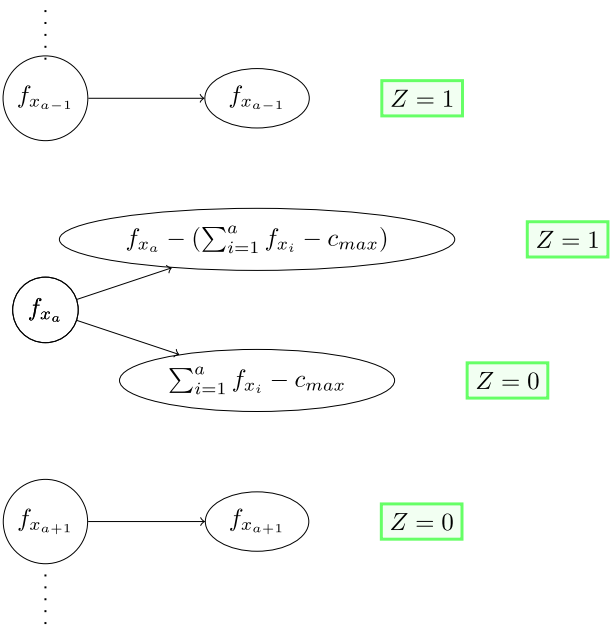


Fig. 27 Arrangement for $Aac(X)_{min} > 0$, $\sum_{i=1}^{a} f_{x_i} - c_{max} \leq \frac{f_{x_a}}{2}$

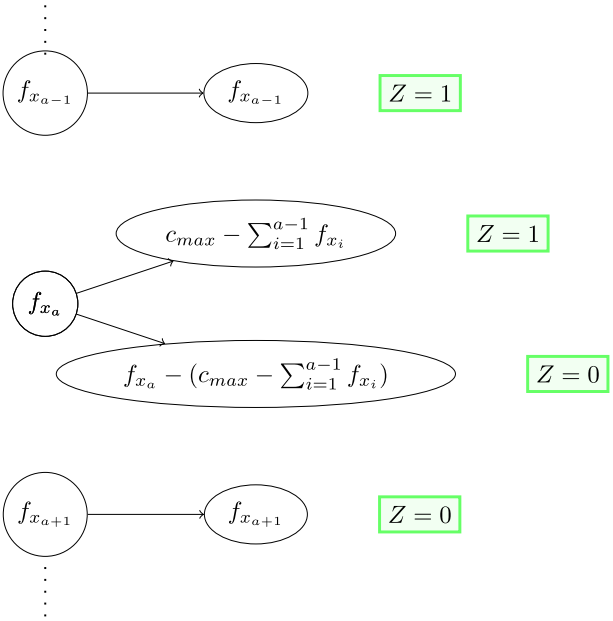


Fig. 28 Arrangement for $Aac(X)_{min} > 0$, $\sum_{i=1}^{a} f_{x_i} - c_{max} > \frac{f_{x_a}}{2}$

Using maximum and minimum values of *Aac* from (8) and (9), we can update upper and lower bounds of *Aac* in (5) and (6) and yield tighter bounds. For two attributes *X* and *Y*, there can be four cases when $LB - Aac(X,l,1,n) > UB - Aac(Y,l,1,n)$.

Case a: $\sum_{i=1}^{a} f_{x_i} - c_{max} \leq \frac{f_{x_a}}{2}, c_{min} \leq \frac{f_{y_1}}{2}$

$$Aac(X,l,1,l) + \left(\sum_{i=1}^{a} f_{x_i} - c_{max}\right) \times \frac{f_{x_a}}{n-l} > Aac(Y,l,1,l) + c_{min} \times \frac{f_{y_1}}{n-l}$$
$$\Rightarrow \frac{f_{y_1} c_{min}}{(n-l)} - \frac{f_{x_a}(\sum_{i=1}^{a} f_{x_i} - c_{max})}{n-l} < Aac(X,l,1,l) - Aac(Y,l,1,l)$$

$$\Rightarrow n - l > \frac{f_{y_1} c_{min} - f_{x_a}(\sum_{i=1}^{a} f_{x_i} - c_{max})}{Aac(X,l,1,l) - Aac(Y,l,1,l)} \tag{10}$$

Case b : $\sum_{i=1}^{a} f_{x_i} - c_{max} \leq \frac{f_{x_a}}{2}, c_{min} > \frac{f_{y_1}}{2}$

$$Aac(X,l,1,l) + \left(\sum_{i=1}^{a} f_{x_i} - c_{max}\right) \times \frac{f_{x_a}}{n-l} > Aac(Y,l,1,l) + \frac{\sum_{j=1}^{e-1} f_{y_j}^2}{2(n-l)} +$$
$$(c_{min} - \frac{\sum_{j=1}^{e-1} f_{y_j}}{2}) \times \frac{f_{y_e}}{n-l}$$
$$\Rightarrow Aac(X,l,1,l) - Aac(Y,l,1,l) > \frac{\sum_{j=1}^{e-1} f_{y_j}^2 + (2c_{min} - \sum_{j=1}^{e-1} f_{y_j}) f_{y_e} - 2f_{x_a}(\sum_{i=1}^{a} f_{x_i} - c_{max})}{2(n-l)}$$

$$\Rightarrow n - l > \frac{\sum_{j=1}^{e-1} f_{y_j}^2 + (2c_{min} - \sum_{j=1}^{e-1} f_{y_j}) f_{y_e} - 2f_{x_a}(\sum_{i=1}^{a} f_{x_i} - c_{max})}{2[Aac(X,l,1,l) - Aac(Y,l,1,l)]} \tag{11}$$

Case c : $\sum_{i=1}^{a} f_{x_i} - c_{max} > \frac{f_{x_a}}{2}, c_{min} \leq \frac{f_{y_1}}{2}$

$$Aac(X,l,1,l) + \frac{(c_{max} - \sum_{i=1}^{a-1} f_{x_i}) f_{x_a}}{n-l} > Aac(Y,l,1,l) + \frac{c_{min} f_{y_1}}{n-l}$$
$$\Rightarrow Aac(X,l,1,l) - Aac(Y,l,1,l) > \frac{c_{min} f_{y_1}}{n-l} - \frac{(c_{max} - \sum_{i=1}^{a-1} f_{x_i}) f_{x_a}}{n-l}$$

$$\Rightarrow n - l > \frac{c_{min} f_{y_1} - (c_{max} - \sum_{i=1}^{a-1} f_{x_i}) f_{x_a}}{Aac(X,l,1,l) - Aac(Y,l,1,l)} \tag{12}$$

Case d : $\sum_{i=1}^{a} f_{x_i} - c_{max} > \frac{f_{x_a}}{2}, c_{min} > \frac{f_{y_1}}{2}$

$$Aac(X,l,1,l) + \left(c_{max} - \sum_{i=1}^{a-1} f_{x_i} \right) \times \frac{f_{x_a}}{n-l} > Aac(Y,l,1,l) + \frac{\sum_{j=1}^{e-1} f_{y_j}^2}{2(n-l)} +$$
$$(c_{min} - \frac{\sum_{j=1}^{e-1} f_{y_j}}{2}) \frac{f_{y_e}}{n-l}$$
$$\Rightarrow Aac(X,l,1,l) - Aac(Y,l,1,l) > \frac{\sum_{j=1}^{e-1} f_{y_j}^2}{2(n-l)} + \frac{(2c_{min} - \sum_{j=1}^{e-1} f_{y_j}) f_{y_e}}{2(n-l)} -$$
$$\frac{(c_{max} - \sum_{i=1}^{a-1} f_{x_i}) f_{x_a}}{n-l}$$

$$\Rightarrow n - l > \frac{\sum_{j=1}^{e-1} f_{y_j}^2 + (2c_{min} - \sum_{j=1}^{e-1} f_{y_j}) f_{y_e} - 2f_{x_a}(c_{max} - \sum_{i=1}^{a-1} f_{x_i})}{2[Aac(X,l,1,l) - Aac(Y,l,1,l)]} \tag{13}$$

Using the upper and lower bounds, top-k attributes of a dataset can be derived by reading $l < n$ records. Algorithm 2 illustrates the steps to find the top-k attributes using the bounds. As we are adopting a greedy approach for finding $Aac(X,l,l+1,n)_{min}$ using equation (9), the returned list of top-k attributes using l records might not always achieve perfect precision compared to that derived using all the n records of database. The algorithm processes l records instead of all the n records to find the top-k attributes, where l can be found using one of the equations in (10)-(13) .

6.2 Experimental evaluation

Algorithm 2 has been implemented in Python 3 and ran on real datasets from Table 6. The result summarized in Table 8 illustrates that although using a greedy approximation approach to find $Aac(X,l,l+1,n)_{min}$, proposed method derives top-k attributes with perfect precision for all the 3 different datasets. Here precision is computed with respect to the top-k attributes found using Algorithm 1. Depending on the data distribution and attribute value domain size, the number of records need to be processed varies. The best result is observed for Census Income dataset with 299,285 records. Our proposed method needs to read only 65% of the total records to derive the top-k attributes for $k = 10$.

inputs: $\mathcal{A}$, n records, k, target variable Z
output: Top-k attributes
$l \leftarrow 1$
$R \leftarrow \{\}$
while $l \leq n$ **do**
  Compute $UB - Aac(X_i, l, 1, n)$ and $LB - Aac(X_i, l, 1, n)$ for $X_i \in \mathcal{A}$
  $kthSmallestUB \leftarrow$ k-th smallest upper bound value of X_i's
  for each $X_i \in \mathcal{A}$ **do**
    if $LB - Aac(X_i, l, 1, n) > kthSmallestUB$ **then** $\mathcal{A} \leftarrow \mathcal{A} \setminus X_i$
    end if
  end for
  if $|\mathcal{A}| = k$ **then**
    break
  end if
  $l \leftarrow l + 1$
end while
$R \leftarrow \mathcal{A}$
Sort R wrt ascending order of $UB - Aac(X_j, l, 1, n) \forall X_j \in R$
Return R

Algorithm 2 Algorithm `top-k-Aac-UB-LB`.

Table 8 Experimental evaluation of Algorithm 2 for $k = 10$

Dataset	n	l	(%) records processed	precision
Census Income	299,285	193,109	64.52	1
Connect-4	67,557	67,396	99.76	1
Kick Car	72,983	72,925	99.92	1

7 Conclusion

The goal of this work is to develop a new measure *Attribute Average Conflict , Aac* to effectively approximate top-k attributes for MI based feature selection, without actually calculating MI. This approximation avoids some of repetitive expensive operations involved in the original MI calculation, such as, logarithms and divisions. Our proposed method is based on using the database concept of *approximate functional dependency* to quantify MI rank of attributes which to our knowledge has not been studied before. We perform extensive experiments using multiple high dimensional synthetic and real datasets with millions of records and implement several baseline algorithms. Our experimental results demonstrate an average of $2x$ speed up compared to the exact method for Mutual Information based feature selection process, while demonstrating highly accurate precision and ndcg. We also demonstrate, on an average, our proposed measure is $4 - 7x$ faster than the state-of-the art method based on adaptive random sampling while exhibiting identical precision and accuracy. We

turn out to be superior to uniform random sampling based baseline in both running time and precision as well. We investigate and establish the upper bound and lower bound of *Aac* to develop an efficient algorithm that finds top-*k* attributes without scanning the full dataset in a single pass. Our experimental evaluation illustrates the applicability of this efficient algorithm.

Author contributions All the authors contributed equally in developing theory and experiments and writing the paper.

Funding The work of Senjuti Basu Roy is supported by NSF grant no. 1942913, 2007935, 1814595, 2118458, and Office of Naval Research grant no. N000141812838, N000142112966. The work of Gautam Das is supported by NSF grant no. 2107296, 2008602, 2037433, 1937143, and USDA grant no. 58-6066-1-034, 58-6066-2-035.

Data availability https://github.com/linsal26/aac-feature-selection

References

Baeza-Yates, R., Ribeiro-Neto, B., & et al. (1999). *Modern information retrieval* Vol. 463. New York: ACM Press.

Caprara, A., Kellerer, H., & Pferschy, U. (2000). The multiple subset sum problem. *SIAM Journal on Optimization*, *11* (2), 308–319. https://doi.org/10.1137/s1052623498348481.

Chandrashekar, G., & Sahin, F. (2014). A survey on feature selection methods. *Computers & Electrical Engineering*, *40*(1), 16–28. https://doi.org/10.1016/j.compeleceng.2013.11.024.

Chen, X., & Wang, S. (2021). Efficient approximate algorithms for empirical entropy and mutual information. In *Proceedings of the 2021 International Conference on Management of Data, pp 274–286*. https://doi.org/10.1145/3448016.3457255.

Cover, T.M., & Thomas, J.A. (1991). *Elements of information theory*. USA: Wiley. https://doi.org/10.1002/0471200611.

Dalkilic, M.M., & Roberston, E.L. (2000). Information dependencies. In *Proceedings of the Nineteenth ACM SIGMOD-SIGACT-SIGART symposium on principles of database systems*. https://doi.org/10.1145/335168.336059 (pp. 245–253).

Dua, D., & Graff, C. (2017). Uci machine learning repository. http://archive.ics.uci.edu/ml.

Elmasri, R., & Navathe, S.B. (2011). *Fundamentals of database systems*. USA: Pearson Education Boston.

Garey, M.R., & Johnson, D.S. (1979). *Computers and intractability a guide to the theory of NP-completeness*. W.H Freeman and Company: New York.

Giannella, C., & Robertson, E. (2004). On approximation measures for functional dependencies. *Information Systems*, *29*(6), 483–507. https://doi.org/10.1016/j.is.2003.10.006.

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, *3*(Mar), 1157–1182.

Han, J., Pei, J., & Kamber, M. (2011). *Data mining: Concepts and techniques*. USA: Elsevier.

Hoque, N., Bhattacharyya, D.K., & Kalita, J.K. (2014). Mifs-nd: a mutual information-based feature selection method. *Expert Systems with Applications*, *41*(14), 6371–6385. https://doi.org/10.1016/j.eswa.2014.04.019.

Huhtala, Y., Kärkkäinen, J., Porkka, P., & et al. (1999). Tane: an efficient algorithm for discovering functional and approximate dependencies. *The Computer Journal*, *42*(2), 100–111. https://doi.org/10.1093/comjnl/42.2.100.

Jović, A, Brkić, K, & Bogunović, N. (2015). A review of feature selection methods with applications. In *2015 38Th international convention on information and communication technology, electronics and microelectronics (MIPRO)*. https://doi.org/10.1109/mipro.2015.7160458 (pp. 1200–1205). IEEE.

Kivinen, J., & Mannila, H. (1995). Approximate inference of functional dependencies from relations. *Theoretical Computer Science*, *149*(1), 129–149. https://doi.org/10.1016/0304-3975(95)00028-u.

Lee, T.T. (1987). An information-theoretic analysis of relational databases– part i: Data dependencies and information metric. IEEE Transactions on Software Engineering (10):1049–1061. https://doi.org/10.1109/tse.1987.232847.

Li, W. (1990). Mutual information functions versus correlation functions. *Journal of Statistical Physics*, *60*(5-6), 823–837. https://doi.org/10.1007/bf01025996.

Liu, J., Li, J., Liu, C., & et al (2010). Discover dependencies from data – a review. *IEEE Transactions on Knowledge and Data Engineering*, *24*(2), 251–264. https://doi.org/10.1109/tkde.2010.197.

Lopes, S., Petit, J.M., & Lakhal, L. (2000). Efficient discovery of functional dependencies and armstrong relations. In *International Conference on Extending Database Technology*. https://doi.org/10.1007/3-540-46439-5_24 (pp. 350–364). Springer.

Mandros, P., Boley, M., & Vreeken, J. (2020). Discovering dependencies with reliable mutual information. *Knowledge and Information Systems*, *62*(11), 4223–4253. https://doi.org/10.1007/s10115-020-01494-9.

Mooney, C.Z. (1997). *Monte carlo simulation 116*. USA: Sage. https://doi.org/10.4135/9781412985116.

Novelli, N., & Cicchetti, R. (2001). Functional and embedded dependency inference: a data mining point of view. *Information Systems*, *26*(7), 477–506. https://doi.org/10.1016/s0306-4379(01)00032-1.

Salam, M.A., Koone, M.E., Thirumuruganathan, S., & et al (2019). A human-in-the-loop attribute design framework for classification. In *The World Wide Web Conference*. https://doi.org/10.1145/3308558.3313547 (pp. 1612–1622).

Vergara, J.R., & Estévez, P.A. (2014). A review of feature selection methods based on mutual information. *Neural Computing and Applications*, *24*(1), 175–186. https://doi.org/10.1007/s00521-013-1368-0.

Wang, C., & Ding, B. (2019). Fast approximation of empirical entropy via subsampling. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. https://doi.org/10.1145/3292500.3330938 (pp. 658–667).