Efficient Algorithms and Human-in-the-loop Approaches for Attribute Design and Selection

by

MD ABDUS SALAM

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2022

To My Parents and Wife

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor Dr. Gautam Das for his continuous guidance, motivation, and support. It has been a great experience working under his supervision. The regular meetings I had with him were very helpful for me in developing the research mindset and investigate problems from various perspectives. Next I would like to thank my collaborator Dr. Senjuti Basu Roy. I learnt a lot from the discussions with her. Her enthusiasm and drive is contagious which inspired me during my research.

I would also like to thank my dissertation committee members: Dr. Ramez Elmasri, Dr. Chengkai Li and Dr. Leonidas Fegaras for their suggestions and comments.

I would like to thank all of my former and current labmates, specially Saravanan, Habibur, Farhadur, Mary, Jees, Sona, Suraj, Shohed for their help in various situations during my journey.

I started my PhD program as a part time student while working full time as a Web Software Developer. As a full time employee it was a challenging task to pursue the journey of PhD. It would not be possible without the amicable atmosphere at work. I would like to thank all my former and current supervisors at work from the Division of Enrollment Management, UT Arlington, for their support to pursue and complete my journey as a part time PhD student.

I am indebted to my father Engr. Mohammad Abdus Sobhan and mother Late Jahanara Begum who instilled the importance of education in me during my childhood. They sacrificed their whole life so that their children could achieve best

education. Their unconditional love and support have provided me the motivation to venture for graduate studies in the USA. I remember my late father-in-law Md. Abdur Razzaque for his inspiration and encouraging words during my PhD journey. Lastly, I thank my loving and caring wife Dr. Shamsun Nahar for always standing beside me and encouraging me to pursue my dreams. It would not be possible to complete this journey without her inspiration and mental support.

March 31, 2022

ABSTRACT

Efficient Algorithms and Human-in-the-loop Approaches for Attribute Design and Selection

Md Abdus Salam, Ph.D.

The University of Texas at Arlington, 2022

Supervising Professor: Gautam Das

Feature engineering and feature selection are two important aspects of data science pipeline. Due to the advancement of data collection techniques in recent years, huge amount of data is becoming available in different industries. Consequently, the importance of data science is increasing for business analytic purpose. Different tools and techniques are being developed to assist data scientists to complete their tasks efficiently. One of the main human involvements in the data science task is for feature engineering and selection. These pre-processing steps will prepare the data in the format desired to be fed into various machine learning algorithms to accomplish predictive tasks. The aim of this dissertation is twofold - first to develop an effective framework to assist data scientist for feature engineering task, and then to develop a new measure to select these features efficiently.

The term *attribute* is used to denote feature in tabular data format. In this dissertation, *a semi-automated, "human-in-the-loop" framework for attribute design* is developed that assists human analysts to transform raw attributes into effective derived attributes for classification problems. The proposed framework is optimization

guided and fully agnostic to the underlying classification model. An algebra with various operators (arithmetic, relational, and logical) to transform raw attributes into derived attributes is presented and two technical problems are solved: (a) the **top-k buckets design** problem aims at presenting human analysts with k buckets, each bucket containing promising choices of raw attributes that she can focus on only without having to look at all raw attributes; and (b) the **top-l snippets generation** problem, which iteratively aids human analysts with top-l derived attributes involving an attribute. For the former problem, an effective exact bottom-up algorithm empowered by pruning capability is presented, as well as random walk based heuristic algorithms that are intuitive and work well in practice. For the latter, a greedy heuristic algorithm is presented that is scalable and effective. Rigorous evaluations are conducted involving 6 different real world datasets to showcase that proposed framework generates effective derived attributes compared to *fully manual or fully automated methods*.

Next, a demonstration of the *semi-automated, "human-in-the-loop" attribute design framework*, namely iFE is proposed. iFE is a desktop application that enables a human analyst to find interpretable derived attributes much quicker than fully manual method. The system first finds k buckets, each containing promising choices of raw attributes that the analyst can focus on only without having to look at all raw attributes. To achieve this, iFE implements a random walk based heuristic algorithm that is intuitive and works well in practice. In the next step, the system iteratively aids the analyst to generate top-l derived attributes within a bucket using arithmetic, relational, and logical operators. The user interface in our system guides the analyst to the final derived attributes in a few number of iterations which saves time and effort as well as boost productivity for the analyst.

Finally a new measure is proposed for efficient approximate mutual information based feature selection. 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, Aac* to effectively approximate top-k attributes, without actually calculating MI. The 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. The effectiveness of the proposed measure is demonstrated with a Monte-Carlo simulation. Extensive experiments are performed using high dimensional synthetic and real datasets with millions of records. Experimental results show that the 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. An analysis is provided for the upper and lower bounds of the proposed new measure and it is shown 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. Experimental evaluation on real datasets is conducted to show the accuracy and effectiveness of this approach.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

Introduction

Human involvement is required in various pre-processing steps of a data science pipeline - from data cleaning to feature engineering and selection. Feature engineering is either fully automated, or fully manual. Domain experts are needed for manual feature engineering tasks. But it can take several passes even for a domain expert to understand important aspects of the data to perform feature engineering task. After this step feature selection is done using some automated tool. Feature selection can be categorized into three main categories - Wrapper, Filtering, and Embedded. Filtering technique uses scoring function based on statistical property of the data and is agnostic of the underlying machine learning (ML) model used in the data science pipeline. Hence, it is one of the popular methods for feature selection. Among different scoring functions used for Filtering technique, Mutual Information (MI) is a popular one for its various favorable statistical properties. Raw dataset is transformed after feature engineering and selection, and this transformed and concise representation of data is fed to the ML algorithm for prediction task.

The term *attribute* is used to denote feature in tabular data. In this dissertation, we develop a semi automated human-in-the-loop attribute design framework that enables the human domain expert to perform attribute design efficiently and effectively. Next, we develop a desktop based application that demonstrates the usability of the framework. Finally, we propose a new measure to effectively approximate Mutual Information (MI) based feature selection. We derive upper and lower bounds on the

proposed new measure and using the bounds, develop efficient algorithm to perform feature selection without scanning the full dataset in a single pass.

1.1 A Human-in-the-loop Attribute Design Framework for Classification

In this work, we investigate a semi-automated "human-in-the-loop" framework that is agnostic to any classification model. Our work is inspired by a handful of recent works [1, 2, 3] which propose a conceptual framework and empirically argue that attribute design by involving human analysts can effectively substitute for either of the two extremes (fully manual or fully automated). However, these do not study how to optimize human involvement in the process or how to guide it.

Our proposed framework is developed around three fundamental aspects. First, We adopt a principled and quantifiable measure of the effectiveness of a derived attribute that is agnostic to the specifics of any underlying classification model. We consider *Mutual Information* (MI in short) [4] to determine the predictive ability of an attribute. The second aspect of our framework is an algebra that dictates how raw attributes are to be combined to produce the derived attributes. We study arithmetic, logical, and relational operators. The third aspect of our framework is the investigation of how to optimize the involvement of humans in the attribute design process and ensure their efforts are successful. This is motivated by a handful of recent works that argue that humans must not be left unguided in the attribute design process [1, 2, 3]. Simply presenting the entire set of raw attributes or hundreds and thousands of raw records to the human analysts might overwhelm them. We propose to present each analyst with small summarized portions of the raw data and raw attributes that is likely to be most helpful for the human to analyze and create useful attributes from. We formulate two technical problems that need to be addressed:

(1) Top-k buckets design: We present each analyst with only k (a small number) *buckets*, each with at most x attributes, that are most predictive. We show that generating these buckets is NP-hard. We provide an exact algorithm, `ExBKT`, and an even more efficient heuristic algorithm called `RandomizedBKT` that performs random walks on the attribute lattice to design the top-k buckets. However, just producing buckets that are highly predictive (high MI) may not be enough. We propose `RandomizedCovBKT` that finds top-k buckets that are not only effective wrt MI, but also they together cover many "good" raw attributes of the dataset.

(2) Top-l snippets generation: Even after the buckets of raw attributes have been generated, the human analyst may have to sift through large volumes of data (i.e., the tuples with those raw attribute values) to design good derived attributes, and this task may still be overwhelming. Thus, we propose an interactive procedure by which an analyst may approach this task.

In summary, this work makes the following contributions:

- Proposed Framework: We initiate the study of a model agnostic semi-automated attribute design framework that judiciously involves human analysts in the loop.
- Technical contributions: We formalize two technical problems around the framework and present several theoretical and algorithmic results.
- Experimental Results: We conduct extensive experiments to demonstrate both effectiveness and scalability of our proposed solutions .

1.2 iFE: Interactive Feature Engineering

In this work, we develop a demonstration of the proposed techniques of semi-automated human-in-the-loop attribute design framework. Our system enables a data scientist to perform feature engineering effectively. User of our system is a domain expert who is either a data scientist or analyst. Given a set of raw attributes, the

task is to predict the target variable. We use the concept of bucket and snippet to perform this task. A bucket is a small subset of raw attributes from which new features can be engineered. It is a way of dealing with high dimensional data by focusing the user's attention on such small but promising subset. Raw attributes in a bucket can be combined using different algebraic operators. Each such combination in a bucket is defined as a snippet. Our framework defines a mechanism to score each bucket and snippet, rank them according to the scoring function, and derive top-k buckets and top-l snippets by applying novel algorithms developed in [5]. Our demonstration shows that our proposed methods help data scientist to craft useful derived attributes much quicker than a fully manual method. Running time of the fully automated techniques vary greatly and they have the limitations of producing large list of derived attributes. Contrarily, the list of derived attributes produced by our framework is customizable though user interaction. Using parallelization feature of the programming language, the system can scale up to multi node clusters as well as scale down to single node multi-core machine. Hence it can be useful for data scientists to determine meaningful important derived attributes from a dataset in a quick and effective way.

1.3 Efficient Approximate Top-k Mutual Information Based Feature Selection

In this work, we propose a new measure to efficiently perform feature selection based on Mutual Information. 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, which leverages scoring functions involving statistical property of the data to select features. *Mutual Information* (MI) is one such popular measure and has been extensively studied in recent work [6, 7, 5], 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

This work makes the following important contributions -

- 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.
- A New measure to approximate MI - We investigate *Approximate Functional Dependency* (AFD) from literature [8, 9, 10, 11, 12, 13] and find deficiency of one popular measure $G3 - error$ [8] to approximate MI based feature ranking. We propose a new measure *Attribute Average Conflict, Aac* that effectively approximates MI, while being much faster computationally. The effectiveness of the proposed measure is demonstrated with a Monte-Carlo simulation
- Experimental Evaluations - We perform extensive evaluations using multiple synthetic and real world datasets (with millions of records and hundreds of attributes) and compare our solutions with multiple methods. 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 for synthetic data, while being $2x$ faster than exact MI calculation. Adaptive uniform sampling[6] 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.

- Efficient algorithm using bounds on proposed measure - We investigate the the upper and lower bounds of the proposed measure *Aac* 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 an attribute. 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 be scanned to compare *Aac* between two attributes 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 *Aac* 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.

CHAPTER 2

A Human-in-the-loop Attribute Design Framework for Classification

In this paper, we present *a semi-automated, "human-in-the-loop" framework for attribute design* that assists human analysts to transform raw attributes into effective derived attributes for classification problems. Our proposed framework is optimization guided and fully agnostic to the underlying classification model. We present an algebra with various operators (arithmetic, relational, and logical) to transform raw attributes into derived attributes and solve two technical problems: (a) the **top-k buckets design** problem aims at presenting human analysts with k buckets, each bucket containing promising choices of raw attributes that she can focus on only without having to look at all raw attributes; and (b) the **top-l snippets generation** problem, which iteratively aids human analysts with top-l derived attributes involving an attribute. For the former problem, we present an effective exact bottom-up algorithm that is empowered by pruning capability, as well as random walk based heuristic algorithms that are intuitive and work well in practice. For the latter, we present a greedy heuristic algorithm that is scalable and effective. Rigorous evaluations are conducted involving 6 different real world datasets to showcase that our framework generates effective derived attributes compared to *fully manual or fully automated methods*.

2.1 Introduction

Attribute design (also known as feature engineering) is one of the most challenging aspects of a data science pipeline, and is considered to be an "arduous process for

data scientists"[1][14, 15], where the raw attributes often need to be transformed into derived attributes that can be more effective for building predictive models such as classifiers. For example, in a healthcare setting involving large, high dimensional and heterogeneous electronic health records (EHR) datasets, an attribute such as the average length of prior hospitalization can be very useful to build effective predictive or classification models on future hospitalization (i.e., readmission) [16]); however, such attributes are not readily available in the raw dataset. For example, Table 2.1 shows that the average length of hospitalization needs to be computed per hospitalization window considering admission and discharge date, and taking the average of these windows.

The state-of-the-art attribute design techniques fall into one of these two extremes: (a) *fully manual*, painstakingly slow and heavily reliant on domain expertise, often requiring data scientists to go through a repetitive trial-and-error exercise until the set of designed attributes are satisfactorily effective or (b) *fully automated* techniques (some notable systems are, Data Science Machine [17][2], ExploreKit [18][3], One Button Machine [19], or Featuretools[4]). Such methods are not model agnostic, require substantial time to identify derived attributes that are opaque to the human analyst.

In this paper, we investigate a semi-automated "human-in-the-loop" framework that is agnostic to any classification model. Our work is inspired by a handful of recent works [1, 2, 3] which propose a conceptual framework and empirically argue that attribute design by involving human analysts can effectively substitute for either

[1]https://www.itproportal.com/features/dont-let-feature-engineering-stagnate-your-ml-projects/

[2]https://people.csail.mit.edu/kalyan/dsm/

[3]https://github.com/giladkatz/ExploreKit

[4]https://docs.featuretools.com/#minute-quick-start

of the two extremes (fully manual or fully automated). However, these do not study how to optimize human involvement in the process or how to guide it.

Proposed Framework: Our proposed framework is developed around three fundamental aspects.

Model agnostic measure: We adopt a principled and quantifiable measure of the effectiveness of a derived attribute that is agnostic to the specifics of any underlying classification model. We consider *Mutual Information* (MI in short) [4] to determine the predictive ability of an attribute. Intuitively, MI is a symmetric measure that captures "correlation" between a pair of attributes and quantifies how much information is contained in one attribute about the other. We are interested in producing derived attributes that have high MI with the target variable (or the class label). MI is known to have several desirable properties: (a) MI is proved to have certain qualitative guarantees when chosen for attribute selection for multiple popular classification models, such as Naive Bayes [4], or linear regression [20]; (b) prior works have also shown that MI optimizes important properties in attribute selection, such as; relevance, complementarity, and redundancy [4] and (c) finally, as we shall show later in the paper, MI satisfies upward closure [21] which is useful for designing effective algorithms.

Attributes algebra: The second aspect of our framework is an algebra that dictates how raw attributes (numerical or categorical) are to be combined to produce the derived attributes. We study arithmetic, logical, and relational operators. Our initial example of average length of prior hospitalization is an example of arithmetic operator, whereas an example of the logical operation is the following Boolean attribute: *elderly* AND *diabetic* OR *covered under medicaid*. A patient is more vulnerable to future hospitalization if she gets an "yes" on this attribute. A logical operator on the other hand is *obesity* that is set to True, when $BMI > 30$.

Guiding human analysts: The third aspect of our framework is the investigation of how to optimize the involvement of humans in the attribute design process and ensure their efforts are successful. This is motivated by a handful of recent works that argue that humans must not be left unguided in the attribute design process [1, 2, 3], Simply presenting the entire set of raw attributes or hundreds and thousands of raw records to the human analysts might overwhelm them. We propose to present each analyst with small summarized portions of the raw data and raw attributes that is likely to be most helpful for the human to analyze and create useful attributes from. We formulate two technical problems that need to be addressed:

(1) Top-k buckets design: We present each analyst with only k (a small number) *buckets*, each with at most x attributes, that are most predictive. For example, when $x = 3$, a useful bucket that is likely to have predictive qualities may contain {elderly, diabetic, covered-by-medicaid}. We show that generating these buckets is NP-hard. We provide an exact algorithm, `ExBKT`, and an even more efficient heuristic algorithm called `RandomizedBKT` that performs random walks on the attribute lattice to design the top-k buckets. However, just producing buckets that are highly predictive (high MI) may not be enough. We propose `RandomizedCovBKT` that finds top-k buckets that are not only effective wrt MI, but also they together cover many "good" raw attributes of the dataset.

(2) Top-l snippets generation: Even after the buckets of raw attributes have been generated, the human analyst may have to sift through large volumes of data (i.e., the tuples with those raw attribute values) to design good derived attributes, and this task may still be overwhelming. Thus, we propose an interactive procedure by which an analyst may approach this task. Given a bucket, the analyst starts designing a derived attribute (i.e., an algebraic expression involving the raw bucket attributes) interactively (i.e, term by term). At each iteration, our framework recommends l

snippets, i.e., it suggests the l best ways to extend the partially created derived attributes using the algebra. More formally, at a given iteration the snippet generation process suggests how to augment the partially composed derived attribute by length j, i.e., combining j new raw attributes with the derived attribute developed thus far. For instance, if *elderly* has already been selected by the analyst, and if $j = 1$, an example snippet will suggest a visual distribution that shows how *elderly* AND *diabetic* correlates with the prediction target variable *hospital readmission*.

Evaluations: Rigorous evaluations are conducted considering 6 real world datasets by comparing our solutions with two fully automated methods (ExploreKit and Featuretools), and a fully manual domain expert guided attribute design process, as well as intuitive baselines. Our experimental results demonstrate that we scale up to $7x - 20x$ faster compared to fully automated (ExploreKit) and the fully manual process, while ensuring similar quality (average improvement 14%). By leveraging the domain expert, our framework avoids falling into the pitfall where derived features could actually be detrimental to the performance (unlike FeatureTools). It also has other appealing properties, such as being easily parallelizable, and exhibiting "anytime behavior" whereby it gives meaningful results at any point of execution. Our scalability results demonstrate that both bucket design and snippet generation procedures are efficient and can work interactively with the human in the loop.

In summary, the paper makes the following contributions:

- Proposed Framework: We initiate the study of a model agnostic semi-automated attribute design framework (Section 2.2) that judiciously involves human analysts in the loop.
- Technical contributions: We formalize two technical problems around the framework and present several theoretical and algorithmic results (Sections 2.3 and 2.4).

- Experimental Results: We conduct extensive experiments to demonstrate both effectiveness and scalability of our proposed solutions (Section 2.5).

2.2 Preliminaries and Formalism

In this section, we describe our data model, present our framework, and formalize the technical problems.

Example 1. *We present a toy running example in Table 2.1 that provides longitudinal data of a heart failure datamart in a hospital. The objective is to build a classifier that predicts whether a patient getting discharged from the hospital will be readmitted within* 6 *months of discharge, considering predictors from base attributes (first* 10 *columns) and derived attributes (designed using the base attributes). The data is augmented by adding the last column, representing the class label per patient per admission instance.*

patient −id	*admission date*	*dis. date*	*gen− der*	*BMI*	*income*	*medi− caid*	senior	*dia− betic*	*primary− diag− nosis*	*read− mission*
p1	10-1-2013	10-23-2013	M	32.5	low	Y	Y	Y	*Congestive heart failure*	N
p1	5-2-2014	5-10-2014	M	32.2	low	Y	Y	Y	*Heart Attack*	Y
p2	5-12-2014	5-14-2014	F	26.7	medium	N	N	N	Arrhythmia	N
p3	10-1-2015	11-1-2015	M	29.9	low	N	N	Y	*Cardio− myopathy*	N
p3	6-10-2016	6-21-2016	M	29.8	low	N	N	Y	*Cardio− myopathy*	N
p4	12-12-2017	12-14-2017	F	23.4	medium	N	N	N	Arrhythmia	N
p4	9-2-2018	9-29-2018	F	27.8	medium	N	N	N	*Heart Attack*	Y

Table 2.1: Example 1 toy heart failure datamart

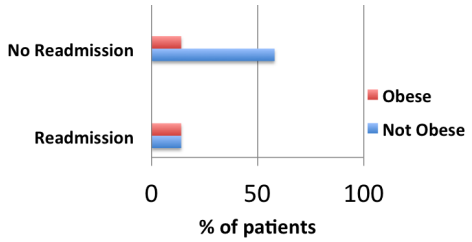


Figure 2.1: A snippet on Obesity

2.2.1 Data Model

Base attributes, records, and target variable: A given dataset is comprised of a set $\mathcal{A}$ of n attributes and m records, as well as an additional target attribute (column) Z. These attributes are referred to as base or raw attributes. In this work, we consider all major types of attributes including numeric, Boolean, and categorical. Our proposed approach can work for a wide variety of predictive modeling tasks including classification and regression (we refer to them classification in general). Depending on the nature of the problem, Z is a continuous variable (regression problem), or has discrete values (classification problem). Using Example 1, $n = 10$ and $m = 7$, Z corresponds to readmission within 6 months of discharge (discrete).

Algebra, $\mathcal{L}$: A set of operators (arithmetic, relational and logical) which are applied to one or more base attributes to combine them. When the base attributes are numeric, we consider arithmetic operators to combine them: addition ($+$), subtraction ($-$), multiplication (x), division ($/$). For Boolean attributes, we consider logical operators, AND, OR. We also support all relational operators that compare two base attributes or a base attribute with a constant (e.g. $A_i > A_j$ or $A > 100$). Our framework is flexible enough to support arbitrary operators. We also support aggregate operators over a set of tuples.

Derived Attribute, d: An attribute that combines two or more base attributes using the algebra $\mathcal{L}$. Using Example 1, we can create a derived attribute *length*

of stay by subtracting discharge date from the admission date. Another derived attribute *obesity* can be obtained through the relational operator $>$ as $BMI > 30$.

Independent variable, dependent variable: A base attribute or a derived attribute is an independent variable (V) in our problem as that is used to predict the target variable Z, which is the dependent variable. Our overall intention is to craft a set of derived attributes as independent variables that are highly "predictive" to the target (dependent) variable.

Mutual Information (MI): We are interested in calculating "predictiveness" of an independent variable V to the target variable Z. For that, we use Mutual Information (MI) that captures information theoretic "correlation" (indeed there exists a relationship between MI and correlation [22] between two random variables that quantifies the amount of information obtained about one through the other). When V and Z are discrete [5], $MI(Z, V)$ is defined as follows:

$$MI(Z, V) = \sum_{z \in Z} \sum_{v \in V} p(z, v) \log \frac{p(z, v)}{p(z)p(v)} \tag{2.1}$$

where $p(z, v)$ is the joint probability function of Z and V, and $p(z)$ and $p(v)$ are the marginal probability distribution functions of Z and V respectively. Of course, V could be a single base attribute, a small set of base attributes, or a derived attribute.

2.2.2 Proposed Framework

Our proposed framework consists of two technical steps. We first design a set of k buckets, each with at most x base attributes. Given a bucket and the algebra $\mathcal{L}$, the analyst then starts composing a small number of derived attributes from the bucket. This next step is referred to as *snippet generation*, and works iteratively with the analyst until she decides to stop. In each step in snippet generation, the

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

analyst extends the currently composed derived attributes by a small amount. Our algorithmic contributions are focused on these two steps. In Section 2.3, we analyze the Top-k buckets design problem and describe our solutions. In Section 2.4, we present our solutions for the Top-l snippets generation problem.

Definition 1. Score of a bucket, $sc(b)$ **:** *For a bucket b with $A_1, A_2, \ldots A_x$ base attributes, the score of b, i.e., $sc(b)$ is the MI between Z and the Cartesian product of the base attributes that are part of b.*

$$sc(b) = MI(Z, [A_1 \times A_2 \times A_3.... \times A_x]) \tag{2.2}$$

Using Example 1, $sc(gender, senior) = MI(readmission, [gender \times senior])$

Definition 2. Coverage of a set of buckets: *coverage of a set of buckets $Cov(b_1, b_2, ..b_k)$ is the size of the union of the base attributes that are present in these buckets.*

$$Cov(b_1, b_2, ..b_k) = |b_1 \cup b_2 \cup ....b_k| \tag{2.3}$$

Using Example 1, if $k = 2, x = 2$, $b_1 = [gender, senior]$, $b_2 = [medicaid, gender]$, then $Cov(b_1, b_2) = 3$.

Definition 3. Snippet, s**:** *A snippet s is a visual representation of a joint distribution between Z and a derived attribute d^s in the snippet.*

Figure 2.1 shows one such snippet between obesity and hospital readmission.

Definition 4. Score of Snippet, $Sc(s)$**:** *Score of a snippet is the MI between Z and derived attribute d^s in the snippet s, i.e., $MI(Z, d^s)$.*

Using Figure 2.1, $Sc(obesity) = MI(readmission, obesity)$.

2.2.3 Problem Definitions

Problem 1: Top-k-Buckets Design: Given a set of attributes A, number of required buckets k and maximum number of attributes in each bucket x, our objective here is to design Top-k buckets based on MI, i.e., finding the k-buckets (each with at most x attributes) with the highest MI and present those buckets to the the human analyst to investigate further for creating derived attributes.

We also investigate the top-k coverage aware bucket generation problem, where the objective is to create "high quality" buckets *that together cover* all base attributes that have high MI with the target variable.

Problem 2: Coverage Aware Top-k buckets: The objective is to create Top-k buckets such that the score of each bucket, $sc(b)$ in Top-k is above a certain threshold δ and $Cov(b_1, b_2, ..b_k)$ is maximized.

Problem 3: Top-l Interactive Snippets Generation: Each step of the interactive snippet generation takes as inputs a bucket b, an integer j, the currently composed set of derived attributes $\mathcal{D}'$, the algebra $\mathcal{L}$; and produces l snippets with highest scores, where each derived attribute d_i'' in snippet s_i is created by extending d_i', that is, by adding j additional attributes that are part of b, involving $\mathcal{L}$.

2.3 Top-k Buckets Design

Top-k Buckets Design takes x (the maximum size of each bucket), k (the number of buckets), the dataset (base attributes A, target variable Z, and the records), and produces Top-k buckets, each of size at most x that are highest in MI with Z.

Theorem 1. *The Top-k Buckets Design Problem is NP-hard, for an arbitrary x and k.*

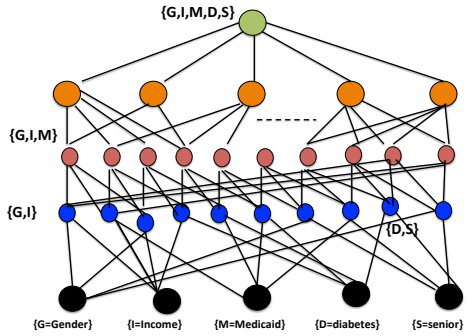


Figure 2.2: Attribute Lattice

Proof. (Sketch): As shown in [23], for an arbitrary maximum bucket size x and number of buckets k, the problem of identifying the number of distinct buckets of maximum size x (that have maximal MI in our case) is #P-Hard. Therefore, the enumeration problem of finding top-k buckets of maximum size x is NP-hard. □

Intuitively, the bucket design problem bears similarity with the itemset mining problems based on association rules (support) or other correlation measures [24], such as Chi-Square [21]. WRT our problem, a base attribute could be considered an item, and therefore mining a bucket with at most x-base attributes is akin to mining an itemset with at most x items that satisfy a certain property. Thereby, the Top-k buckets generation problem seemingly appears similar to Top-k itemset mining problems.

Most popular and effective algorithms in this problem space make use of the *upward or downward closure property of the itemsets* based on the underlying measures (for example, support is downward closed, while chi-square is upward closed [21]). These properties enable efficient algorithm design for the itemset mining problems.

Popular algorithms such as Apriori [24] make use of this property extensively in mining frequent itemsets. They execute in a bottom-up manner by discarding

any itemset from consideration whose subsets are not frequent based on the support threshold [24].

However, designing an Apriori [24] type of algorithm is impractical for our problem for several reasons: firstly, because *MI is not downward closed;* secondly, because our problem does not have a *support threshold* like Apriori, and finally because our problem has *more constraints* (x and k) as inputs.

Lemma 1. *Mutual Information is upward closed.*

Proof. Without loss of generality, let us assume Z is the target variable and A_1, A_2, A_3 are three base features. We have to prove $MI(Z, [A_1, A_2]) \leq MI(Z, [A_1, A_2, A_3])$.

Based on MI definition [25]:

$$MI(Z, [A_1, A_2]) = H(Z) - H(Z|[A_1, A_2]) \tag{2.4}$$

where $H(Z|[A_1, A_2])$ is the conditional entropy of Z given A_1, A_2.

Using Equation 2.4, we therefore prove $H(Z|[A_1, A_2]) \geq H(Z|[A_1, A_2, A_3])$.

$H(Z|[A_1, A_2, A_3])$

$$\begin{aligned}
&= \Sigma_{a_1,a_2,a_3} p(a_1, a_2, a_3) \times \\
&\Sigma_z p(z|a_1, a_2, a_3) \log p(z|a_1, a_2, a_3) \\
&\leq \Sigma_{a_1,a_2,a_3} p(a_1, a_2) \times p(a_3) \Sigma_z p(z|a_1, a_2) \times \\
&p(z|a_3) \log p(z|a_1, a_2) \log p(z|a_3) \\
&\leq H(Z|[A_1, A_2]) \times \Sigma_{a_3} p(a_3) \Sigma_z p(z|a_3) log p(z|a_3) \\
&\leq H(Z|[A_1, A_2])
\end{aligned} \tag{2.5}$$

Therefore, $H(Z|[A_1, A_2]) \geq H(Z|[A_1, A_2, A_3])$, proving $MI(Z, [A_1, A_2]) \leq MI(Z, [A_1, A_2, A_3])$. □

An Apriori-Like Algorithm is Impractical. Since MI is upward closed, a level-by-level algorithm such as Apriori must be done in a top-down fashion for our problem, as opposed to the typical bottom-up fashion. Our problem has an additional constraint on the maximum bucket size that requires the algorithm to climb to level x first before it can start generating possible buckets. Moreover, our bucket generation problem does not come with any provided threshold of MI. Instead, it has the number of buckets constraint k. Therefore, it has to start at level x (which will have $\binom{n}{x}$ number of buckets of size x) with the highest MI value possible as the threshold and determine all the size x buckets that satisfy the threshold. If the number of generated buckets is less than k, it then has to reduce the MI threshold value systematically until a total of exactly k buckets have been generated. Naturally, to faithfully reproduce Apriori for our problem, one has to make several runs of the algorithm, until exactly k buckets have been produced. Clearly, such a process is computationally impractical for large n, x, and k.

Next, we describe our solutions for this problem - first an exact algorithm (`ExBKT`) that produces the top-k buckets of at most size x with the highest MI, and then random walk based algorithms that are highly efficient and work well in practice.

2.3.1 Exact Algorithm

Algorithm `ExBKT` extensively exploits a few observations that we make about MI. In particular, while computing the MI of a set of attributes with the target variable Z, we observe that one can produce effective *lower and upper bounds* on MI. These bounds shall allow us to design a bottom up algorithm that goes level by level and effectively prunes a set of candidate attributes (buckets) by using upper bounds. The observation is rather simple - between two subsets of candidate buckets, if one has higher lower bound of MI than the other bucket's upper bound of MI, then

the former bucket should get promoted as the winner between these two buckets. Algorithm `ExBKT` makes use of this observation to prune buckets that are never going to be part of the top-k results. Before we describe this algorithm in detail, we describe how to effectively compute lower and upper bound (LB and UB respectively) of MI of a set of attributes (candidate buckets).

Theorem 2. *Upper Bound: Given a set of t base attributes* $A_1, A_2,$
$\ldots A_t$ *and the target variable Z,* $MI(Z, [A_1, A_2, \ldots A_t]) \leq H(Z) + H(A_1, A_2, \ldots A_t)$

Proof.

$$\begin{aligned} MI(Z, [A_1, A_2, \ldots A_t]) &= H(Z) - H(Z|[A_1, A_2, \ldots A_t]) \\ &= H(Z) - H(Z, [A_1, A_2, \ldots A_t]) + \\ H(A_1, A_2, \ldots A_t) \text{ since } &[H(Y|X) = H(X,Y) - H(X)] \end{aligned} \tag{2.6}$$

Therefore,

$$MI(Z, [A_1, A_2, \ldots A_t]) \leq H(Z) + H(A_1, A_2, \ldots A_t) \tag{2.7}$$

Applying the chain rule of entropy, the right hand side of the inequality could be expressed further as $H(Z) + H(A_1) + H(A_2|A_1)$
$+H(A3|A_1A_2) + \ldots H(A_t|A_1, A_2, \ldots A_{t-1})$ □

Theorem 3. *Lower Bound: Given a set of t base attributes* $A_1, A_2,$
$\ldots A_t$ *and the target variable Z,* $MI(Z, [A_1, A_2, \ldots A_t]) \geq$
$MI(Z, [A_1, A_2, \ldots A_{t-1}])$

Proof. This comes directly from lemma 1. □

Algorithm Description: Algorithm `ExBKT` runs in a bottom-up fashion. It starts at the bottom of attribute lattice with singleton base attributes as buckets, and gradually walks up the lattice, level by level. For illustration purposes, consider only 5 attributes from Example 1, abbreviated $\{G, I, M, D, S\}$. The bottom layer refers

to the 5 singleton base attributes and their computed MI wrt the target variable Z. Once the score of each of size 1 bucket is computed, then in the next level, the algorithm combines two base attributes and computes 10 buckets of size 2. For each bucket b_i, it maintains two scores: the lower bound score of b_i, $sc^{lb}(b_i)$ derived from Theorem 3 and the upper bound score of b_i, $sc^{ub}(b_i)$ derived from Theorem 2. Between two buckets b_i and b_j, if the upper bound of b_i (i.e., $sc^{ub}(b_i)$) is smaller than the lower bound of score of b_j (i.e., $sc^{lb}(b_j)$), then b_i and all of its supersets get dropped from further consideration. Using Figure 2.2, if $sc^{lb}\{G, I\}$ is larger than the upper bound score $sc^{ub}\{M, D\}$, then the latter bucket gets dropped from further consideration. Additionally, all buckets that contain $\{M, D\}$ as some base attributes do not need to be considered. If $x = 3$, then the algorithm continues to climb up the attribute lattice to the next level, finding buckets with at most size 3 base attributes, but applies pruning between the buckets using the lower and upper bound as before. Once it finishes the traversal, it produces the top-k buckets with the k-highest MI with the target variable. The pseudo-code is described in Algorithm 1.

Lemma 2. *Algorithm* `ExBKT` *produces the exact top-k buckets.*

Proof. (sketch:) The intuitive argument is that `ExBKT` only drops a bucket and its super-set if its upper bound score is not larger than the lower bound score of other buckets, which will produce the exact solution, because of Lemma 1. □

Running Time: In the worst case `ExBKT` may take $O(n^x)$, hence it is exponential. However, in practice, the pruning can be very effective and the algorithm converges much sooner.

2.3.2 Random Walk Based Algorithms

Algorithm 1 Algorithm `ExBKT`

inputs: $\mathcal{A}$, m records, x, k, target variable Z.

output: Top-k buckets, each of size at most x.

compute all buckets of size 1 and their score.

$i = 2$

while $i \leq x$ **do**

Compute $sc^{ub}(b)$ and $sc^{lb}(b)$ of each bucket b;

IF $sc^{ub}(b_w) \leq sc^{lb}(b_y)$

Drop b_w and any superset of b_w

$i = i + 1$

end while

While `ExBKT` works well in practice, it has an exponential running time in the worst case. Furthermore, `ExBKT` is not easily extensible when we have to optimize any other criteria in addition to MI. Specifically, for our proposed problem *Coverage Aware Top-k buckets*, `ExBKT` can not be adapted to generate the best k-buckets with the highest coverage. Therefore, we propose faster heuristic alternatives that provide good solutions most of the time.

Our proposed Algorithm `RandomizedBKT` is motivated by random walk on the attribute lattice [21] and has a unified solution for both *Top-k buckets Design* and *Coverage Aware Top-k buckets Design* problems. The core idea is to compose a bucket of size x, by performing a random walk on the attribute lattice (refer to Figure 2.2). An attribute is added to a bucket by using weighted sampling without replacement. The weight of an attribute is based on the optimization criteria. A bucket is formed by performing a random walk on the attribute lattice, until its maximum size x has been reached. This random walk is repeated, until k-unique buckets have been derived and

any additional criteria (e.g., threshold δ for the latter problem) has been satisfied. With this high level description above, now we describe how to compute the weight of an attribute in different scenarios. For the *Top-k buckets Design* problem, the weight of an item is directly proportional to its MI, i.e., $weight(A) = MI(Z, A)$.

The *Coverage Aware Top-k buckets Design* problem requires that in addition to MI threshold δ, the coverage of the Top-k buckets also must be maximized. The weight of an attribute is a ratio; it is proportional to its MI, but inversely proportional to the number of times it is present in other buckets that have been computed thus far. Indeed, this latter criteria is designed to ensure high coverage.This condition can be extended to include feature cost as well, so that costly features are assigned lower preference. As before, the algorithm terminates when we obtain top-k unique buckets. Using Figure 2.2, if $k = 5$ and $x = 2$, when the random walk produces the 5th bucket, if M has appeared in all other 4 buckets, the weight of M is dampened by a factor of 4 even if it is high in MI with readmission. The pseudo-code is described in Algorithm 2.

Running Time: Each run of `RandomizedBKT` takes at most $O(n)$ time. The total running time of the algorithm is dominated by the number of different random walks that needs to be performed before it returns k buckets.

2.4 Top-l Interactive Snippets Generation

Recall that we propose snippet generation as an interactive process that continues until the human is done crafting the attributes. Each step of this process takes as inputs a bucket b, an integer j, the currently composed set of derived attributes $\mathcal{D}'$, and the algebra $\mathcal{L}$ to produce l snippets with highest scores (MI with Z), where each derived attribute d''_i in snippet s_i is created by extending d'_i by adding j additional

Algorithm 2 Algorithm `RandomizedBKT`

inputs: set of base attributes $\mathcal{A}$, m records, target variable Z, x, k, δ (for the coverage aware problem)

output: Top-k buckets of size at most x

$\mathcal{B} = \{\}$

while $|\mathcal{B}| < k$ **do**

  while $size(b_i) < x$ **do**

    $weight(A) = MI(Z, A)$ ▷ for the top-k variant

    $weight(A) \propto \frac{MI(Z,A)}{\#A \text{ has been used before}}$ ▷ for the Coverage aware variant

    Sample A based on $weight(A)$ and add it to b_i

  end while

  IF $sc(b) < \delta$ ▷ only for the Coverage aware variant

    Drop b

  $\mathcal{B} = \mathcal{B} \cup b$

end while

attributes that are part of b, involving $\mathcal{L}$. Each of these l snippets are recommended to the human as visual distributions to aid her design derived attributes.

Therefore, our computational challenge is to produce l snippets effectively in each step involving each $d'_i \in \mathcal{D}'$. A natural choice is to investigate a greedy algorithm that builds the final derived attribute bottom-up by selecting the best base attributes from the bucket and combining them with the best operators to recommend a snippet. In order for this greedy algorithm to provide any provable guarantee, this greedy selection process needs to satisfy certain properties. For example, given four attributes A_i, A_j, A_r, A_s, if $MI(Z, A_i) > MI(Z, A_r)$, and $MI(Z, A_j) > MI(Z, A_s)$, then it seems intuitive that if we combine the two attributes with higher MI (A_i, A_j)

A_i	A_j	A_r	A_s	Z
1	-1	0	0	0
2	-2	1	0	1
3	-3	0	1	0
4	-4	1	1	1

Table 2.2: Example where $MI(Z, A_i + A_j) < MI(Z, A_r + A_s)$

with an operator, the resulting derived attribute should have higher MI than the combination of the other two attributes (A_r, A_s) using the same operator. Unfortunately, as we prove below with counter examples, such a property fails to hold *even for very simple operators such as arithmetic addition.* This makes the snippet generation more challenging than the previous step of bucket generation.

Lemma 3. *Given four attributes A_i, A_j, A_r, A_s, if $MI(Z, A_i) > MI(Z, A_r)$, and $MI(Z, A_j) > MI(Z, A_s)$, $MI(Z, A_i+A_j)$ may or may not be greater than $MI(Z, A_r+A_s)$*

Proof. (Sketch): We prove this by counter examples. We present two examples - one shows $MI(Z, A_i+A_j) < MI(Z, A_r+A_s)$ and the other shows the inequality other way. Table 2.2 shows the first scenario. To create the second scenario, if we replace A_j with $5, 6, 7, 8$ in the 4 different rows respectively, we have $MI(Z, A_i+A_j) > MI(Z, A_r+A_s)$.

□

Proposed Algorithm: Our proposed algorithm `GSnippet` is a greedy algorithm - it still generates a snippet bottom up in each step. Given a bucket b, it first sorts the base attributes in the bucket in descending MI. Then given j (the number of additional attributes to extend a partially created derived attribute d'_i), it finds the top-j attributes in b that are not in d'_i. These are the candidate set of attributes for the snippets that involve d'_i. After that, it attempts to combine these j attributes

considering all the operators in $\mathcal{L}$ with d'_i. It ranks each of these created combinations and produces the l-combinations as snippets that have the top-l MI score with Z. Algorithm 3 presents the pseudo-code.

The reason that `GSnippet` exhaustively considers all the operators in $\mathcal{L}$ (and cannot make any greedy or more efficient look-ups) is because of Lemma 3. In order to avoid exhaustive search, other alternative metrics to MI need to be explored that can guarantee upper/lower bound for features combined under algebraic operators. Nevertheless, as we shall show in our experiment, `GSnippet` runs at interactive speeds and produces effective recommendations for the human analyst.

Using Example 1, if $b = \{admission.date, dis.date\}$, $l = 1$, $j = 2$, $\mathcal{L} = \{+, -, \times, /\}$, and current $d = \{\}$ (i.e., empty), then `GSnippet` will rank $admission.date + dis.date$, $admission.date - dis.date$, $admission.date \times dis.date$, $admission.date/dis.date$, and take the one which has the highest MI with readmission.

Algorithm 3 Algorithm `GSnippet`

inputs: bucket b, a derived attribute d'_i, target variable Z, j, l, $\mathcal{L}$

output: Top-l snippets involving d'_i

Sort attributes in b in descending order of MI

Select set S with top-j attributes from b that are not in d'_i

$\mathcal{C} = S \bigcup d'_i$

Combine $c_i \oplus c_j \oplus .... \oplus d'_i$, $\oplus \in \mathcal{L}$, $c_i \in \mathcal{C}$

Rank each combination wrt MI

Return top-l

Running Time: Each run takes $O(|\mathcal{L}|^j logl)$ time, the majority of which is spent on brute-forcing on the operator set $\mathcal{L}$.

2.5 Experimental Evaluation

We conducted comprehensive experimental analysis to compare our proposed approach with fully automated methods, as well as fully manual (domain expert guided) solution. We investigated both quality and running time in this process.

2.5.1 Experimental Setup

Hardware and Platform. All our experiments were conducted on a quad-core 2.2 GHz machine with 16 GB of RAM and 1 TB of hard disk. We used Python to implement our algorithms and used scikit-learn for building the classification models.

Datasets. We evaluated our algorithms against a wide variety of datasets that are considered to be popular choice for attribute design problems. Due to lack of space, we report our results on 5 datasets from UCI repository and one from Kaggle. They cover a diverse array of domains (agriculture, medicine, and e-commerce) and contain attributes that are amenable to constructing derived features. Table 2.3 has more details.

DataSet	**# Records**	**# raw attributes**
pollen	3848	5
delta_elevators	9517	6
mammography	11183	6
space	3107	6
diabetes	768	8
home	506	14

Table 2.3: Reported datasets characteristics

Compared Methods. *(1) Fully Automated Methods.* We implemented two fully automated state-of-the-art approaches for comparison - Featuretools[6] and ExploreKit[7]. They both have open source repositories that allow us to evaluate them fairly.

(2) Our proposed algorithms. These are the solutions that are presented in Sections 2.3 and 2.4.

(3) Buckets Design Baseline algorithms. We compared our proposed buckets design algorithm against an intuitive baseline algorithm (referred to as `Greedy`) that groups attributes on mutual information. It started with an empty bucket and x attributes were added through importance sampling greedily, where the importance is proportional to the marginal increase in MI. The process was repeated k times.

(4) Fully manual method. In this scenario, a domain expert was involved in crafting the derived attributes. We present a case study towards that in Section 2.5.5.

Evaluation of our proposed framework. Our proposed framework has two steps: the first one, top-k buckets design, is fully automated and does not require any human involvement. We have described three algorithms for that. The second step, top-l snippets generation, proposes the list of top l snippets for each of the derived attribute chosen by the human analyst. The human analyst can either accept the choice, select another snippet or even construct a new one. This process is repeated interactively. The framework continues to retain the top-l choices, if no human guidance is given and is fast enough to be done in near real-time. Finally, the designed derived and the base attributes are passed through existing popular classification models with an average of 10 runs.

Parameter Settings.

x, k, j, l, random walk: We varied the size of a bucket x between 2 to the maximum

[6]https://docs.featuretools.com/#minute-quick-start

[7]https://github.com/giladkatz/ExploreKit

number of base attribute with the default value being 5. The snippet extension parameter j is set to 1. Finally, both k and l were set to 5 by default. We ran the random walk for 1000 iterations and picked top-k from it.

Classification models. While our process is classifier agnostic, for the purpose of comparison, we considered two popular classifiers, Support Vector Machines (SVM) and Random Forests (RF). For the latter, we use 100 trees using a depth of 2. Training and testing are performed with a $70\% - 30\%$ split of the data, akin to ExploreKit.

Algebra. By default, we used only arithmetic operators. We then varied the grammar to include logical, relational and other aggregate operators. Overall, our approach can support all the operators described in both ExploreKit and Featuretools.

Performance Measures. *Qualitative measures.* For measuring the classifier performance, we reported the Area Under the Curve (AUC). This has been used in prior work such as ExploreKit as it provides a holistic view of the classifier and attribute design. Higher the AUC, better the classifier performance. We reported the percentage improvement in AUC with base(raw) attributes and base+derived attributes. For example, if the classifier had an AUC of 0.7 with base attributes but 0.8 with base+derived attributes, the improvement is $\frac{0.8-0.7}{0.7} = 14\%$.

Scalability measure. We used time in seconds to evaluate the efficiency of our algorithms.

Data	**EK-SVM**	**Time (s)**	**FT-SVM**	**Time (s)**	**MI-SVM**	**Time (s)**
pollen	1.93%	1063	2.07%	93	2.10%	42
delta_elevators	0.23%	3980	-3.20%	465	0.20%	180
mammography	31.61%	2413	41.93%	204	38.00%	191
space	3.83%	986	1.77%	135	3.20%	186
diabetes	4.63%	343	-13.64%	468	3.60%	390

Table 2.4: AUC improvement and runtime comparison with fully automated methods using SVM

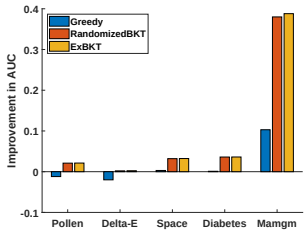


Figure 2.3: Comparing Baselines : AUC

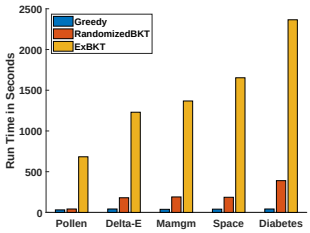


Figure 2.4: Comparing Baselines : Running Time

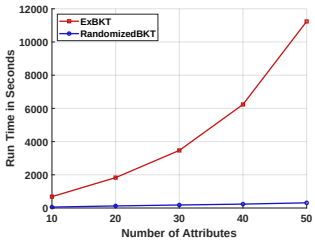


Figure 2.5: Varying Number of Attributes

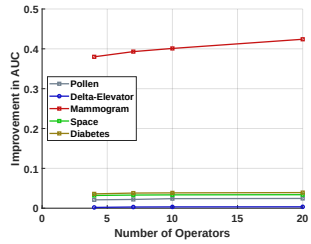


Figure 2.6: Varying Algebra

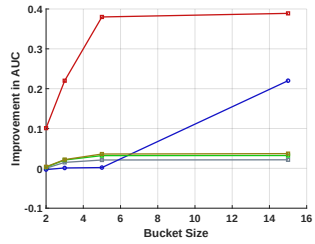


Figure 2.7: Varying Bucket Size

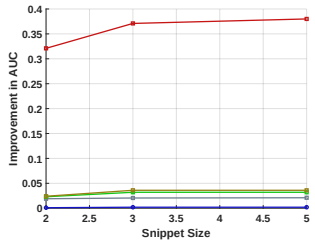


Figure 2.8: Varying Snippet Size

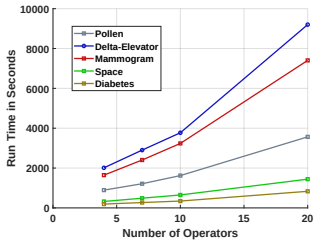


Figure 2.9: Varying Algebra

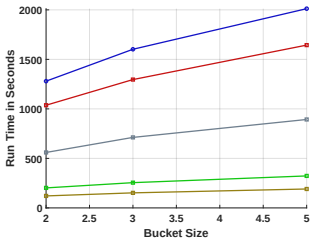


Figure 2.10: Varying Bucket Size

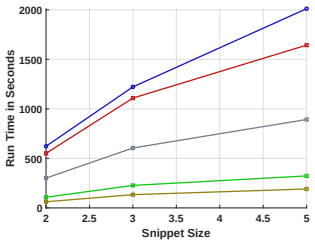


Figure 2.11: Varying Snippet Size

2.5.2 Summary of Results

Our experimental analysis answered the following key questions:

- **How did our proposed framework compare against fully automated or fully manual solutions?** We found that we scale significantly better ($10x - 20x$ faster than ExploreKit, $7x$ better than fully manual) than both of these extremes, with comparable AUC improvements (on an average 14%).

Data	EK-RF	Time (s)	FT-RF	Time (s)	MI-RF (s)	Time
pollen	10.56%	10233	-3.71%	40	-1.00%	42
delta_elevators	0.75%	37611	-1.13%	106	0.70%	180
mammography	-0.01%	16502	5.34%	113	4.00%	191
space	4.09%	10676	4.92%	106	4.20%	186
diabetes	4.46%	3474	3.11%	462	3.30%	390

Table 2.5: AUC improvement and runtime comparison with fully automated methods using RandomForest

Due to human involvement, it also avoided the worst case behavior of the automated methods (FeatureTools showed decrease in performance at times). Sections 2.5.3, 2.5.5 present these results.

- **How did our proposed algorithms compare?** We observed that we attained higher qualitative performance, while being scalable compared to other baselines. The exact algorithm `ExBKT` produced optimal buckets, while `RandomizedBKT` was much faster and produced results comparable to `ExBKT`. `Greedy` was inferior in quality. We focused on the coverage variant of `RandomizedBKT` so as to provide a diverse set of buckets to the human analyst. Algorithm `GSnippet` was interactive to the human analyst and works in real time. Section 2.5.4 presents these results.
- **What kind of attributes did our proposed framework generate?** We observed that the attributes produced by our framework were intuitive and meaningful to human analyst. Section 2.5.5 presents some of these results.

2.5.3 Comparison with fully automated methods

We compared the performance of our proposed approach against automated systems in both classifier performance and efficiency. Tables 2.4 and 2.5 show the results for SVM and RF classifiers respectively. On an average, the AUC improvement

of the classifiers were more than 14% through attribute design. Our proposed framework was almost 10x-20x faster than ExploreKit, comparable with FeatureTools (but FeatureTools caused sudden decline in AUC at times with the derived attributes).

2.5.4 Analysis of presented algorithms

In this section, we present quality and running time study of our algorithms with the baselines, described in Section 2.5.1. We note that the implemented baselines were inferior in AUC improvements, hence we only present running time of our solutions.

Comparison with Baselines. We began by comparing our algorithms for bucket construction : an exact algorithm `ExBKT`, a random walk based algorithm handling coverage `RandomizedBKT` and `Greedy`. Figures 2.3 and 2.4 show the results. As expected, `ExBKT` took substantial amount of time but gave the best results. `RandomizedBKT` was much faster and provided almost identical results for AUC improvement. `Greedy` was very efficient but qualitatively inferior.

Varying Number of Operators in the Algebra. We began by supporting arithmetic operators $(+, -, \times, /, \%)$ and then systematically added more logical, relational and aggregate operators. As expected, increasing the number of operators causes a slow down in our approach. Recall that our first stage of bucket generation was agnostic to both classifier and grammar. The resulting buckets often contained only a handful of attributes and hence our algorithm was quite fast even for a large number of operators. This can be seen in Figure 2.9 where the improvement is (sub)-linear in the number of operators. Figure 2.6 shows the corresponding impact on AUC. This shows that the additional operators often only provides negligible improvement in AUC. We conjuncture that for most attributes, only a small set of operators are most relevant.

Varying Bucket Size x. Next, we varied the maximum number of attributes x in each bucket. The impact of k (the number of buckets to return) was minimal. Our algorithms have a natural anytime property where it can be stopped at any time and pick the best k buckets. x affected the length of the random walk and had a major impact on runtime and AUC. Figures 2.7 and 2.10 show the result. As expected, increasing bucket size improved the AUC but it stagnated quickly. This confirmed with our hypothesis that most derived attributes often consist of few base attributes. The bucket size must not be too small - otherwise, we might miss meaningful group of attributes. It must also not be too large - otherwise, we might expend runtime for no meaningful improvement of AUC. We found a value of 5 to be good both from runtime perspective and the required cognitive impact on the human analyst. The runtime increased almost linearly with larger bucket size.

Varying Snippet Size j. In our final set of experiments, we varied the snippet size. Here we also noted that the impact of l (the number of snippets to return) was minimal. Figures 2.8 and 2.11 show the results. As expected, increasing snippet size had minimal impact on AUC. The improvement more or less topped out at snippet of size 3. This once again confirmed our hypothesis that derived attributes were often constructed from a handful of base attributes. The increase in time for larger snippet size was mostly linear.

Varying Number of Attributes. In order to highlight the scalability of our algorithms, we varied the number of base attributes by duplicating some of the base attributes randomly. Figure 2.5 shows the impact on running time of our exact and random walk based algorithm. As expected, the running time increased dramatically for `ExBKT` while the increase was marginal for `RandomizedBKT`. Our approach was scalable to large increase in the number of columns. The impact of increasing number of rows were rather minimal.

2.5.5 Comparison with fully manual method

We present a case study comparing a fully manual approach (a typical trial-and-error based attribute design exercise a data scientists goes through) and our approach. We used the popular Boston dataset from Kaggle [8] that seeks to predict house prices from from features like its area, number of bedrooms, etc.

Attribute	Interpretation
lstat	lower status of the population (percent).
tax	full-value property-tax rate per $10,000.
ptratio	pupil-teacher ratio by town.
dis	weighted mean of distances to five Boston employment centres.
rad	index of accessibility to radial highways.
black	$1000(Bk - 0.63)^2$ where Bk is the proportion of blacks by town.
crim	per capita crime rate by town.
rm	average number of rooms per dwelling.
zn	proportion of non-retail business acres per town.
nox	nitrogen oxides concentration (parts per 10 million).

Table 2.6: Boston home dataset attributes

We involved a data scientist to construct the derived attributes and she came up with the following 6 additional derived attributes after extensive analysis. lstat / tax, ptratio / tax, dis * rad / tax, black * crim, rm * (zn + indus) and (nox *

[8] https://www.kaggle.com/c/boston-housing/data

indus) / tax. The interpretation of these attributes are presented in Table 2.6. The data scientist took about 10 hours to manually craft these 6 derived attributes, out of which 3 hours were needed to understand and explore the data. The remaining 7 hours were needed for a trial and error process, where the expert tried many possible derived attributes, analyzed the correlation, tweaked the attributes, and repeated the process. While another data scientist with similar expertise was guided by our framework, she took *only* 1 *hour* to craft the same 6 derived attributes. For both cases, we observed an AUC improvement of 14%. This case study anecdotally showcase that our proposed framework is capable to drastically reduce the latency (7 hour vs 1 hr, i.e, $7x$ improvement) by aiding the domain expert.

2.6 Related Work

Human-in-the-loop Query Answering : A growing number of systems make use of lay workers (known as crowdsourced workers) using commercial platforms (e.g., AMT and CrowdFlower) or for academic use. Examples of applications include not only sentence translation, photo tagging and sentiment analysis, but also query answering (CrowdDB [26], Qurk [27], Deco [28], sCOOP, FusionCOMP, MoDaS, CyLog/Crowd4U), entity resolution (such as CrowdER [29]), planning queries [30], perform matching [31], or counting [32]. A series of works [33, 34, 35, 36, 37] have proposed variations of crowdsourced join to address entity resolution task for query answering. Authors in [38, 39, 40, 41, 42, 43] have proposed various techniques of performing crowdsourced version of top-k item selection. Crowdsourcing based filtering has been extensively introduced in [44, 45]. Crowdsourced *find* focuses on selecting one or more qualified items [46, 47]. Unlike these works, we involve humans for attributes design, which occurs at a later stage of the data science pipeline — therefore these prior works do not extend to our problem.

Human-in-the-loop Supervised Modeling: Machine learning literature has involved humans to obtain labels [48, 49, 50, 51] primarily for the classification problem. Active learning involving humans (expert as well as lay workers) has also been discussed in recent works such as [52, 53, 54, 55]. These works utilize humans for accurately predicting class labels, thus the humans function as an added module to help the algorithm steer toward the correct assignment of class labels. The main distinction between these works and ours is that, these works leverage human mainly for data labeling and not for attribute design.

Attribute Design: How to develop automated methods for designing/engineering attributes (commonly referred to as as feature engineering) has been discussed in recent works [18, 17, 19, 56, 57, 58], primarily in machine learning literature. Attribute engineering tools, such as, ExploreKit, Featuretools, Data Science Machine, One Button Machine [18, 17, 59, 60, 61, 19] rely on fully automated approaches. Typically they take longer to run and do not offer adequate explainability or intuition, as the discovered attributes remain opaque to the human analyst interested in broader ad-hoc data exploration. They also ignore the availability of humans to guide their exploration. Finally, the effectiveness of most of these tools depend on the underlying classification model, whereas, we present a model agnostic approach. Nevertheless, we use some of these tools in our experimental analysis for comparison purposes. A very few recent works [1, 2, 3] propose a conceptual framework and describe how to involve humans for designing attributes to improve the accuracy of predictive machine learning models, such as classifiers. While we borrow inspiration from these recent works [1, 2, 3], we present the framework with mathematical rigor and investigate optimization opportunities.

2.7 Final Remarks

We present an optimization guided semi-automated model-agnostic "human-in-the-loop" framework for designing derived attributes by leveraging humans. The main contribution of our work is to provide an optimization guided framework for data scientists that will help them to craft meaningful derived attributes much quicker than a fully manual method. On the other hand, running time of the fully automated techniques vary greatly; for example, FeatureTools is significantly faster compared to ExploreKit, but they all have the limitations of producing derived attributes that are opaque. Contrarily, the derived attributes produced by our framework are interpretable.

We present two computational problems inside our proposed framework - *top-k buckets design* and *top-l snippet generation* problems. We present effective solutions for solving both problems. We compare our proposed approach with two fully automated state-of-the-art tools, as well as fully manual domain expert designed derived attributes. Our rigorous quality evaluations using 6 real world datasets demonstrate that we are as effective as *fully automated methods* and we scale up significantly better compared to a fully manual solution involving two domain experts. Our scalability results demonstrate that both bucket design and snippet generation are interactive and ensure real time response with the analysts. As an ongoing problem, we are investigating how to revisit these problems when the datasets contain significant missing values. We are also investigating how to involve multiple domain experts in buckets design and snippets generation problems.

CHAPTER 3

iFE: Interactive Feature Engineering

Feature engineering is an important step in the data science pipeline. Derived attributes can play crucial role in the accuracy of machine learning tasks. Hence a human analyst usually makes several passes through the dataset to come up with most effective derived attributes which is a fully manual procedure. On the other hand, fully automated procedures can provide a set of derived attributes to the analyst quicker than the manual method, but this set may contain many opaque derived attributes that are not good for interpretability.

We demonstrate iFE, a desktop application that enables a human analyst to find interpretable derived attributes much quicker than fully manual method. The system first finds k buckets, each containing promising choices of raw attributes that the analyst can focus on only without having to look at all raw attributes. To achieve this, iFE implements a random walk based heuristic algorithm that is intuitive and works well in practice. In the next step, the system iteratively aids the analyst to generate top-l derived attributes within a bucket using arithmetic, relational, and logical operators. The user interface in our system guides the analyst to the final derived attributes in a few number of iterations which saves time and effort as well as boost productivity for the analyst.

3.1 Introduction

Attribute design (also known as feature engineering) is one of the most challenging aspects of a data science pipeline. The raw attributes often need to be transformed into

derived attributes, are more effective for building predictive models such as classifiers. For example, Diabetes Pedigree Function is an indicator of the likelihood of diabetes based on family history. A young person having larger score for this function is more likely to be diabetic than an older one with smaller score. A derived attribute"Age × DiabetesPedigreeFunction" can indicate this relationship which is not present in the raw diabetes dataset. Current attribute design techniques fall into one of these two extremes: (a) *fully manual* - slow and heavily reliant on domain expertise, often requiring data scientists to go through a repetitive trial-and-error exercise until the set of designed attributes are satisfactorily effective or (b) *fully automated* techniques (some notable systems are, Data Science Machine [17][1], ExploreKit [18][2], One Button Machine [19], or Featuretools[3]). Such methods are not model agnostic, and require substantial time to identify derived attributes, often opaque to the human analyst. Figure 3.1 shows the steps of generating derived attributes using fully automated methods. If data analyst does not specify specific attributes and use all the attributes in a dataset, there will be hundreds of derived attributes by combining the raw attributes using only basic arithmatic operators. Selecting a small set of derived attributes from this large list is a challenging task for the analyst.

In this paper, we present a demo that implements the techniques proposed in [5]. Our system enables an analyst to perform feature engineering effectively. Given a set of raw attributes, the analyst tries to predict the target variable. We introduce the concept of bucket and snippet to perform this task. A bucket is a small subset of raw attributes from which new features can be engineered. It is a way of dealing with very high dimensional data by focusing the analyst's attention on such small

[1]https://people.csail.mit.edu/kalyan/dsm/

[2]https://github.com/giladkatz/ExploreKit

[3]https://docs.featuretools.com/#minute-quick-start

but promising subset. Raw attributes in a bucket can be combined using different algebraic operators. Each such combination in a bucket is defined as a snippet. Our framework defines a mechanism to score each bucket and snippet, rank them according to the scoring function, and derive top-k buckets and top-l snippets by applying novel algorithms. *Mutual Information* (MI) is used as the scoring function in this regard. In some sense, buckets and snippets are a way of "horizontally compressing" big and wide datasets. After the analyst loads the dataset, the system operates in two stages to suggest the derived attributes to her. In the first stage, system runs the top-k bucket generation algorithm to provide top-k buckets of raw attributes to the analyst. The analyst can focus on creating derived attributes using attributes from a given bucket. The system is generic and allows creation of derived attributes using arithmetic, relational, and logical operators. Furthermore, it can even suggest top-l snippets of derived attributes to her. The analyst can choose snippets from each bucket to come up with the final set of derived attributes. The system provides a final result that lists classification accuracy for raw and derived attributes, and the accuracy improvement measure from using the derived attributes. Figure 3.2 depicts the steps of derived attribute generation using buckets and snippets.

iFE implements a semi-automated "human-in-the-loop" framework proposed in [5]. MI is used as the model agnostic measure; the framework proposes to produce derived attributes that have high MI with the target variable (or the class label). MI is known to have several desirable properties that make it an ideal candidate for adopting as a model agnostic measure in the system framework [5]. The important aspect of the framework is the investigation of how to optimize the involvement of humans in the attribute design process and ensure their efforts are successful. *iFE* implements the algorithms proposed by the framework and provides an intuitive user

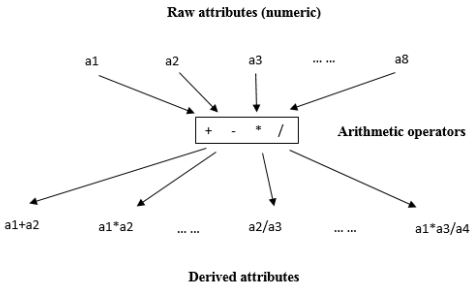


Figure 3.1: Derived Attribute Generation in other Framework

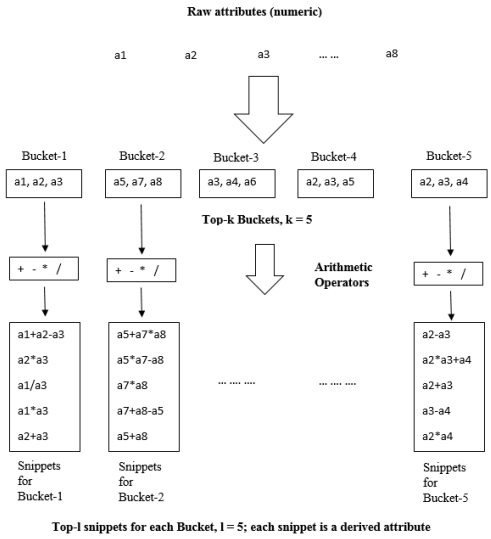


Figure 3.2: Derived Attribute Generation in *iFE*

interface through which a data scientist/analyst can easily perform the task of derived attribute generation in few steps.

3.2 System Overview

Figure 3.3 demonstrates the architecture of *iFE*. *iFE* is implemented as a desktop application. The system consists of two main components - User Interface and Backend. Design and functionality of each component is described in the following subsections.

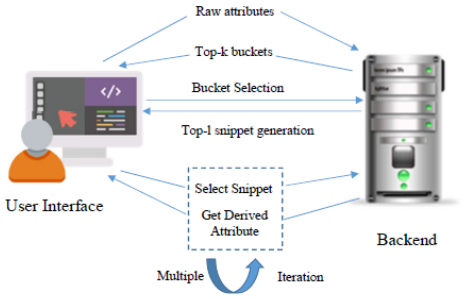
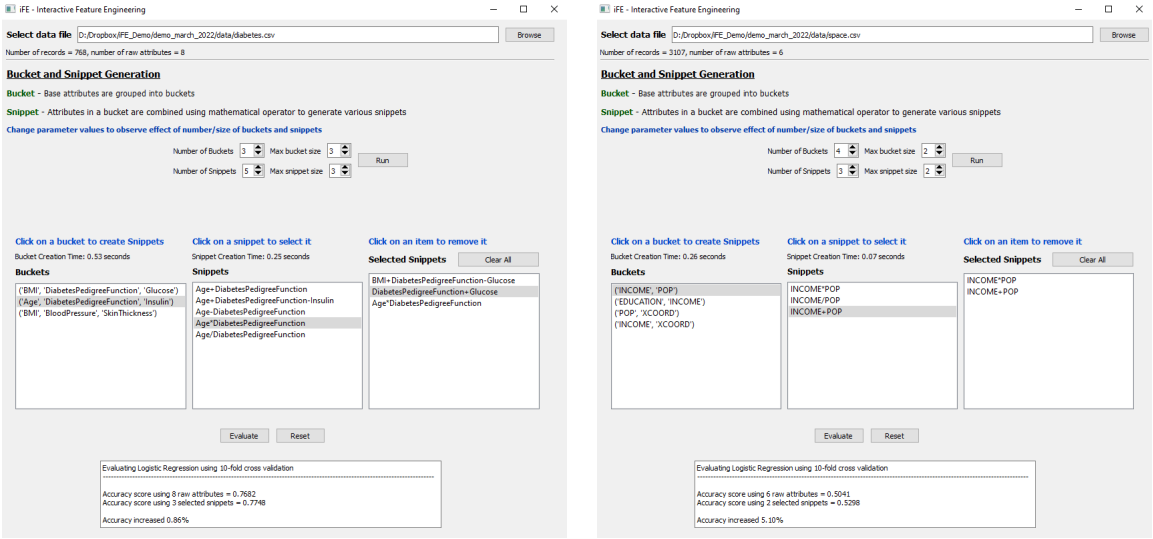


Figure 3.3: Architecture of *iFE*



(a) Demo Scenario 1

(b) Demo Scenario 2

Figure 3.4: User Interface

3.2.1 User Interface

iFE has an interactive user interface to input datafile and generate desired derived attributes in a few steps. User interface is demonstrated in Figure 3.4. First the user selects the datafile from computer by clicking *Browse* button and system provides an overview of the number of records and columns in the dataset. After that user inputs various parameters to generate buckets and snippets. System provides default values of the parameters which the user can change according to her preference. User can select number of buckets, maximum size of a bucket, number of snippets

and maximum snippet size. The number of buckets refer to the value k for generating top-k buckets, max bucket size is the value of x which is the maximum number of raw attributes in a bucket, number of snippets is the value of l for generating top-l snippets for each bucket, and the max snippet size refers to maximum expression size for a snippet. System first generates top-k buckets, and then top-l snippets for each bucket. After user clicks on *Run* button, system first generates top-k buckets and displays in the *Buckets* list. Then user clicks on any bucket and system generates top-l snippets for that specific bucket and renders the *Snippets* list. User can click on any snippet and system adds that to *Selected Snippets* list. Clicking on a selected snippet in the *Selected Snippets* list will remove that entry from the list, which user can add again by clicking on that snippet from the *Snippets* list. User can also remove all the selected snippets by clicking *Clear All* button on top of the *Selected Snippets* list. User can select snippets from different buckets and evaluate the effectiveness of the selected snippets by clicking *Evaluate* button. The result window is rendered below the *Evaluate* button displaying the accuracy using default raw attributes in the dataset and the accuracy using only selected snippets. Besides, increase or decrease of accuracy percentage is shown in the result which helps the user to understand the efficacy of her choice. *Reset* button provides option to clear the *Buckets*, *Snippets* and *Selected Snippets* lists and run the process using updated parameter settings. The interactive feature to add or delete any number of snippets and evaluate the selected snippets on the fly provides a useful tool for the user for feature engineering.

3.2.2 Backend

The backend runs the algorithms to generate top-k buckets and top-l snippets. We implement a random walk based algorithm for bucket generation that is highly

efficient and works well in practice. For snippet generation, a greedy algorithm is implemented that generates a snippet bottom up in each step [5].

3.2.3 Top-k bucket generation Algorithm

Two algorithms have been developed in [5] to generate Top-k bucket - exact and random-walk. The exact algorithm runs in a bottom-up fashion starting with singleton base attributes as buckets and gradually walking up the lattice level by level [5]. The random-walk based algorithm works well in practice which has been implemented in this demonstration. The core idea of the random walk algorithm is to compose a bucket of size x, by performing a random walk on the attribute lattice [5]. Here, attribute lattice is the power set of attributes. An attribute is added to a bucket by using weighted sampling without replacement. The weight of an attribute is based on the optimization criteria. A bucket is formed by performing a random walk on the attribute lattice, until its maximum size x has been reached. This random walk is repeated, until k-unique buckets have been derived. For the *Top-k buckets Design* problem, the weight of an item is directly proportional to its MI, i.e., $weight(A) = MI(Z, A)$ where Z is the target variable. The weight of an attribute is a ratio; it is proportional to its MI, but inversely proportional to the number of times it is present in other buckets that have been computed thus far. The algorithm terminates when we obtain top-k unique buckets.

3.2.4 Top-l snippets generation Algorithm

A greedy algorithm is implemented to generate top-l snippets. It generates a snippet bottom up in each step. Given a bucket b, it first sorts the base attributes in the bucket in descending MI. Then given j (the number of additional attributes to

extend a partially created derived attribute d'_i), it finds the top-j attributes in b that are not in d'_i. These are the candidate set of attributes for the snippets that involve d'_i. After that, it attempts to combine these j attributes considering all the operators in $\mathcal{L}$ (let, $\mathcal{L} = \{+, -, \times, /\}$) with d'_i. It ranks each of these created combinations and produces the l-combinations as snippets that have the top-l MI score with Z.

3.2.5 Technical Challenges

Parallel Processing A real time implementation of *iFE* dealing with large number of dataset requires large execution time. This issue is addressed by adopting parallel processing. The option of parallel processing greatly reduces the execution time and provides near real time response for moderately large set of data.

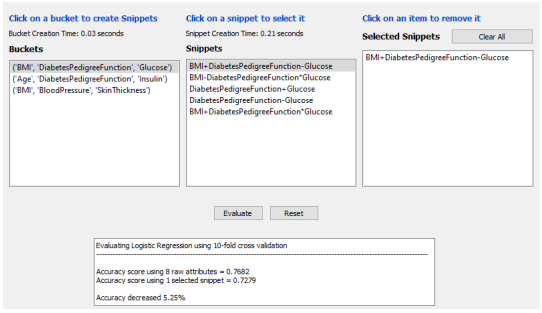
Caching When number of raw attributes increases, the attribute lattice to explore also increases exponentially. Using caching helps to avoid recalculating already existing MI score for a bucket of features. Hence caching is incorporated in the implementation for *iFE*.

3.3 Demo Plan

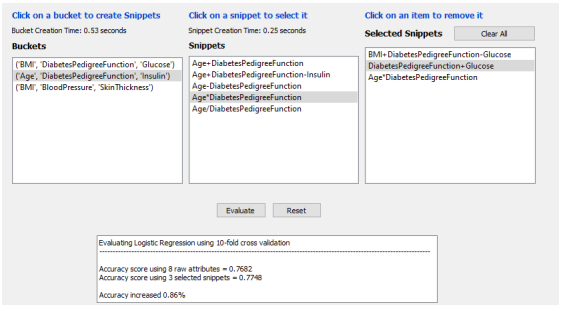
In this section we describe the system setup and demonstration scenario for *iFE*. First we briefly describe the hardware and system setup. We then provide two case studies for user demonstration.

3.3.1 Hardware and System Setup

Python 3 and PyQt5 has been used to develop *iFE* desktop application. It can run on Windows/Mac/Linux. To address the technical challenge of parallel process-



(a) Evaluation for single snippet



(b) Evaluation for multiple snippets

Figure 3.5: Interactive Output

ing, *iFE* can be configured to use Dask [4] library. By using Dask we can enable efficient parallel computations on single machines by leveraging their multi-core CPUs as well as run our code on a distributed cluster if needed. In this demonstration, numpy [5] library functions have been used in the code to utilize all the available cores of the single machine processor to mimic the parallel processing scenario.

3.3.2 Demonstration Scenarios

We provide two demonstration scenarios for generating derived attributes. User of this system is data analyst/ data scientist. The scope of the demo is restricted to dataset with numeric attributes where the target attribute indicates the class of the instance.

3.3.2.1 Demo Scenario 1

Figure 3.4(a) demonstrates this scenario. User first loads the Pima Indian Diabetes dataset [6] from the computer using *Browse* button. User then proceeds

[4]http://docs.dask.org/en/latest/why.html

[5]https://numpy.org/

[6]https://www.kaggle.com/uciml/pima-indians-diabetes-database

to select parameter values for bucket and snippet generation. User selects 3 and 5 as the value for Number of Buckets(k) and Number of Snippets(l) respectively, and 3 as the value of Max bucket size and snippet size. User then clicks on *Run* button. System generates top-3 buckets of attributes each having at most 3 attributes and displays in *Buckets* list. Here the first bucket has the largest MI score and buckets are sorted in descending order of MI. Figure 3.5(a) demonstrates the scenario where user clicks on the first bucket and system generates top-5 snippets according to MI score and renders *Snippets* list. In this demonstration, we use 4 operators $\{+, -, \times, /\}$; system combines bucket members using these operators to generate snippets for the selected bucket. Next, user clicks on the first snippet and system adds it in the *Selected Snippets* list. User clicks on *Evaluate* button and system displays the result. Here user finds that the accuracy decreases 5.25% for using selected single snippet instead of 8 raw attributes. A logistic regression model is evaluated using 10-fold cross validation of the input data. User proceeds to select different buckets and snippets from these buckets and evaluate the result on the fly. Figure 3.5(b) demonstrates the interactive scenario where user selects third snippet 'Age $\times$ DiabetesPedigreeFunction' and evaluates the result. Using the three snippets the accuracy is increased by 0.86%. User observes that she gets a slightly better accuracy using 3 snippets instead of 8 raw attributes and might decide to use these snippets for further model building activities in the data science pipeline. In this way, *iFE* enriches user experience through interactive interface to generate useful derived attributes in few steps.

3.3.2.2 Demo Scenario 2

Figure 3.4(b) demonstrates this scenario. User loads the space dataset [7] and proceeds to select number of buckets and snippets different from Demo Scenario 1. User selects the first bucket and then selects 2 snippets from the first bucket. User then clicks on *Evaluate* button and observes that accuracy increase is 5.10% for using these 2 snippets instead of 6 raw attributes. User decides to use these 2 snippets and does not proceed further.

3.4 Conclusion

We propose to demonstrate *iFE*, a semi-automated desktop application that can help a data scientist/analyst quickly determine important derived attributes. The main contribution of our work is to implement an optimization guided framework for data scientists that will help them to craft useful derived attributes much quicker than a fully manual method. On the other hand, running time of the fully automated techniques vary greatly [5], and they have the limitations of producing large list of derived attributes. Contrarily, the list of derived attributes produced by our framework is customizable though user interaction. Using Dask parallelization feature, the system can scale up to multi node clusters as well as scale down to single node multi-core machine. Hence it can be useful for data scientists to determine meaningful important derived attributes from a dataset in a quick and effective way.

[7] https://www.openml.org/d/737

CHAPTER 4

Efficient Approximate Top-k Mutual Information Based Feature Selection

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.

4.1 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 technique is the filtering based methods [62, 63], which leverages scoring functions involving statistical property of the data to select features. *Mutual Information* (MI) [4, 64] is one such popular measure [4] and has been extensively studied in recent work [6, 7, 5], due to its information theoretic interpretation, and ability in quantifying the predictive power of the attributes in a model agnostic fashion.

Problem Settings and Challenges. 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 4.2)** . Our setting expects the attributes to exhibit *high skew*, that is, *there exits a handful of x values that perfectly identify the corresponding z values, thereby finding those x values can reduce uncertainty in Z substantially.* This is indeed common in real world, as there exists only a handful of fraudulent transactions in the mix of a large number of valid ones, and these fraudulent ones exhibit some property (feature values) that are not present otherwise, or the presence of a very small number of Native Americans in a dataset that are afflicted disproportionately with a number of chronic illnesses, such as, Type 2 Diabetes, alcohol abuse, and suicide. Finally, such process of filtering based

feature selection is to be repeatedly performed during data exploration, hence our focus is to select top-k features with high accuracy by enabling interactive response time.

Contribution (1) - Connecting Functional Dependency with MI Approximation. A functional dependency (FD) **(Section 4.2)** in a relation R is an expression $X \rightarrow Z$, where $\{X, Z\} \subseteq R$, which holds, if all pairs of tuples that agree on X, agree also on Z. *While FD is studied from the perspective of checking integrity constraints in a database, we realize there does exist a subtle connection between FD and MI, as in if $\{X, Z\}$ exhibits full functional dependency, then they also have the "perfect" MI.* However, full functional dependency (FFD) is unlikely to to be present in real data, thus our effort is to quantify a "non-perfect" or *approximate functional dependency* (AFD) between the predictive attributes (or features) and the class label. We are aware of of a variety of works that compute approximate functional dependency [8, 9, 10, 11, 12, 13]. These works focus on application of AFD in schema design and finding interesting relationship from data, but not for feature selection. *We take inspiration from these works to design our proposed measure, as none of these works study FD in the context of MI estimation, which solely is our focus here.* **Section 4.6** contains further details.

Contribution (2)- A New measure to approximate MI. We propose a new measure *Attribute Average Conflict* **(Section 4.3)** that effectively approximates MI, while being much faster computationally. One can notice that the actual MI formula (Equation 4.1) 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. *Attribute Average Conflict (Aac)* captures the minimum number of violations of each value of $x \in X$, eliminating which a perfect "one-to-one correlation" could be established between X and Z, and *weigh* that by considering the "relative frequency"

of the corresponding x value. This way, the attribute X with the smallest Aac is likely to have the highest $MI(X, Z)$. Hence ordering attributes with respect to ascending Aac should provide us with the similar (if not identical) list of attributes with respect to descending MI. Interestingly, this approximation significantly improves the underlying computational process, as MI between X, Z could be estimated just by counting the co-occurrence frequencies of different (x, z) values, leading to significant speed up. Thus, some of the expensive operations (such as logarithm and division) that are present in standard MI calculation is completely avoided in this way.

Aac bears some similarity with $G3 - error(X, Z)$ [8], that is a known *error measures* to quantify approximate functional dependency(AFD) between X and Z. It captures the minimum number of records that violates the functional dependency criteria and must be deleted to satisfy functional dependency between $X \rightarrow Z$. However, unlike $G3 - error$, Aac also captures the relative frequency of each x value as its weight, leading to a different expression and outperforms $G3 - error$ empirically. Effectiveness of our proposed measure is demonstrated through monte-carlo [65] simulation **(Section 4.3.4)**.

Comparison with sampling. An obvious approach to enable efficiency is through data sampling, and there exist notable recent works [6, 7] that perform uniform random sampling adaptively on the data to efficiently compute MI to satisfy an error bound. Naturally, the effectiveness of such sampling based approaches are heavily reliant on the underlying data distribution. In fact, when the data distribution is skewed, uniform random sampling is known to perform poorly [66], as it misses out capturing the "rare" instances from the data, which is purely our focus here. In a nutshell, what we argue and empirically demonstrate is that our proposed technique for MI estimation gives rise to faster and more accurate feature selection methods, compared to the uniform random sampling based methods [6, 7]. Nevertheless, one

can potentially apply our proposed approach on the sampled data. In that sense, our work complements any sampling based MI estimation approach.

Contribution (3)- Experimental Evaluations We perform extensive evaluations **(Section 4.4)** 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[6]. 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[6] 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 the upper and lower bounds of the proposed measure *Aac* **(Section 4.5)** 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 be scanned to compare *Aac* between two attributes 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.

4.2 Preliminaries and Definitions

Example 2. *We describe a toy example in Table 4.1 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.*

recordId	*Ethnicity*	*Age* < 30	*PlaysBasketBall*
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 4.1: Running Example 2

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

Table 4.2: Measures for Example 2

4.2.1 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 2, $n = 10$,$m = 2$, $\mathcal{A} = \{Ethnicity, Age < 30\}$, $X_1 = Ethnicity$, $X_2 = Age < 30$, $Z = PlaysBasketball$.

Mutual Information (MI) [67]: Mutual Information (MI) captures information theoretic "correlation" [22] 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:

$$MI(X, Z) = \sum_{x \in X} \sum_{z \in Z} p(x, z) \log \frac{p(x, z)}{p(x)p(z)} \tag{4.1}$$

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 $MI(X)$ for brevity instead of $MI(X, Z)$. Using entropy [67] of Z denoted as $H(Z)$ and conditional entropy [67] between Z and X denoted as $H(Z|X)$, $MI(Z, X)$ is defined as

$$MI(X, Z) = MI(Z, X) = H(Z) - H(Z|X) \tag{4.2}$$

MI is symmetric [67], that is $MI(X, Z) = MI(Z, X)$ which is stated in equation (4.2). In Example 2, $MI(Ethnicity, PlaysBasketball) = 0.1916$, and $MI(Age < 30$, $PlaysBasketball) = 0.2812$. Next, we define 3 key terms from Database literature.

Functional Dependency (FD) [68]: A functional dependency between X and Z, denoted by $X \rightarrow Z$, is a constraint on the possible tuples that can form a

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

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 2, $Ethnicity = Black$ is associated with $PlaysBasketBall = Yes$ in r1,r2,3, 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) [69]: 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 $e(X \rightarrow Z) = \min\{|s||s \subseteq r \text{ and } X \rightarrow Z \text{ holds in } r \setminus s\}$ $/|r|$ [69]. In Example 2, $Ethnicity$ and $Age < 30$ determine values of $PlaysBasketball$ for records r4 through r10, but records r1,r2, and r3 violates FD. We say that AFD holds from $Ethnicity$ and $Age < 30$ to $PlaysBasketball$ respectively.

G3-error [8]: The number of tuples need to be deleted from relation r to achieve FD is defined as G3-error [8].

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

In Example 2, $G3 - error(Ethnicity \rightarrow PlaysBasketball, r) = 3$, $G3 - error(Age < 30 \rightarrow PlaysBasketball, r) = 3$.

4.2.2 Problem Statement

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 $MI(X_k, Z) \geq MI(X_j, Z) \forall j \in \{k+1, m\}$.

4.3 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.

4.3.1 MI calculation Steps

Taking a close look at equation (4.1), 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.

4.3.2 Proposed Measure: Attribute Average Conflict

There has been previous works in database and data mining community regarding various measures for Approximate Function Dependency (AFD) [8, 9, 10]. 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 (Equation 4) 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 4.2), 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 2, $AV - conflict(Ethnicity = Black) = 3$.

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.

$$AV - average - conflict(x_v) = AV - conflict(x_v) \times \frac{n_{x_v}}{n} \tag{4.3}$$

Here n_{x_v} denotes the number of records where X has value v. In Example 2, $AV - average - conflict(Ethnicity = Black) = 3 \times 0.8 = 2.4$;

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.

$$Aac(X) = \sum_{x_v \in X} AV - average - conflict(x_v) \tag{4.4}$$

In Example 2, $Aac(Ethnicity) = 2.4 + 0 = 2.4$; $Aac(Age < 30) = 1.8 + 0 = 1.8$. $G3 - error$ 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 $\{X_1, X_2\} \in \mathcal{A}$, $MI(X_1) > MI(X_2) \implies Aac(X_1) < Aac(X_2)$ and vice versa.

4.3.3 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 subsection 4.3.1. Aac provides us with the answer to the **Top-K()** query faster than the exact MI based method.

Algorithm 4 Algorithm `topK-Aac`

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

As illustrated in Algorithm 4, 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 4.4.

4.3.4 Monte-Carlo Simulation

We conduct a Monte-Carlo simulation [65] 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.

4.3.4.1 Model setup

Let $\mathcal{L}$ 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, $\mathcal{L} = 8$, 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.

y	*G3− error support %*	*G3− error contra diction %*	*Aac support %*	*Aac contra diction %*
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 4.3: Simulation results for binary values

Do main size	*G3− error support %*	*G3− error contra diction %*	*Aac support %*	*Aac contra diction %*
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

Table 4.4: Simulation results varying domain size; $y = 0.3$

4.3.4.2 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 at-

Algorithm 5 Algorithm `calcProbDist`

input: list $\mathcal{L}$ of possible attribute-value combination X_1,X_2,Z , y

output: Probability of each item in $\mathcal{L}$

$probList \leftarrow \{\}$

$counter \leftarrow 0$

$upper \leftarrow 1$

$totalProb \leftarrow 0$

$lenL \leftarrow$ number of elements in $\mathcal{L}$

$numZeroN \leftarrow y \times lenL$

while $\mathcal{L}$ is nonempty **do**

  $item \leftarrow$ randomly choose an element from $\mathcal{L}$

  if $counter <$ ⌈ numZeroN ⌉ **then**

    $varProb \leftarrow 0$

    $counter \leftarrow counter + 1$

  else

    $varProb \leftarrow$ choose a value uniformly at random between 0 to $upper$

  end if

  $probList[item] \leftarrow varProb$

  $totalProb \leftarrow totalProb + varProb$

  $upper \leftarrow 1 - totalProb$

  remove $item$ from $\mathcal{L}$

end while

Return $probList$

tribute 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 demon-

strate sparsity/skewness). For each run, with a specific zero fraction value $y\%$, we consider $\lceil \mathcal{L} \times y\% \rceil$ 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, $MI(X_1) > MI(X_2) \implies Aac(X_1) < Aac(X_2)$, 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 other competing methods.

Algorithm 5 illustrates the steps for each such assignment. The input is the list $\mathcal{L}$ of possible attribute value combinations ((0,0,0),(0,0,1),(0,1,0),... (1,1,1) for binary case) and y which is the fraction of total records that we want to assign 0 probability deliberately. We initialize $totalProb$ and $upper$ to 0 and 1 respectively. We choose an item uniformly at random from $\mathcal{L}$, choose a value $varProb$ uniformly at random between 0 and $upper$ and assign that to the item. We add probability of the selected item $varProb$ to $totalProb$, and update $upper$ by subtracting $totalProb$ from 1. We use a *counter* variable to track *item* from list that is assigned 0 probability. After the while loop ends, *probList* will contain all items from $\mathcal{L}$ along with respective probability, which is returned as output. The algorithm ensures the total probability of all items in $\mathcal{L}$ adds up to 1 and some fraction of records will get 0 probability.

4.3.4.3 Simulation results

Table 4.3 illustrates the simulation results for binary attributes. Here, $y = 0.6$ means that 60% of the possible attribute value combination are assigned 0 probability and the probability assignment for rest of the 40% combination 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 4.4 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.

4.4 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 this experiment and the numbers are presented as the average of 5 runs. 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.

4.4.1 Experimental Setup

Datasets

- Synthetic data: Two types of synthetic data is used in the experiment. First, we generate highly skewed binary dataset 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 4.5. Here the column $p(X = 0)$ and $p(X = 1)$ indicate the probability of 0 and 1 for the attribute respectively.

Next, we use artificially generated Madelon dataset from OpenML [2] repository. This dataset has 2600 records and 500 attributes, the target attribute is binary i.e., $Z = \{1, 2\}$. The dataset is not highly skewed, rather half of the records are assigned to $Z = 1$ and the other half are assigned to $Z = 2$. Madelon dataset was part of the NIPS 2003 feature selection challenge.

- Real world data: We report our findings on five real world datasets from UCI [3] , Keel [4] and OpenML[5] repository. The Datasets are summarized in Table 4.6.
  - Census Income [70]: This dataset contains census data extracted from the 1994 and 1995 population surveys conducted by the U.S. Census Bureau. Original dataset contains 40 attributes including integer and categorical types. For this experiment 28 categorical attributes are used and missing values are imputed using most-frequent values in the attribute. The

[2]https://www.openml.org/d/1485

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

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

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

target has binary class label indicating whether a person has income > $50K$ or not. Data source - http://archive.ics.uci.edu/ml/datasets/Census-Income+%28KDD%29

- Kick Car: There is a risk in purchasing a used car at an auto auction that the vehicle might have serious issues that prevent it from being sold to customers. This unfortunate incident is termed as "kicks" by auto community. 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. Missing values are imputed using most-frequent values in the attribute. Data source - https://www.openml.org/d/41162
- Connect-4: 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. Thus, every attribute contains a nominal value which describes if a given position is void or if it has been occupied by one player. The task is to predict which player is likely to win the match. Origianl 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), in this experiment. Data source - https://sci2s.ugr.es/keel/dataset.php?cod=193
- Penbased: This dataset contains attributes for Penbased recognition of handwritten digits. Original dataset has 16 integer attributes all of which are used in this experiment. Each attribute has domain [0,100], so the domain size is 101. The target class in original dataset has 10 labels for recognizing digits 0-9. In this experiment, the target is converted to have binary labels to recognize digit 4. If an instance rec-

ognized digit 4, the target is set to 1, otherwise to 0. Data source - https://sci2s.ugr.es/keel/dataset.php?cod=70

– Penbased-bin-3: This is the same Penbased dataset above, but each attribute is discretized to have domain [0-2], so the domain size of each attribute is 3. Rest of the configuration for the experiment remains same as the Penbased 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 4.5: Synthetic Dataset

Dataset	n	m
Census Income	299,285	28
Kick Car	72,983	17
Connect-4	67,557	42
Penbased	10,992	16
Penbased-bin-3	10,992	16

Table 4.6: Real Datasets

Implemented Baselines. Our proposed measure in Section 4.3 *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. [6]* We implement the *Swope* method proposed in [6] 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 4.

Performance Measures For computing the accuracy of the proposed approach, we present precision [71], ndcg-score [72], 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 $\epsilon = 0.5$ [6] , and for Urs sample size $= 0.65n$. Unless otherwise stated, speed up is presented wrt baseline MI.

4.4.2 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. For the second type of synthetic data without high skew (i.e. Madelon), *Aac* does not achieve perfect precision and ndcg-score, but the total-mi for the selected attributes is very close to the exact baseline MI. For speedup comparison, *Aac* is $1.65x$ faster than MI, $11x$ faster than *Swope*, and $1.18x$ faster than Urs, which demonstrates the superiority of *Aac* for Madelon dataset.

Similar observation holds in the real data experiments. *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 *Aac*. 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, Connect-4 and Pen-Based-bin-3, where most of the attributes ($> 50\%$) have small domain size (< 10). But for datasets where most of the attributes ($> 50\%$) have larger domain size (≥ 10), $G3$ achieves slightly better precision and ndcg-score than *Aac* which is observed for Kick car and Penbased datasets. For speedup comparison, on average *Aac* is $2x$ faster than MI, $1.3x$ faster than Urs and $6.7x$ faster than *Swope* across all 5 datasets. Thus, *Aac* turns out to be the unanimous choice considering both accuracy and efficiency.

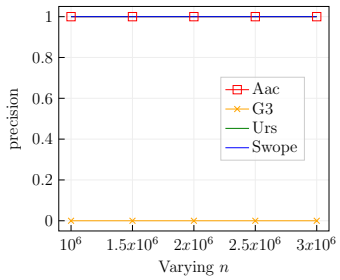


Figure 4.1: Precision; m=50, k=10

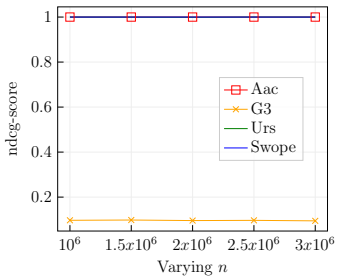


Figure 4.2: Ndcg-score; m=50, k=10

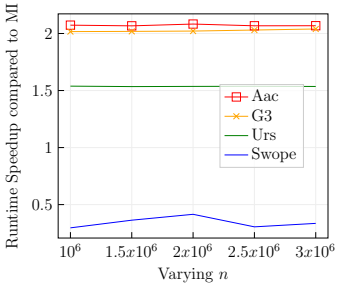


Figure 4.3: Speedup; m=50, k=10

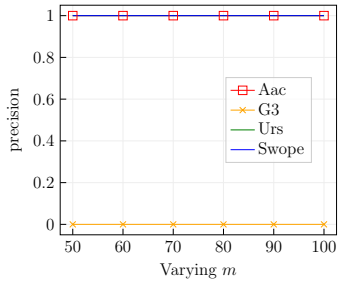


Figure 4.4: Precision; n=10^6, k=10

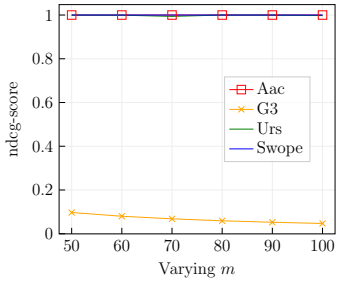


Figure 4.5: Ndcg-score; n=10^6, k=10

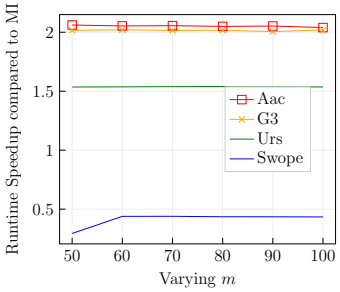


Figure 4.6: Speedup; n=10^6, k=10

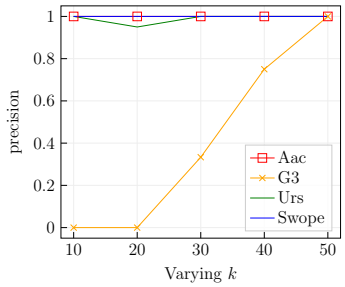


Figure 4.7: Precision; n=10^6, m=50

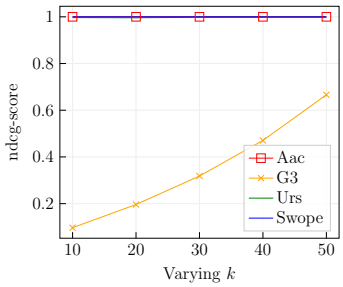


Figure 4.8: Ndcg-score; n=10^6, m=50

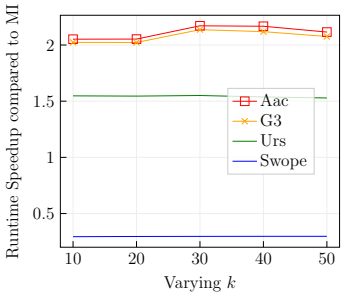


Figure 4.9: Speedup; n=10^6, m=50

4.4.3 Synthetic Data Experiments

4.4.3.1 Vary n

Figure 4.1 - 4.3 illustrate the results. **Quality.** Only *G3* shows poor precision and ndcg-score. *Aac*, *Swope* and *Urs* achieve perfect precision and ndcg-score. **Effi-**

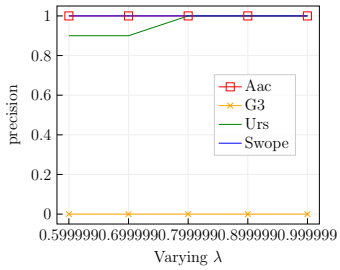


Figure 4.10: Precision; n=10^6, m=50,k=10

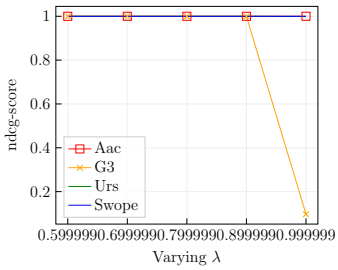


Figure 4.11: Ndcg-score; n=10^6,m=50,k=10

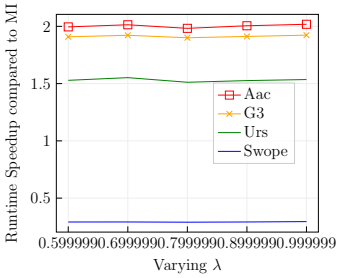


Figure 4.12: Speedup; n=10^6, m=50,k=10

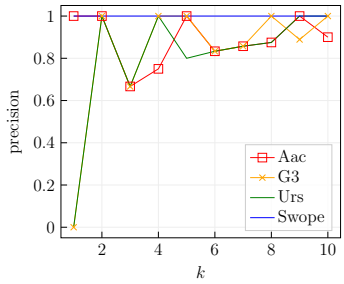


Figure 4.13: Precision:Madelon

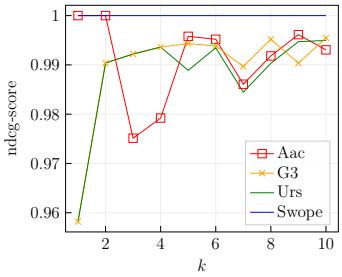


Figure 4.14: Ndcg-score:Madelon

ciency. *Aac* is $4x-6x$ times faster than *Swope*, $1.3x$ faster than Urs and has similar speedup compared to $G3$.

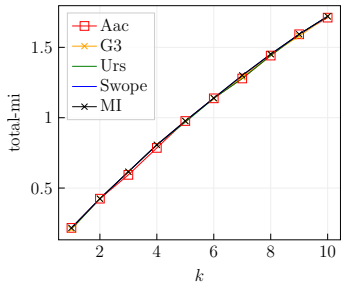


Figure 4.15: Total-MI:Madelon

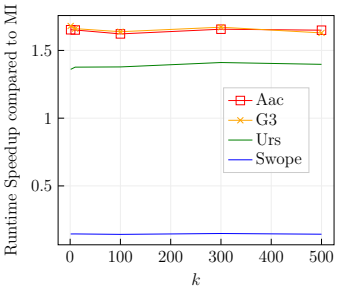


Figure 4.16: Speedup:Madelon

4.4.3.2 Vary m

Figure 4.4 - 4.6 illustrate the results. We observe similar result as of varying n for both quality and efficiency measures. **Quality.** *Aac*, *Swope* and Urs achieve perfect precision and ndcg-score whereas $G3$ shows poor precision and ndcg-score. **Efficiency.** *Aac* is $4x - 6x$ times faster than *Swope*, $1.3x$ faster than Urs and has similar speedup compared to $G3$.

4.4.3.3 Vary k

Figure 4.7 - 4.9 illustrate the results. **Quality.** *Aac* and *Swope* show perfect precision and ndcg-score. Urs has lower precision and ndcg-score for $K = 20$, which supports the fact uniform random sampling cannot proivde stable top-k attributes for highly skewed data. $G3$ has precision zero for $k < 25$ and precision increases with k and achieves perfect precision for $k = m$. Urs also shows lowest ndcg-score. **Efficiency.** *Aac* is $2x$ faster than MI, $4x - 6x$ times faster than *Swope*, $1.3x$ faster than Urs, and shows similar speedup compared to $G3$.

4.4.3.4 Vary skew λ

Figure 4.10 - 4.12 illustrate the results. **Quality.** Urs shows lower precision for $\lambda = 0.599999$ and 0.699999, and $G3$ has a precision of zero for various λ used in this setting. Both *Aac* and *Swope* show perfect precision and ndcg-score. **Efficiency.** *Aac* is $2x$ faster than MI, $4x - 6x$ times faster than *Swope*, $1.3x$ faster than Urs, and shows similar speedup compared to $G3$.

4.4.3.5 Madelon Dataset

Quality Precision, ndcg-score and total-mi is observed for Madelon dataset by varying k from 1 to 10 as shown in Figure 4.13 - 4.15. Precision and ndcg-score of our proposed method *Aac* is not perfect as the Madelon dataset does not have highly skewed distribution, but the total-mi for *Aac* is very close to basline which indicates that the selected top-k attributes have MI close to the attributes in the list returned by baseline MI method. *Swope* iteratively expands the sample size and scans all records in the final iteration, hence achieving perfect precision and ndcg-score. **Efficiency** Figure 4.16 shows the speedup for $Madelon$ dataset. *Aac* is $1.65x$ faster than MI, $11x$ faster than *Swope* , $1.18x$ faster than Urs. It has similar runtime and speedup compared to $G3$ for this dataset. The large speedup using *Aac* compared to *Swope* is due to the fact that *Swope* iteratively expands the sample size until it reads most of the data and in each iteration MI is calculated to meet some upper and lower bound criteria.

4.4.4 Real Data Experiments

4.4.4.1 Quality

We vary k from 1 to 10 and observe precision, ndcg-score and total-mi as for Census Income (Figure 4.17 - 4.19), Kick Car (Figure 4.21 - 4.23), Connect-4 (Figure 4.25 - 4.27), Penbased (Figure 4.29 - 4.31), and Penbased-bin-3 (Figure 4.33 - 4.35) . As these real world datasets do not have highly skewed distribution, precision and ndcg-score for *Aac* is not perfect. But the total-mi for *Aac* is close to baseline indicating that the selected top-k attributes have MI close to the attributes in the list returned by baseline MI method.

- *Census Income* Figure 4.17 - 4.19 show the qualitative evaluation for this dataset. There are 28 attributes out of which 11 attributes have domain size ≥ 10, that is, 39.2% attributes have domain size ≥ 10. Hence most of the attributes have domain size < 10. The largest domain size is 51, and the average domain size across all attributes is 14.4. Both *Aac* and $G3$ cannot achieve perfect precision and ndcg-score for this dataset, but *Aac* performs better than $G3$. Besides, *Aac* has better total-mi than $G3$ for this dataset. Urs and *Swope* achieve perfect precision and ndcg-score for this dataset.
- *Kick Car* Figure 4.21 - 4.23 show the qualitative evaluation for this dataset. There are 17 attributes out of which 11 attributes have domain size ≥ 10, so 64.7% attributes have domain size ≥ 10. Most of the attributes have domain size ≥ 10. The largest domain size is 1063 and aveargе domain size across all attributes is 142.58. $G3$ has better precision, ndcg-score, and total-mi than *Aac* for most of the k values. Both Urs and *Swope* show perfect precision and ndcg-score for this dataset.
- *Connect-4* Figure 4.25 - 4.27 show the qualitative evaluation for this dataset. There are 42 attributes each having domain size of 3. So, all the attributes have domain size < 10. *Aac* shows has better precision, ndcg-score and total-mi that $G3$ for this dataset. Precision and ndcg-score for Urs is not perfect. *Swope* shows perfect precision and ndcg-score.
- *Penbased* Figure 4.29 - 4.31 show the qualitative evaluation for this dataset. There are 16 attributes each having domain size of 101. So all the attributes have domain size ≥ 10. Here $G3$ shows better overall precision and ndcg-score than *Aac*, and total-mi is slightly better for $G3$ compared to *Aac*. Urs does not have perfect precision and ndcg-score for this dataset, whereas *Swope* has perfect precision and ndcg-score.

- *Penbased-bin-3* Figure 4.33 - 4.35 show the qualitative evaluation for this dataset. Here each of the 16 attributes are discretized to have bin size of 3, so all the attributes have domain size < 10. *Aac* is better than *G*3 in terms of precision, ndcg-score and total-mi for this dataset. *Urs* cannot achieve perfect precision and ndcg-score. *Swope* shows perfect precision and ndcg-score for this dataset as well.

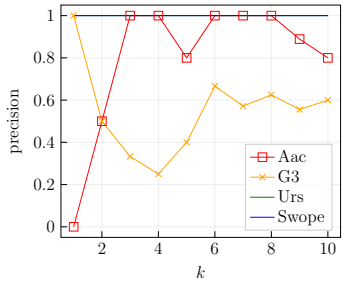


Figure 4.17: Precision:Census Income

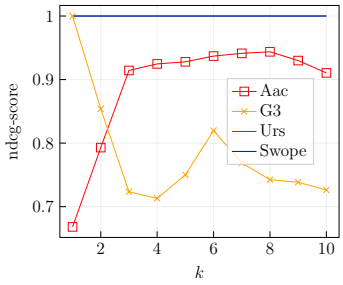


Figure 4.18: Ndcg-score:Census Income

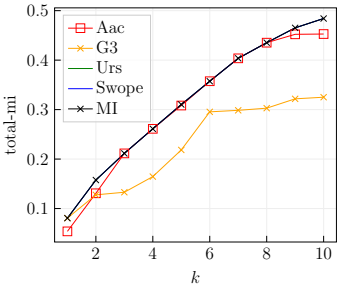


Figure 4.19: Total-MI:Census Income

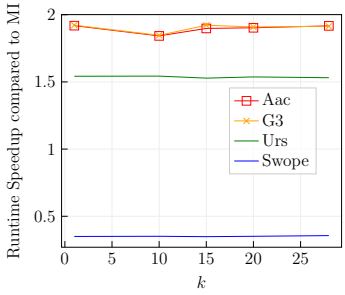


Figure 4.20: Speedup:Census Income

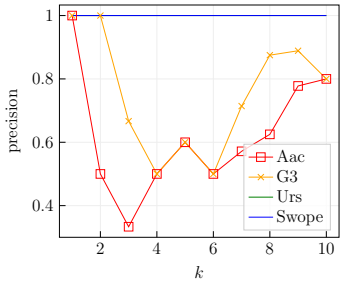


Figure 4.21: Precision:Kick Car

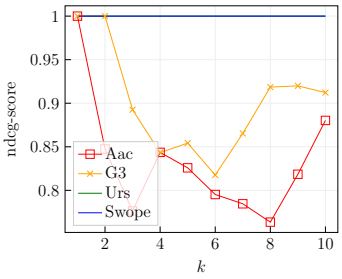


Figure 4.22: Ndcg-score:Kick Car

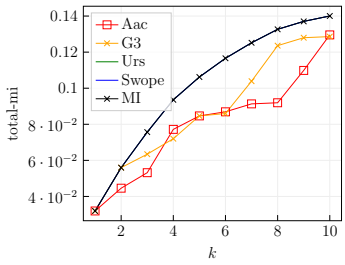


Figure 4.23: Total-MI: Kick Car

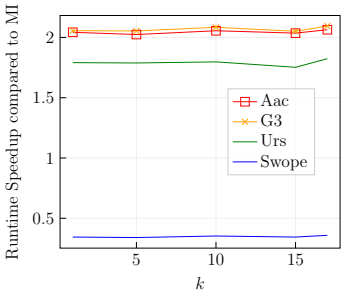


Figure 4.24: Speedup:Kick Car

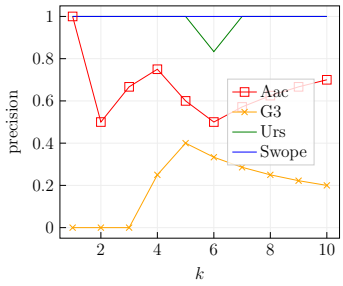


Figure 4.25: PrecisionConnect-4

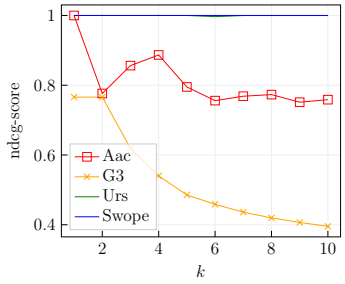


Figure 4.26: Ndcg-score:Connect-4

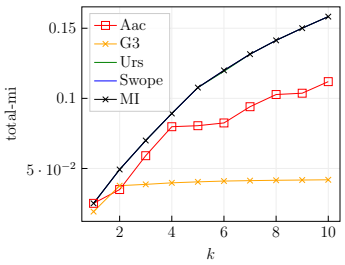


Figure 4.27: Total-MI: Connect-4

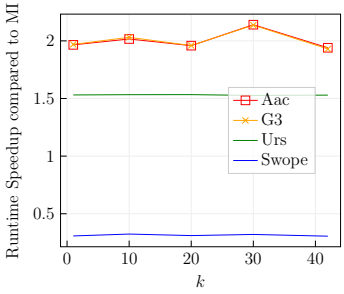


Figure 4.28: Speedup: Connect-4

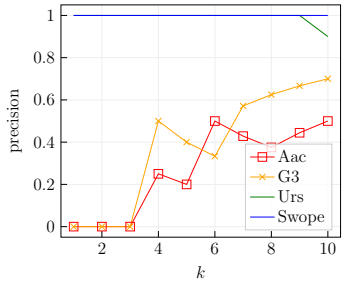


Figure 4.29: Precision:Penbased

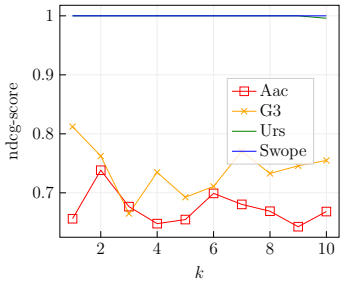


Figure 4.30: Ndcg-score:Penbased

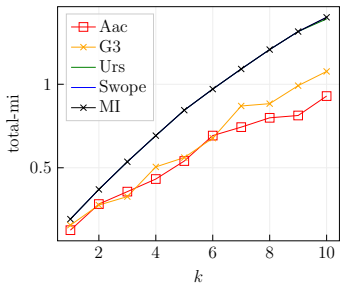


Figure 4.31: Total-MI:Penbased

4.4.4.2 Efficiency

We vary k from 1 to m with different interval for the 5 real datasets and compare speedup against baseline MI as shown in Figure 4.20, Figure 4.24, Figure 4.28, Figure 4.32, and Figure 4.36. *Aac* shows better speedup than MI,Urs and *Swope* for all the datasets, and has similar speedup compared to $G3$. The large speedup using

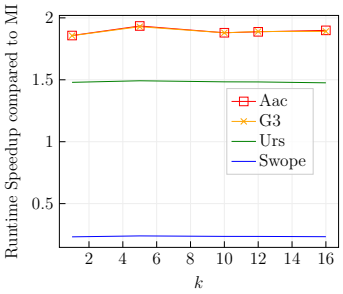


Figure 4.32: Speedup:Penbased

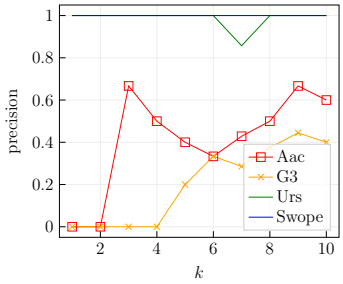


Figure 4.33: Precision:Penbased-bin-3

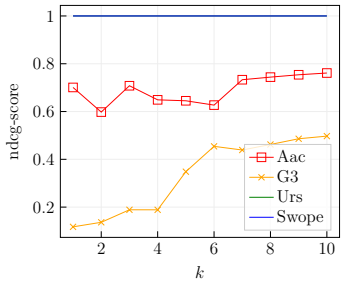


Figure 4.34: Ndcg-score:Penbased-bin-3

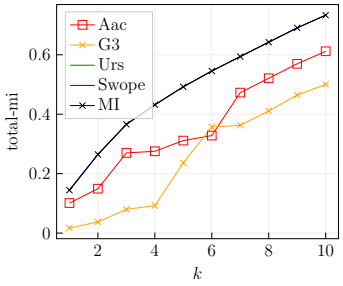


Figure 4.35: Total-MI:Penbased-bin-3

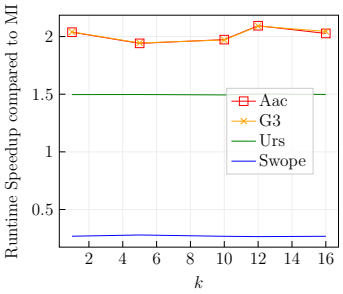


Figure 4.36: Speedup:Penbased-bin-3

Aac compared to *Swope* is due to the fact that *Swope* iteratively expands the sample size until it reads large number of records and in each iteration MI is calculated to meet some upper and lower bound criteria.

- *Census Income* Figure 4.20 shows the speedup evaluation. *Aac* is $2x$ faster than MI, $1.3x$ faster than Urs, $5.68x$ faster than *Swope* and has similar runtime compared to $G3$.
- *Kick Car* Figure 4.24 shows the speedup evaluation. *Aac* is $2x$ faster than MI, $1.15x$ faster than Urs, $5.87x$ faster than *Swope* and has similar runtime compared to $G3$.
- *Connect-4* Figure 4.28 shows the speedup evaluation. *Aac* is $2x$ faster than MI, $1.31x$ faster than Urs, $6.4x$ faster than *Swope* and has similar runtime compared to $G3$.

- *Penbased* Figure 4.32 shows the speedup evaluation. *Aac* is $1.9x$ faster than MI, $1.29x$ faster than Urs, $8x$ faster than *Swope* and has similar runtime compared to $G3$.
- *Penbased-bin-3* Figure 4.36 shows the speedup evaluation. *Aac* is $2x$ faster than MI, $1.35x$ faster than Urs, $7.5x$ faster than *Swope* and has similar runtime compared to $G3$.

4.5 Upper and Lower Bounds of New Measure

We investigate the 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 databases where the data dictionary, or metadata can be used to get the marginal frequencies of attributes and target variable.

Table 4.7 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. Theorem 4 and 5 provide the minimum and maximum values for $Aac(X, 0, 1, n)$.

Theorem 4. $Aac(X, 0, 1, n)_{min} = 0$

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 (4.3), $AV - average - conflict(x_i) = 0$, and using this in equation (4.4), $Aac(X) = 0$.

□

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 $X=i$
$f_{x_iz_0}$	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}	$min(f_{z_0}, f_{z_1})$
$Aac(X)_{max}$	Maximum value of Aac for attribute X
$Aac(X)_{min}$	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

Table 4.7: Notations used for deriving bounds

Lemma 4. *For $|Dom(X)|=1$, $Aac(X,0,1,n)_{max}$ is achieved if $f_{x_iz_0}=\frac{n}{2}$*

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 $f_{x_iz_0}=\frac{n}{2}$. 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 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, $Aac(X,0,1,n)=(\frac{1}{2}-\epsilon)f_{x_i}\times\frac{n}{n}=(\frac{1}{2}-\epsilon)f_{x_i}<\frac{1}{2}f_{x_i}$. Hence, $Aac(X,0,1,n)_{max}$ is acheived if $f_{x_iz_0}=\frac{n}{2}$. □

Theorem 5. $Aac(X,0,1,n)_{max}=\frac{n}{2}$

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. Let, $f_{x_0} = n_1$, $f_{x_1} = n_2$. From Lemma 4, $AV - average - conflict(x_0)_{max} = \frac{n_1}{2} \times \frac{n_1}{n} = \frac{n_1^2}{2n}$,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.

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.

$$\begin{aligned} UB - Aac(X, l, 1, n) &= Aac(X, l, 1, l) + Aac(X, l, l + 1, n)_{max} \\ &= Aac(X, l, 1, l) + \frac{n - l}{2} \end{aligned} \tag{4.5}$$

$$\begin{aligned} LB - Aac(X, l, 1, n) &= Aac(X, l, 1, l) + Aac(X, l, l + 1, n)_{min} \\ &= Aac(X, l, 1, l) \end{aligned} \tag{4.6}$$

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

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

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 (4.7)

$$
l > n - 2[Aac(X, l, 1, l) - Aac(Y, l, 1, l)] \tag{4.7}
$$

The upper and lower bounds in equation (4.5) and (4.6) 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 (4.5) and (4.6) might not work well in practice, resulting in a large value for l close to n in equation (4.7). Next, we derive tighter upper and lower bounds.

4.5.1 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

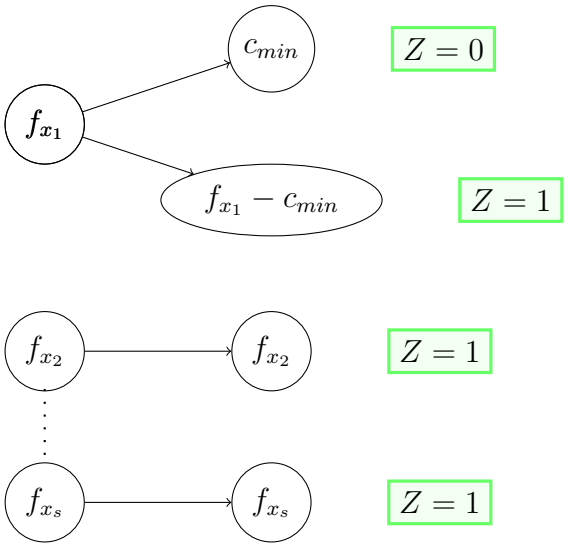


Figure 4.37: Arrangement for $Aac(X)_{max}$ when $c_{min} \leq \frac{f_{x_1}}{2}$

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.

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 5. *if* $f_{x_1} > f_{x_2}$, *then* $AV - avearge - conflict(x_1)_{max} > AV - avearge - conflict(x_2)_{max}$

Proof. From Lemma 4, $AV - avearge - conflict(x_1)_{max} = \frac{f_{x_1}}{2} \times \frac{f_{x_1}}{n}$, and $AV - avearge - conflict(x_2)_{max} = \frac{f_{x_2}}{2} \times \frac{f_{x_2}}{n}$. As $f_{x_1} > f_{x_2}$, hence $AV - avearge - conflict(x_1)_{max} > AV - avearge - conflict(x_2)_{max}$. □

Using Lemma 4 and 5 we can derive the following equation for $Aac(X, l, l + 1, n)_{max}$

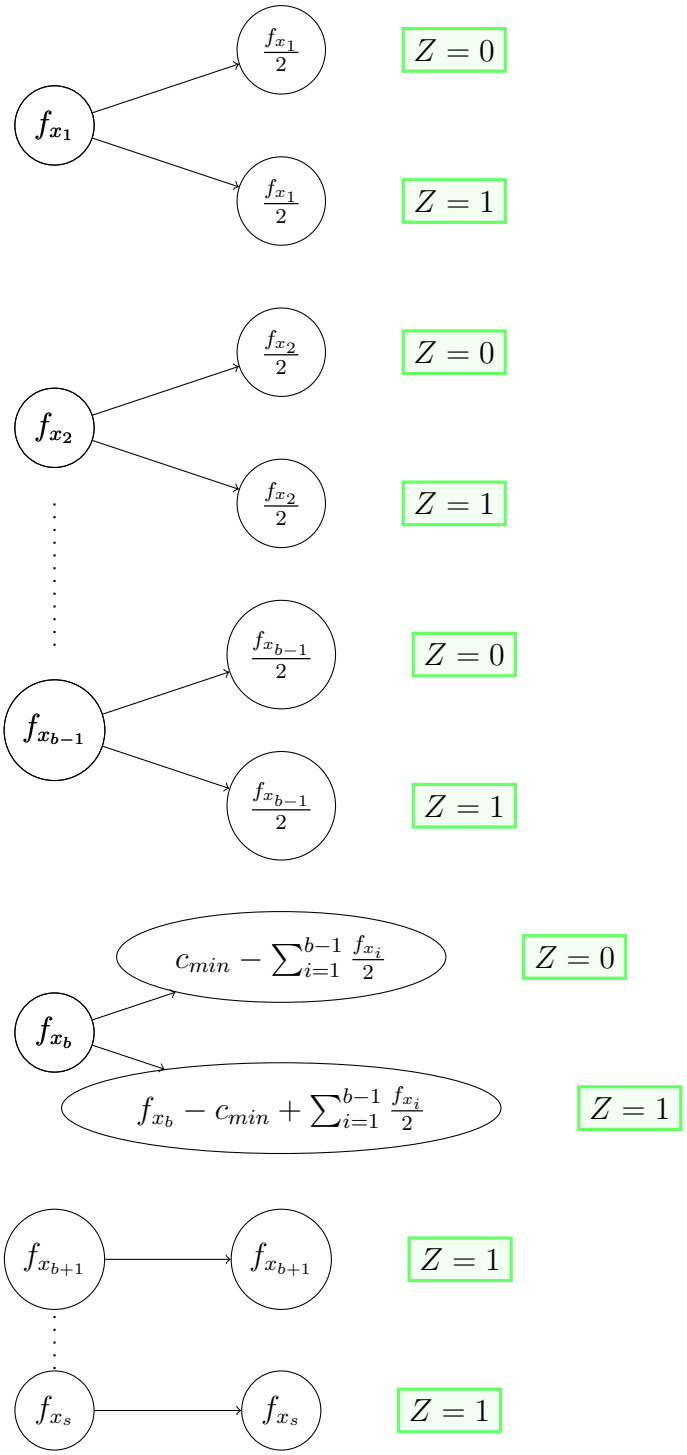


Figure 4.38: Arrangement for $Aac(X)_{max}$ when $c_{min} > \frac{f_{x_1}}{2}$

$$Aac(X, l, l+1, n)_{max} = \begin{cases} c_{min} \times \frac{f_{x_1}}{n-l} & \text{if } c_{min} \leq \frac{f_{x_1}}{2} \\ \frac{\sum_{i=1}^{b-1} f_{x_i}^2}{2(n-l)} + (c_{min} - \frac{\sum_{i=1}^{b-1} f_{x_i}}{2}) \times \frac{f_{x_b}}{n-l} & \text{otherwise} \end{cases} \quad (4.8)$$

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

Proof. From Lemma 5, 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 4.37 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 4.38 illustrates this scenario. Hence, $Aac(X, l, l+1, n)_{max} = \frac{\sum_{i=1}^{b-1} f_{x_i}^2}{2(n-l)} + (c_{min} - \frac{\sum_{i=1}^{b-1} f_{x_i}}{2}) \times \frac{f_{x_b}}{n-l}$. □

Example 3 and 4 illustrates the two cases for equation (4.8) for $n = 100$, $Aac(X, 0, 1, n)_{max} = Aac(X)_{max}$..

Example 3. *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,* $Aac_{max} = c_{min} \times \frac{f_{x_1}}{n} = 15 \times \frac{40}{100} = 6$.

Example 4. *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,* $Aac_{max} = \frac{\sum_{i=1}^{b-1} f_{x_i}^2}{2n} + (c_{min} - \frac{\sum_{i=1}^{b-1} f_{x_i}}{2}) \times \frac{f_{x_b}}{n} = \frac{40^2}{2 \times 100} + (30 - \frac{40}{2}) \times \frac{30}{100} = 11$

From Theorem 4 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 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 6. *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 [73] which completes the proof. □

A simple 1/2 -approximation algorithm [74] 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(4.9) for finding $Aac(X, l, l+1, n)_{min}$

$$Aac(X, l, l+1, n)_{min} = \begin{cases} (\sum_{i=1}^{a} f_{x_i} - c_{max}) \times \frac{f_{x_a}}{n-l} & \text{if } \sum_{i=1}^{a} f_{x_i} - c_{max} \leq \frac{f_{x_a}}{2} \\ (c_{max} - \sum_{i=1}^{a-1} f_{x_i}) \times \frac{f_{x_a}}{n-l} & \text{otherwise} \end{cases} \tag{4.9}$$

where, $f_{x_1} \geq f_{x_1} \geq ... \geq f_{x_a}$, 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 $AV - average - conflict$

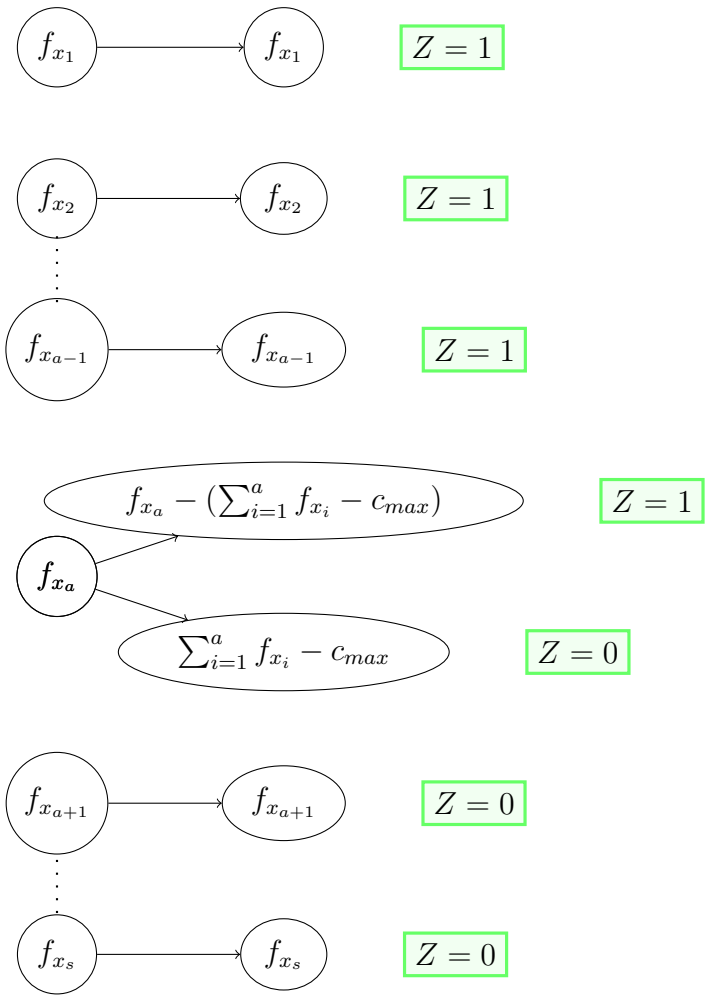


Figure 4.39: Arrangement for $Aac(X)_{min} > 0$ when $\sum_{i=1}^{a} f_{x_i} - c_{max} \leq \frac{f_{x_a}}{2}$

will be 0 for all such f_{x_i}. Figure 4.39 and 4.40 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 $Aac(X, l, l+1, n)$.

□

Example 5 and 6 illustrates the two cases of equation (4.9) for $n = 100$, $Aac(X, 0, 1, n)_{min} = Aac(X)_{min}$.

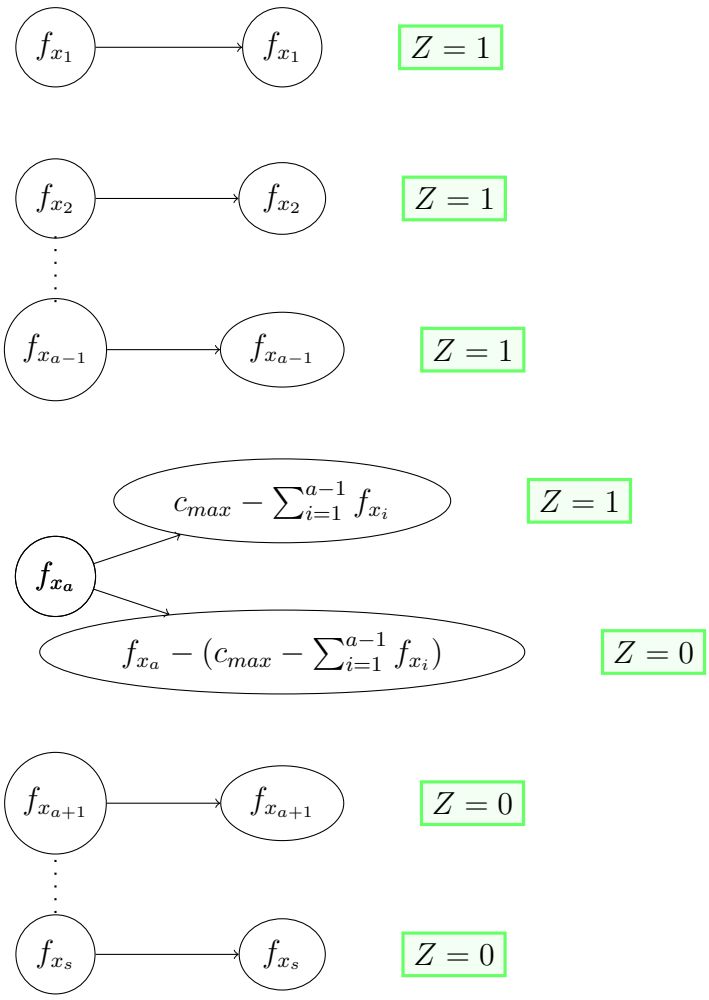


Figure 4.40: Arrangement for $Aac(X)_{min} > 0$ when $\sum_{i=1}^{a} f_{x_i} - c_{max} > \frac{f_{x_a}}{2}$

Example 5. *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 6. *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,*

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

Using maximum and minimum values of Aac from (4.8) and (4.9), we can update upper and lower bounds of Aac in (4.5) and (4.6) 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: $\sum_{i=1}^{a} f_{x_i} - c_{max} \leq \frac{f_{x_a}}{2}$, $c_{min} \leq \frac{f_{y_1}}{2}$

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

$$\Rightarrow n - l > \frac{f_{y_1} c_{min} - f_{x_a}(\sum_{i=1}^{a} f_{x_i} - c_{max})}{Aac(X, l, 1, l) - Aac(Y, l, 1, l)} \tag{4.10}$$

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

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

$$\Rightarrow n - l > \frac{\sum_{j=1}^{e-1} f_{y_j}^2 + (2c_{min} - \sum_{j=1}^{e-1} f_{y_j}) f_{y_e} - 2f_{x_a}(\sum_{i=1}^{a} f_{x_i} - c_{max})}{2[Aac(X, l, 1, l) - Aac(Y, l, 1, l)]} \tag{4.11}$$

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

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

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

$$\Rightarrow n - l > \frac{c_{min} f_{y_1} - (c_{max} - \sum_{i=1}^{a-1} f_{x_i}) p_a}{Aac(X,l,1,l) - Aac(Y,l,1,l)} \tag{4.12}$$

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

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

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

$$\Rightarrow n - l > \frac{\sum_{j=1}^{e-1} f_{y_j}^2 + (2c_{min} - \sum_{j=1}^{e-1} f_{y_j}) f_{y_e} - 2 f_{x_a} (c_{max} - \sum_{i=1}^{a-1} f_{x_i})}{2[Aac(X,l,1,l) - Aac(Y,l,1,l)]} \tag{4.13}$$

Using the upper and lower bounds, top-k attributes of a dataset can be derived by reading $l < n$ records. Algorithm 6 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 (4.9), 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 (4.10)-(4.13) .

Algorithm 6 Algorithm `top-`k`-Aac-UB-LB`

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** $\mathcal{A} \leftarrow \mathcal{A}/X_i$

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

4.5.2 Experimental Evaluation

Algorithm 6 has been implemented in Python 3 and ran on real datasets from Table 4.6. The result is summarized in Table 4.8. It is observed that although using a greedy approximation approach to find $Aac(X, l, l+1, n)_{min}$, our proposed solution derives top-k attributes with perfect precision for all the 5 different datasets. Here

Dataset	n	l	(%) *records processed*	*precision*
Census Income	299,285	193,109	**64.52**	1
Penbased-bin-3	10,992	10,360	94.25	1
Penbased	10,992	10,989	99.97	1
Connect-4	67,557	67,396	99.76	1
Kick Car	72,983	72,925	99.92	1

Table 4.8: Experimental evaluation of Algorithm 6

precision in computed with respect to the top-k attributes found using Algorithm 4. This shows the empirical evidence for the efficacy of proposed algorithm. 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. Census Income dataset has various domain sizes across different attributes, and most of the attribute domain size < 10. For the datasets Penbased-bin-3, Penbased, and Connect-4, the domain size is same for all attributes. Kick Car dataset has various domain sizes for different attributes, but most of the attributes have domain size ≥ 10. It is observed that larger number of records are processed for dataset with attributes having same domain size.

4.6 Related Works

Feature Selection Feature selection is an important topic for machine learning and data mining discipline. A plethora of work exists for various feature selection techniques. A review of feature selection methods with application can be found in [62, 63]. Various feature selection methods can be broadly categorized under - Filtering , Wrapper and Embedded & Hybrid methods. MI based feature selection is a filtering method where features are ranked according to MI score wrt the class

variable. [4] 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, but our aim is to devise a faster method without actually calculating MI . MI has been used for feature engineering tasks as well [5]. Recent works have proposed sampling based techniques for selecting top-k features using empirical MI [7, 6]. [6] focuses on improving speed of the calculation compared to [7],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 [8], 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 [75, 76, 69]. A notion of information dependency has been introduced in [9] which is used to explore AFD. [9] provides new definition for measuring AFD and shows how their measure compares with the one provided by the earlier work of [8]. Methods for quantifying approximation degree of AFD have been proposed in [10] based on prior work of [9]. A recent work proposed method for discovering dependencies using reliable MI [11]. 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.

4.7 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 dependencies* 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 proposed approach is able to use only 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. Our experimental evaluation illustrates the applicability of this efficient algorithm for real world datasets.

REFERENCES

[1] M. J. Smith, R. Wedge, and K. Veeramachaneni, "Featurehub: Towards collaborative data science," in *IEEE International Conference on Data Science and Advanced Analytics*. IEEE, 2017, pp. 590–600.

[2] J. Y. Zou, K. Chaudhuri, and A. T. Kalai, "Crowdsourcing feature discovery via adaptively chosen comparisons," *arXiv preprint arXiv:1504.00064*, 2015.

[3] J. Cheng and M. S. Bernstein, "Flock: Hybrid crowd-machine learning classifiers," in *ACM Conference on Computer Supported Cooperative Work & Social Computing*. ACM, 2015, pp. 600–611.

[4] J. R. Vergara and P. A. Estévez, "A review of feature selection methods based on mutual information," *Neural computing and applications*, vol. 24, no. 1, pp. 175–186, 2014.

[5] M. A. Salam, M. E. Koone, S. Thirumuruganathan, G. Das, and S. Basu Roy, "A human-in-the-loop attribute design framework for classification," in *The World Wide Web Conference*. ACM, 2019, pp. 1612–1622.

[6] X. Chen and S. Wang, "Efficient approximate algorithms for empirical entropy and mutual information," in *Proceedings of the 2021 International Conference on Management of Data*, 2021, pp. 274–286.

[7] C. Wang and B. Ding, "Fast approximation of empirical entropy via subsampling," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 658–667.

[8] J. Kivinen and H. Mannila, "Approximate inference of functional dependencies from relations," *Theoretical Computer Science*, vol. 149, no. 1, pp. 129–149, 1995.

[9] M. M. Dalkilic and E. L. Roberston, "Information dependencies," in *Proceedings of the nineteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, 2000, pp. 245–253.

[10] C. Giannella and E. Robertson, "On approximation measures for functional dependencies," *Information Systems*, vol. 29, no. 6, pp. 483–507, 2004.

[11] P. Mandros, M. Boley, and J. Vreeken, "Discovering dependencies with reliable mutual information," *Knowledge and Information Systems*, vol. 62, no. 11, pp. 4223–4253, 2020.

[12] J. Liu, J. Li, C. Liu, and Y. Chen, "Discover dependencies from data—a review," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 2, pp. 251–264, 2010.

[13] T. T. Lee, "An infornation-theoretic analysis of relational databases—part i: Data dependencies and information metric," *IEEE Transactions on Software Engineering*, no. 10, pp. 1049–1061, 1987.

[14] J. Heaton, "An empirical analysis of feature engineering for predictive modeling," in *SoutheastCon, 2016*. IEEE, 2016, pp. 1–6.

[15] M. R. Anderson and M. Cafarella, "Input selection for fast feature engineering," in *Data Engineering (ICDE), 2016 IEEE 32nd International Conference on*. IEEE, 2016, pp. 577–588.

[16] S. Basu Roy, A. Teredesai, K. Zolfaghar, R. Liu, D. Hazel, S. Newman, and A. Marinez, "Dynamic hierarchical classification for patient risk-of-readmission," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 1691–1700.

[17] J. M. Kanter and K. Veeramachaneni, "Deep feature synthesis: Towards automating data science endeavors," in *IEEE International Conference on Data Science and Advanced Analytics*. IEEE, 2015, pp. 1–10.

[18] G. Katz, E. C. R. Shin, and D. Song, "Explorekit: Automatic feature generation and selection," in *Data Mining (ICDM), 2016 IEEE 16th International Conference on*. IEEE, 2016, pp. 979–984.

[19] H. T. Lam, J.-M. Thiebaut, M. Sinn, B. Chen, T. Mai, and O. Alkan, "One button machine for automating feature engineering in relational databases," *arXiv preprint arXiv:1706.00327*, 2017.

[20] B. Frénay, G. Doquire, and M. Verleysen, "Is mutual information adequate for feature selection in regression?" *Neural Networks*, vol. 48, pp. 1–7, 2013.

[21] S. Brin, R. Motwani, and C. Silverstein, "Beyond market baskets: Generalizing association rules to correlations," in *Acm Sigmod Record*, vol. 26, no. 2. ACM, 1997, pp. 265–276.

[22] W. Li, "Mutual information functions versus correlation functions," *Journal of statistical physics*, vol. 60, no. 5-6, pp. 823–837, 1990.

[23] G. Yang, "The complexity of mining maximal frequent itemsets and maximal frequent patterns," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2004, pp. 344–353.

[24] R. Agarwal, R. Srikant *et al.*, "Fast algorithms for mining association rules," in *Proc. of the 20th VLDB Conference*, 1994, pp. 487–499.

[25] M. Madiman, "On the entropy of sums," in *Information Theory Workshop, 2008. ITW'08. IEEE*. IEEE, 2008, pp. 303–307.

[26] A. Feng, M. J. Franklin, D. Kossmann, T. Kraska, S. Madden, S. Ramesh, A. Wang, and R. Xin, "CrowdDB: Query processing with the VLDB crowd," *PVLDB*, vol. 4, no. 12.

[27] A. Marcus, E. Wu, D. Karger, S. Madden, and R. Miller, "Human-powered sorts and joins," *PVLDB.*, 2011.

[28] H. Park and J. Widom, "Query optimization over crowdsourced data," in *VLDB*, 2013.

[29] J. Wang, T. Kraska, M. J. Franklin, and J. Feng, "Crowder: Crowdsourcing entity resolution," *Proc. VLDB Endow.*, vol. 5, no. 11, pp. 1483–1494, 2012. [Online]. Available: http://vldb.org/pvldb/vol5/p1483_jiannanwang_vldb2012.pdf

[30] H. Kaplan, I. Lotosh, T. Milo, and S. Novgorodov, "Answering planning queries with the crowd," in *PVDLB*, 2013.

[31] J. Wang, G. Li, T. Kraska, M. J. Franklin, and J. Feng, "Leveraging transitive relations for crowdsourced joins," in *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2013, New York, NY, USA, June 22-27, 2013*, K. A. Ross, D. Srivastava, and D. Papadias, Eds. ACM, 2013, pp. 229–240. [Online]. Available: https://doi.org/10.1145/2463676.2465280

[32] A. Marcus, D. Karger, S. Madden, R. Miller, and S. Oh, "Counting with the crowd," in *PVLDB*, 2013.

[33] J. Wang, T. Kraska, M. J. Franklin, and J. Feng, "Crowder: Crowdsourcing entity resolution," *Proceedings of the VLDB Endowment*, vol. 5, no. 11, pp. 1483–1494, 2012.

[34] J. Wang, G. Li, T. Kraska, M. J. Franklin, and J. Feng, "Leveraging transitive relations for crowdsourced joins," in *ACM SIGMOD International Conference on Management of Data*. ACM, 2013, pp. 229–240.

[35] S. E. Whang, P. Lofgren, and H. Garcia-Molina, "Question selection for crowd entity resolution," *Proceedings of the VLDB Endowment*, vol. 6, no. 6, pp. 349–360, 2013.

[36] N. Vesdapunt, K. Bellare, and N. Dalvi, "Crowdsourcing algorithms for entity resolution," *Proceedings of the VLDB Endowment*, vol. 7, no. 12, pp. 1071–1082, 2014.

[37] S. Wang, X. Xiao, and C.-H. Lee, "Crowd-based deduplication: An adaptive approach," in *ACM SIGMOD International Conference on Management of Data*. ACM, 2015, pp. 1263–1277.

[38] X. Chen, P. N. Bennett, K. Collins-Thompson, and E. Horvitz, "Pairwise ranking aggregation in a crowdsourced setting," in *ACM International Conference on Web Search and Data Mining*. ACM, 2013, pp. 193–202.

[39] B. Eriksson, "Learning to top-k search using pairwise comparisons," in *Artificial Intelligence and Statistics*, 2013, pp. 265–273.

[40] S. Guo, A. Parameswaran, and H. Garcia-Molina, "So who won?: dynamic max discovery with the crowd," in *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*. ACM, 2012, pp. 385–396.

[41] A. R. Khan and H. Garcia-Molina, "Hybrid strategies for finding the max with the crowd: technical report," Stanford InfoLab, Tech. Rep., 2014.

[42] T. Pfeiffer, X. A. Gao, Y. Chen, A. Mao, and D. G. Rand, "Adaptive polling for information aggregation." in *AAAI*, 2012.

[43] P. Ye and D. Doermann, "Combining preference and absolute judgements in a crowd-sourced setting," in *International Conference on Machine Learning*, 2013, pp. 1–7.

[44] A. G. Parameswaran, H. Garcia-Molina, H. Park, N. Polyzotis, A. Ramesh, and J. Widom, "Crowdscreen: Algorithms for filtering data with humans," in *ACM SIGMOD International Conference on Management of Data*. ACM, 2012, pp. 361–372.

[45] A. Parameswaran, S. Boyd, H. Garcia-Molina, A. Gupta, N. Polyzotis, and J. Widom, "Optimal crowd-powered rating and filtering algorithms," *Proceedings of the VLDB Endowment*, vol. 7, no. 9, pp. 685–696, 2014.

[46] A. D. Sarma, A. Parameswaran, H. Garcia-Molina, and A. Halevy, "Crowd-powered find algorithms," in *IEEE International Conference on Data Engineering*. IEEE, 2014, pp. 964–975.

[47] T. Yan, V. Kumar, and D. Ganesan, "Crowdsearch: exploiting crowds for accurate real-time image search on mobile phones," in *International Conference on Mobile Systems, Applications and Services*. ACM, 2010, pp. 77–90.

[48] C.-J. Ho, S. Jabbari, and J. W. Vaughan, "Adaptive task assignment for crowdsourced classification," in *International Conference on Machine Learning*, 2013, pp. 534–542.

[49] J. Bragg, D. S. Weld *et al.*, "Crowdsourcing multi-label classification for taxonomy creation," in *AAAI Conference on Human Computation and Crowdsourcing*, 2013.

[50] M. Imran, C. Castillo, J. Lucas, P. Meier, and S. Vieweg, "Aidr: Artificial intelligence for disaster response," in *International Conference on World Wide Web*. ACM, 2014, pp. 159–162.

[51] C. Sun, N. Rampalli, F. Yang, and A. Doan, "Chimera: Large-scale classification using machine learning, rules, and crowdsourcing," *Proceedings of the VLDB Endowment*, vol. 7, no. 13, pp. 1529–1540, 2014.

[52] Y. Yan, R. Rosales, G. Fung, and J. G. Dy, "Active learning from crowds." in *International Conference on Machine Learning*, vol. 11, 2011, pp. 1161–1168.

[53] B. Mozafari, P. Sarkar, M. Franklin, M. Jordan, and S. Madden, "Scaling up crowd-sourcing to very large datasets: a case for active learning," *Proceedings of the VLDB Endowment*, vol. 8, no. 2, pp. 125–136, 2014.

[54] M. Fang, J. Yin, and D. Tao, "Active learning for crowdsourcing using knowledge transfer." in *AAAI*, 2014, pp. 1809–1815.

[55] J. Zhong, K. Tang, and Z.-H. Zhou, "Active learning from crowds with unsure option." in *IJCAI*, 2015, pp. 1061–1068.

[56] M. R. Anderson, D. Antenucci, V. Bittorf, M. Burgess, M. J. Cafarella, A. Kumar, F. Niu, Y. Park, C. Ré, and C. Zhang, "Brainwash: A data system for feature engineering." in *CIDR*, 2013.

[57] U. Khurana, D. Turaga, H. Samulowitz, and S. Parthasrathy, "Cognito: Automated feature engineering for supervised learning," in *IEEE International Conference on Data Mining Workshops*. IEEE, 2016, pp. 1304–1307.

[58] C. Zhang, A. Kumar, and C. Ré, "Materialization optimizations for feature selection workloads," *ACM Transactions on Database Systems*, vol. 41, no. 1, p. 2, 2016.

[59] F. Seide, G. Li, X. Chen, and D. Yu, "Feature engineering in context-dependent deep neural networks for conversational speech transcription," in *IEEE Workshop on Automatic Speech Recognition and Understanding*. IEEE, 2011, pp. 24–29.

[60] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[61] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *ACM International Conference on Multimedia*. ACM, 2014, pp. 675–678.

[62] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of machine learning research*, vol. 3, pp. 1157–1182, 2003.

[63] A. Jović, K. Brkić, and N. Bogunović, "A review of feature selection methods with applications," in *2015 38th international convention on information and communication technology, electronics and microelectronics (MIPRO)*. Ieee, 2015, pp. 1200–1205.

[64] N. Hoque, D. K. Bhattacharyya, and J. K. Kalita, "Mifs-nd: A mutual information-based feature selection method," *Expert Systems with Applications*, vol. 41, no. 14, pp. 6371–6385, 2014.

[65] C. Z. Mooney, *Monte carlo simulation*. Sage, 1997, no. 116.

[66] J. Neyman, "On the two different aspects of the representative method: the method of stratified sampling and the method of purposive selection," in *Breakthroughs in statistics*. Springer, 1992, pp. 123–150.

[67] T. M. Cover, *Elements of information theory*. John Wiley & Sons, 1999.

[68] R. Elmasri, *Fundamentals of database systems*. Pearson Education India, 2008.

[69] Y. Huhtala, J. Kärkkäinen, P. Porkka, and H. Toivonen, "Tane: An efficient algorithm for discovering functional and approximate dependencies," *The computer journal*, vol. 42, no. 2, pp. 100–111, 1999.

[70] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[71] J. Han, J. Pei, and M. Kamber, *Data mining: concepts and techniques*. Elsevier, 2011.

[72] R. Baeza-Yates, B. Ribeiro-Neto *et al.*, *Modern information retrieval*. ACM press New York, 1999, vol. 463.

[73] M. R. Garey and D. S. Johnson, "Computers and intractability: a guide to np-completeness," 1979.

[74] A. Caprara, H. Kellerer, and U. Pferschy, "The multiple subset sum problem," *SIAM Journal on Optimization*, vol. 11, no. 2, pp. 308–319, 2000.

[75] N. Novelli and R. Cicchetti, "Functional and embedded dependency inference: a data mining point of view," *Information Systems*, vol. 26, no. 7, pp. 477–506, 2001.

[76] S. Lopes, J.-M. Petit, and L. Lakhal, "Efficient discovery of functional dependencies and armstrong relations," in *International Conference on Extending Database Technology*. Springer, 2000, pp. 350–364.