Monotone Functions and Expert Models for Explanation of Machine Learning Models

Harlow Huber
Department of Computer Science
Central Washington University
Ellensburg, WA, USA
Harlow.Huber@cwu.edu

Boris Kovalerchuk
Department of Computer Science
Central Washington University
Ellensburg, WA, USA
Boris.Kovalerchuk@cwu.edu

***Abstract*—There are significant difficulties for the acceptance of black-box Machine Learning (ML) models by subject matter experts (SMEs) despite significant achievements of many black-box models. A promising way to address this problem is by building a trustable, qualitative, interpretable models for the task based on SME knowledge. Such qualitative models can work as qualitative explainers of black-box models or as sanity checks for them. In this paper, qualitative models operate with ordinal attributes, which can be Boolean or k-valued attributes with a small *k*. Humans easier understand and reason with such attributes that with continuous numeric attributes with many more values. Some ML tasks do not have sufficient training data. Building qualitative models with a SME ("expert models") is a way to solve these tasks solely with a SME's knowledge. The proposed Monotone Ordinal Expert Knowledge Acquisition (MOEKA) system allows building "expert models" through interview phases with the SME. Monotonicity is a key property of the system that is tested and used to shorten the interview process with a SME along with new methods for selecting the order of questions. MOEKA rather directly approximates domain knowledge in contrast with other ML explainers, which approximate black-box ML models. MOEKA can be also used as a method of database searching. Several case studies on gout, diabetes, and housing demonstrate efficiency of MOEKA.**

Keywords—Machine Learning, Data Mining, Monotone Boolean Functions, Ordinal Data, Monotone Ordinal Functions, Monotonicity, Subject Matter Expert (SME), Interview, Optimization, Knowledge Discovery, Visualization.

I. INTRODUCTION

Machine learning (ML) methods demonstrated significant achievements recently. However, there are significant difficulties for acceptance of ML models by the subject matter experts (SMEs). They may not trust machine learning (ML) models that are black boxes for the SME [1, 2, 3]. Even if good ML models are built for some domain, they will go unused if SMEs do not trust them.

Several approaches have been proposed in machine learning to resolve this issue [2, 4, 5]. A promising way to solve this problem is building a trustable qualitative interpretable model for the task based on SME knowledge and using it to help to understand, interpret, trust and/or improve the machine learning model built from data [5]. In contrast data-driven models based on the training data are not based on answers provided by SMEs. Another benefit is combining the expert mining approach with visualization using the Multiple Disk Form (MDF) [22].

This paper generalizes the Boolean expert mining approach detailed in [5, 21, 22] to the *ordinal* expert mining approach. This paper is organized as follows: Section II describes the background of the proposed method, Section III presents concepts and algorithms, Section IV defines the proposed process in detail, Section V presents new efficient methods for ordering questions, Section VI describes case studies, and Section VII summarizes the paper with the outline of the future work.

II. BACKGROUND

Below we provide the background knowledge and introduce the main concepts of the proposed the Monotone Ordinal Expert Knowledge Acquisition (MOEKA) system.

A. Expert Model Aproach

Qualitative ML models operate with attributes, which have few values (Boolean attributes three-valued attributes, and so on). In contrast, numeric ML models deal with numeric attributes with potentially an infinite number of values. Qualitative models are easier for humans to use for several reasons. It includes human memory limitations when operating with many numeric values and attributes and the ease of reasoning with a few qualities rather than detailed quantities [6, 7, 8]. For example, it is easier and simpler to reason with a qualitative term cold, instead of the precise temperature -15^0C.

Building a qualitative model for a task based on SME knowledge is a longstanding ML problem in the knowledge acquisition domain [29]. Typically, SMEs may not be able to build their own ML models, themselves not being data scientists. Therefore, a practical approach is interviewing the SME to build a model on some set of attributes. The number of these questions in the interview can grow exponentially with the number of attributes and the number of values in each attribute. The model that is directly built with an SME is called an "**expert model**" [5]. In this research, the expert model is a function that is created by interviewing the SME using MOEKA. The brute-force method is to have the SME manually classify each *n*-D point in the entire feature/attribute space. This technique of manually labelling and/or relabeling every *n*-D point is quite tedious and a SME should not be expected to spend such a large amount of time to do so. The optimized MOEKA needs significantly fewer SME classification actions than with the brute force approach to build the expert model. The MOEKA method generates a series of questions about the *n*-D points for the SME. In this context, a "question" refers to the classification of a single *n*-D

point/sample. However, by using the properties of *monotonicity*, the potential number of questions is significantly reduced, which make MOEKA an attractive addition to other ML methods. The MOEKA method generalizes the previously existing strategy of "Expert Data Mining" [5] by allowing the use of ordinal (multi-valued) attributes and target attributes, referred to as ***k*-valued attributes**, as well as the new concept of efficient ordering of questions to decrease the number of questions.

MOEKA can also be used for sanity check for ML models that use monotone data. It involves inspecting each difference on a point-by-point basis between the data-driven ML model and the expert model built with SME knowledge. True and false positives and negatives can be collected and analyzed to see any differences in precision, recall, etc.

Another benefit is with lossless visualization of the expert model [22]. The output of the MOEKA process is a function that can be represented in domain specific language as well as visually. The visual representation in [22] is lossless for Boolean data and encompasses the entirety of the feature space for the ML task with clear separations between classes. This aids the SMEs in acceptance of the expert model and can potentially help the explainability of ML models. The focus of this paper is on discovering expert models with a broader set of techniques than in past work [5, 21, 22].

Furthermore, the MOEKA method differs greatly from existing ML strategies. Commonly, ML classifier methods use a real dataset to train the model on, whereas the MOEKA method bypasses the need for a real dataset in favor of a SME. The resulting classifier is directly derived from the SME, leading to its high interpretability and explainability for the SME being presented in the terms of the domain language. Most other monotone classifiers reviewed below use monotonicity as a guarantee that the model will match monotonicity restraints in the task, either through some modifications in the data preprocessing step, and/or the training step of building a model.

B. Literature Review

Below we review the current literature on monotone functions in ML in comparison with the characteristics of MOEKA. Many current approaches fundamentally are data driven and need explicit training data. In contrast, the MOEKA method seeks to reduce the complexity in training classifiers by bypassing the need for explicit training data in favor of using SMEs as implicit "training data." With MOEKA, the expert knowledge becomes the ML model.

Some methods use monotonicity to enhance the learning process by injecting SME knowledge into the model built on the data. For example, one study uses monotonicity to classify cancer patients based on a set of precedents [9]. The precedents are classified *n*-D points for indolent cancer or aggressive cancer. There are also defined degrees for cancer indicators (attributes/features) to be indolent cancer or aggressive cancer. When patient indicators are to a lower or equal degree of indolent cancer, then that patient has indolent cancer.

When patient indicators for cancer are to a higher or equal degree of aggressive cancer, then that patient also has aggressive cancer. This is the monotonicity property for indolent and aggressive cancer is coming from the SME. This monotonicity is used to expand the classifications of precedents to the yet unclassified new *n*-D points on the *prediction stage*. Instead, MOEKA uses monotonicity to *reduce the time* and *to mitigate the lack of data* in the *training stage*. With MOEKA, the data monotonicity is verified with SME and is assumed after SME confirmation of data monotonicity.

Furthermore, monotone decision trees [12, 13] offer the benefit of being able to use continuous attributes as well as ordinal attributes. In [12, 13], algorithms for monotone decision trees are capable of outputting functions for multiple ordinal classes, which are classes that are able to be ranked by some order. They also can use continuous attributes converted to interval-valued attributes. MOEKA focuses on *ordinal attributes*. It is possible to take continuous or real data and transform them into interval-valued data and treat them as ordinal attributes, but the specifics of those transformations are not internal to the MOEKA method. MOEKA and monotone decision trees are similar in terms of form of the output function.

Next, MOEKA differs from popular computational ML explanation methods like Shapley Additive Explanations (SHAP) and Local Interpretable Model-agnostic Explanations (LIME). SHAP attempts to explain ML models by giving each attribute of the model a value that represents its *importance* to the ML model. LIME computes a *local linear model* for the ML model's individual predictions as an "explanation" for that individual prediction.

Thus, fundamentally both SHAP and LIME do not produce a *new interpretable model* by interviewing the domain expert but attempt to explain the *existing uninterpretable black-box* ML model. The expert model built by interviewing an SME is understandable by the SME due to its design. In contrast, explanations derived from the black box models may or may not be understandable by the SME and can be inconsistent with SME domain knowledge.

Specifically, some known issues with LIME and SHAP are that they can be "tricked to provide any desired explanation by modifying the model in data support" [27]. They are only *quasi-explanations* that can mislead users about the importance of the attributes [28]. The explanations must be accepted by SMEs in order to be considered true explanations of the model. There are also no internal methods of verifications for the local approximations output by SHAP and LIME, so it is hard to trust these explanations [28].

In contrast, MOEKA does not attempt to build an explainable model based on some black-box model. The MOEKA system builds a completely separate "expert model" directly with the SME. The expert model is an ordinal or Boolean function approximation of the SME knowledge instead of a linear approximation of the ML model like LIME

When the expert model is used as an explainable method for the ML model, we are comparing the extracted SME domain knowledge to the ML model. LIME and SHAP attempt to explain ML models without using external domain knowledge but only relying on the black-box model itself.

Lastly, there are some other methods with research into non-monotone noise [14-18], that can corrupt data monotonicity. This is an important research area for enhancing MOEKA in the future, but it is beyond the current study, though we conducted some preliminary studies in this area.

III. Concepts and Algorithms

In this section, the definitions and algorithms of the MOEKA method are defined.

A. Monotonicity and Monotone Functions

Monotonicity defines a pattern that a function of n variables can have. This pattern is such that the function is always decreasing or increasing provided that the n-D input is decreasing or increasing in all variables. In the context of MOEKA, we only consider the **increasing monotone functions** $f(\mathbf{x})$:

$$\text{If } \mathbf{x} \geq \mathbf{y} \text{ then } f(\mathbf{x}) \geq f(\mathbf{y}),$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and $\mathbf{y} = (y_1, y_2, \ldots, y_n)$ are n-D points, and

$$\mathbf{x} \geq \mathbf{y} \Leftrightarrow x_1 \geq y_1 \,\&\, x_2 \geq y_2 \ldots \&\, x_n \geq y_n.$$

Monotone Boolean Function (**MBF**) are simply monotone functions of n Boolean variables $x_1, x_2, \ldots, x_n$ with Boolean outputs.

***k*-valued Monotone Functions**, or **Monotone Ordinal Functions** (**MOF**) are monotone functions that have k number of ordinal outputs. Therefore, the outputs have some natural ordering, such as small, medium, large, or 0, 1, and 2 for $k = 3$.

k-valued ordinal attributes are attributes which have k ordinal values. MBFs have only Boolean attributes ($k = 2$), but with MOF we also consider attributes with $k = 2$. More general MOFs allow SMEs to be more specific than Boolean.

Monotonicity and MBF are known concepts that are external to MOEKA, but MOF is a unique term and concept to MOEKA that was created to generalize the previous work [5, 21] to ordinal data with $k > 2$.

B. Algorithms for Hansel Chains

Hansel introduced the chains of Boolean n-D points in 1966 for MBFs, which are now known as Hansel Chains [21]. Below we define the Hansel Chains generalized from MDF to MOF. Each **Hansel Chain** (HC) is a set of ordered n-D points, e.g., $\mathbf{x}_1 < \mathbf{x}_2 < \mathbf{x}_3 < \mathbf{x}_4$, which are used to store the n-D points within the space of the MBF/MOF. The **low unit** is the n-D point of MBF that marks the border between two classes within a HC [5], e.g., when $f(\mathbf{x}_1) = f(\mathbf{x}_2) = 0$, but $f(\mathbf{x}_3) = f(\mathbf{x}_4) = 1$ we have $\mathbf{x}_3$ as a low unit. Some HC may not have a low unit if all n-D points of the chain have $f(\mathbf{x}) = 0$. HC cover the entirety of the n-D space (all possible n-D points for the MBF/MOF), and each HC is unique with no duplicate n-D point in different HCs. We can have *several "low units"* on a single HC for MOF, which we will call **low *k*-values**. For example, $f(\mathbf{x}_1) = 0$ $f(\mathbf{x}_2) = 1$, $f(\mathbf{x}_3) = 1$, but $f(\mathbf{x}_4) = 2$. Here $\mathbf{x}_2$ is low 1-value and $\mathbf{x}_4$ is low 2-value.

Below we outline the **algorithm to construct all HCs for MBF** of n variables, following [5] and [21]. The starting chain for constructing HCs of all dimensions is a single HC of dimension 1: $\mathbf{S}_1 = \{ (0), (1) \}$.

Step 1. *Append* $\mathbf{S}_1$ to make two 2-D chains for MBF:

$$\mathbf{S}_{2.1} = \{(0, 0), (0, 1)\}, \mathbf{S}_{2.2} = \{(1, 0), (1, 1)\}$$

Step 2. *Move* the largest n-D point from one isomorphic set to another isomorphic set for all isomorphic sets. Sets are isomorphic when they have the same number of n-D points. Thus, (1, 1) is removed from $\mathbf{S}_{2.2}$ to the end of $\mathbf{S}_{2.1}$

$$\mathbf{S}_{2.1} = \{ (0, 0), (0, 1), (1, 1) \}, \mathbf{S}_{2.2} = \{ (1, 0) \}.$$

After this step has been performed, the two sets are considered Hansel Chains. This process is iterative for n dimensions. These algorithms are detailed in [5, 21, 22]. The new algorithm for MOF is a generalization of the first step by appending up to k values instead of only the Boolean values of zero and one. Example, $k = 3$, dimension 1: $\mathbf{S}_1 = \{ (0), (1) (2)\}$.

MOF Step 1. *Append* $\mathbf{S}_1$ to make two 2-D chains for MOF:

$$\mathbf{S}_{2.1} = \{(0, 0), (0, 1), (0,2)\}, \mathbf{S}_{2.2} = \{(1, 0), (1, 1),(1.2)\}$$

$$\mathbf{S}_{2.3} = \{(2, 0), (2, 1),(2.2)\}$$

MOF Step 2. Move (1,2) to $\mathbf{S}_{2.1}$ and (2,2) to $\mathbf{S}_{2.2}$

$$\mathbf{S}_{2.1} = \{ (0, 0), (0, 1), (0, 2), (1,2) \},$$

$$\mathbf{S}_{2.2} = \{ (1, 0), (1, 1),(2,2) \}, \mathbf{S}_{2.3} = \{(2, 0), (2, 1)\}$$

C. Algorithms to restore MBFs and MOFs

Algorithms to restore MBFs and MOFs traverse HCs in multiple ways, expanding the process from [5, 21, 22] for MBF to MOF. Consider an n-D Boolean space with the generic HC:

$$\mathbf{S} = \{ a, b, c, d, e \}.$$

where $a < b < c < d < e$. If the SME states that $f(a) = 1$, then it can be assumed that $f(b) = 1$. Since $f(b) = 1$, then $f(c) = 1$ and so on to $f(e)=1$, since $f(a) = 1$ is known by the SME. Conversely, if it is stated that that $f(\mathrm{e}) = 0$, then $f(a) = f(b) = f(c) = f(d) = 0$. The n-D points with classes that were not specified by the SME are "*expanded*" through monotonicity instead of being classified by the SME. The n-D point that was classified by the SME is called the "*source of expansion*". When the expanded point is greater than its source of expansion, then that expansion is called an "*up-expansion*." When the expanded n-D point is lesser than its source of expansion, then that expansion is called a "*down-expansion*."

k_{n+1} refers to the number of values of the MOF (number of classes/labels). For Boolean functions ($k_{n+1} = 2$), if an n-D point has a class of 0 or 1, then it can expand another n-D point in either the upwards or downwards direction. However, for $k_{n+1} > 2$, it is possible to expand in both directions. For example, consider these Boolean n-D points:

$$a = (1, 0, 1), b = (1, 1, 1), c = (1, 0, 0).$$

If $k_{n+1} = 3$ and $f(a) = 1$, then b and c can be expanded, like $f(b) = 2$ and $f(c) = 0$, with $f(c) \leq f(a) \leq f(b)$. In this example, $f(b)$ is the up-expansion and $f(c)$ is the down-expansion. One caveat with up and down-expansions is that the exact class for $f(c)$ and $f(b)$ is unknown when $f(a) = 1$. Knowing $f(a) = 1$ we can only get $f(c) \leq 1$ and $f(b) \geq 1$. The SME or user still needs to confirm if $f(c) = 0$ or 1 and if $f(b) = 1$ or 2. Therefore, the middle values for any value of k_{n+1} are referred as *"weak" values*. If a user assigns a class to a n-D point, and that class is the smallest or greatest possible value with respect to k_{n+1}, then the expansions from that n-D point will be *"strong" values*. If $f(a) = 2$, then $f(b) = 2$ and $f(c) \leq 2$. $f(b) = 2$ is a strong value but $f(c) \leq 2$ is a weak value that will need to be confirmed. Strong answers using the lowest and highest possible class will reduce the number of questions that need to be asked to the user.

The system traverses all HCs to find the low units. Since there can be multiple low units on each chain due to the possibility of multiple ordinal class values as shown above, it is necessary to define our MOF for the multiple values of k_{n+1}. For instance, we may have $f(\mathbf{x}) = 1$ if $x_3 = 1$ in $\mathbf{x}$, and $f(\mathbf{x}) = 2$ if $x_5 =$

1 in **x**. We do not need to define $f(\mathbf{x}) = 0$ explicitly, but we can exploit the absence of $f(\mathbf{x}) = 1$ and $f(\mathbf{x}) = 2$ to conclude that $f(\mathbf{x}) = 0$ if **x** contains $!x_3!x_5$, e.g., for $n = 5$ this **x** is (1,1,0,1,0).

Moreover, since the *n*-D points may have ordinal attributes, they need to be defined in the function as well. Let **k** = (3, 3, 3) for a dimension of 3, and $\mathbf{x}_3 = (1, 1, 0)$ and $\mathbf{x}_5 = (1, 2, 2)$. Typically, an MBF in the DNF form will be something like $f(\mathbf{x}) = x_1x_2 \vee x_2x_3$. This does not work for MOF because it is unclear what values for each x_i are needed for $f(\mathbf{x}) = 1$ and $f(\mathbf{x}) = 2$. These ordinal attribute values must be directly input into the DNF form of the function. Using the previous *n*-D points of $\mathbf{x}_3$ and $\mathbf{x}_5$, that would be: $f(\mathbf{x}) = 1 \Leftrightarrow (x_1 \geq 1)(x_2 \geq 1)$ and $f(\mathbf{x}) = 2 \Leftrightarrow (x_1 \geq 1)(x_2 = 2)(x_3 = 2)$.

Furthermore, dual expansions are a type of monotonic expansion that works with MOF only with more than one expansion source. If there are two *n*-D points within an HC that have the same value, and the *n*-D points between those border points are unclassified, then they will have the same class to preserve monotonicity.

IV. MOEKA Process

The MOEKA system is functional for the features that are described in this section. The following steps present the **MOEKA system process for MBF/MOF**.

Step 1. SME answers some "pilot questions" before classification can start. This user identifies attributes, the number of them, and the *k*-values of each attribute, which is the number of ordinal values an attribute can have ($k = 2$ for Boolean attributes) and confirm monotonicity for these attributes.

Step 2. System divides all possible *n*-D points of the *n*-D space into a set of HCs that fully cover this space.

Step 3. System forms a preliminary order of chains and *n*-D points within chains to be asked SME to answer.

Step 4: System-SME interaction to produce MBF/MOF.

Step 4.3. System analyzes currently collected answers by SME and can adjust preliminary order of chains and *n*-D points to be asked from SME.

Step 4.4. System query SME in accordance with order of chains and *n*-D points from step 4.3.

Step 4.5. Repeat steps 4.3 and 4.4 until all chains and questions are answered.

Step 5. System analyzes all collected data and produces an MBF/MOF.

V. Selection of the Orders of Questions

This section presents alternative orders of questions. The number of questions for the SME heavily depends on the order of questions asked. Examples in this paper illustrate it. The MOEKA system allows the selection of different orders to fit better the properties of individual functions, which helps in some situations (shown in case studies). In the previous work [5, 21, 22], the order of questions was fixed starting at the shortest chain and going to the longest chain without changing the order. The Hansel lemma guarantees that with chains, we can fully restore the MBF. The number of questions in this optimized process cannot be less than the sum of the number of upper zeroes and low units of the MBF. This is a low bound of the number of questions for the SME [5, 21]. The upper bound is 2^n for *n*-D Boolean space.

Thus, there is a plenty of room to decrease the number of questions within these bounds. Changing the order of questions is a promising way for such improvement. The key idea is getting tips from the SME on where in the chains low units can be located.

A. Alternative Orders of Questions

We group alternative orders of question in three categories:
(1) *intra-chain variations*, where orders of questions asked within a chain vary, but the order of chains is fixed,
(2) *inter-chain variations*, where the orders of chains vary, but the order of cases within the chains is fixed,
(3) *intra- and inter-chain variations,* where both orders of questions and HCs vary.

MOEKA supports these categories of orders. Table 1 lists the orders of each category (1)-(3), which are defined later.

TABLE 1: TYPES OF ORDERS.

Intra-chain variations	Inter-chain variations	Intra- and inter-chain variations
Top First	SHCF (Shortest HC First)	Chain Jump
Bottom First	LHCF (Longest HC First)	Hierarchical Model
Binary Search	Default Order	Majority
True Attributes	Minimal Spanning Tree Order	
	Shortest Path	
	Linkage Sort	

1) Intra-chain Orders

Bottom First*:* starting questioning from the smallest *n*-D point of the HC and going to the largest *n*-D point.

Top First*:* starting questioning from the largest *n*-D point of the HC and traversing sequentially to the smallest *n*-D point.

Binary Search*:* the assumption is that with binary search, the low unit can be found faster than sequentially iterating through a chain through the **Bottom First** or **Top First** approach. It can be confirmed by SME in the pilot questions. Also, binary search can be used when we do not have any tips from the SME. In this case, the motivation for the binary search is a high frequency of MBFs where the half of all attributes is present in each clause in the whole set of all MBFs.

The binary search process on chains is described below. For each HC, the algorithm first finds the smallest and largest unclassified *n*-D points on the chain. Then, the SME provides the value $f(\mathbf{x})$ for the middle case **x** between these cases. If $f(\mathbf{x}) = 0$ then the greater adjacent point **y** on the chain is asked from the SME. If $f(\mathbf{y}) = 1$ then **y** is a low unit. If $f(\mathbf{x}) = 1$, then the next smaller case **z** on the chain is asked from SME. If $f(\mathbf{z}) = 0$, then **x** is a low unit. If neither **x** nor **y** is a low unit, then a new middle point of the respective half of the chain is computed, and the process is continued in the same binary search way.

For MOF this process in more complex, because each chain can require searching for multiple *k*-value units, requiring expansions in both directions. For instance, for $k_{n+1} = 4$ and $f(\mathbf{x}) = 2$, we may have *n*-D points **w** and **u** on the chain such that $f(\mathbf{w}) = 1$ for $\mathbf{w} < \mathbf{x}$ and $f(\mathbf{u}) = 3$ for $\mathbf{u} > \mathbf{x}$. Here **x**, **w** and **u** can be 2-value, 1-value and 3-value low units, respectively. Let's consider a chain that consists of elements **x**, **w** and **u** and another point **z** such that $f(\mathbf{z}) = 0$: { **z**, **w**, **x**, **u** }. Here **x**, **w** and **u**

are k-value units (the border element between the classes of 0, 1, 2, and 3), which needs to be identified by asking the SME.

True Attributes*:* Here the order of the questions is dictated by "true attributes". These are important attributes identified by SME that should be always equal to 1 or greater to satisfy the value of the target function to be 1 or greater. It means that in every chain only cases with these attributes equal to 1 or greater are asked. All other cases are not skipped. This process does not order allowed cases and is augmented by other orders. Example. If the attribute x_1 always needs to be 1 for the SME to be satisfied with $f(\mathbf{x}) = 1$, then all questions with $x_1 = 0$ are skipped. For MOF, the value of k_{n+1} needs to be specified. The true attributes process plays a role like a feature reduction process in ML. The difference is that the feature is not "reduced," rather, it becomes a known value due to SME knowledge. Labelling an attribute as *true* means the number of potential questions is reduced to 2^{n-1} from 2^n for MBF.

2) Inter-chain Orders

Shortest HC First (**SHCF**): start with the smallest chain in ascending order of chain length.

Longest HC First (LHCF): start with the largest chain in descending order of chain length.

Default Order: the order of chain creation in the HC creation algorithm.

Minimal Spanning Tree Order (MSTO)*:* create a Minimal Spanning Tree (MST) from a complete graph with chains as nodes, then base the order on that MST (more on this is in the case study section).

Shortest Path Order (SPO): create a shortest path from a complete graph (more on this is in the case study section).

Linkage Sort Order (LSO): sort the HCs based on the linkages ("distances") between chains (more on this is in the case study section).

3) Inter- and Intra-chain Orders

Chain Jumping (CJ)*:* chain jumping refers to jumping from a HC A to another HC B if the answer for the starting questions on the chain A does not expand to the rest of the chain A. The assumption is that by going to the next chain B, the previous chain A may have their n-D points expanded from answers obtained on chain B. This process allows flexibility to jump back to chain A depending on answers on chain B and can be combined with other orders. Jumps can be considered as a part of an initial exploration of chains before regular questioning of n-D points on each chain will start.

Majority*:* the majority technique is to ask questions about n-D points of a specific number of "*positive" values first*. The idea is that the SME may believe that these points are more important, so those questions will be asked first. For example, let's take a non-binary 3-D point, where

$$\mathbf{k} = (k_1, k_2, k_3, k_4) = (3, 5, 2, k_4).$$

Here the smallest n-D point is (0, 0, 0) and the largest is (2, 4, 1). If the SME states that most likely two values need to be positive to have $f(\mathbf{x}) = 1$, then n-D points $\mathbf{x}$ = (2, 4, 0) and $\mathbf{y}$ = (1, 1, 0) would be acceptable. If range between points $\mathbf{x}$ and $\mathbf{y}$ is large for some attribute starting with the point with middlemost value of k_i for that attribute can shorten the number of questions to SME. Furthermore, if this technique is used, then the n-D points which are majority-flagged will be answered first, and then the rest of the questions will be asked with the given order.

Hierarchical Modeling*:* in some situations, due to a larger number of attributes, the number of questions becomes too large to feasibly conduct an interview with the SME. In these cases, a hierarchical grouping of some of the attributes can greatly reduce the number of questions with a potential loss in accuracy, depending on the MBF/MOF in question.

For example, for 15 Boolean attributes the brute force method takes 32,768 (2^{15}) questions. If these attributes are able to be split into 3 groups of 5, then there are 32 (2^5) potential questions for each group and 8 (2^3) questions between each group for a total of 104 ($32 \times 3 + 8$) questions. This may simplify or oversimplify the relationship between attributes. For example, let's say there is some clause in the MBF that is 12 attributes long and another that is 4 attributes long: $f(x) = x_1x_5x_7x_{10}x_{11}x_{12}x_{13}x_{14} \vee x_1x_6x_7x_{13}$. It is not guaranteed that those clauses can be preserved with a hierarchical model of 3 groups of 5 attributes. Let's define the groups as $y_1 = \{x_1,\ldots,x_5\}$, $y_2 = \{x_6,\ldots,x_{10}\}$, and $y_3 = \{x_{11},\ldots,x_{15}\}$. The clauses will be able to be preserved if the user defines $f(y) = y_1y_2y_3$, $f(y_1) = x_1x_5 \vee x_1 = x_1$, $f(y_2) = x_7x_{10} \vee x_6x_7$, and $f(y_3) = x_{11}x_{12}x_{13}x_{14} \vee x_{13}$. However, this also introduces unintended clauses in the MBF: $f(x) = x_1x_5x_7x_{10}x_{13} \vee x_1x_7x_{10}x_{13}$. Therefore, it is important to pick groups that do not overlap with other groups. Furthermore, the dependencies between attributes for grouping can be identified either by the SME or by computational analysis. In some cases, only expert information is available. Therefore, we focus on the interactive process where the SME makes this grouping.

B. Selection of the Order

Here we present some heuristics which aid in the selection of order of questions to present to the SME.

1) Heuristic 1

Select SHCF when we expect that $f(\mathbf{x}) = 1$ when expected low units $\mathbf{x}$ contain roughly the same number of positive values (1 or greater) and values of 0. It is established experimentally that often shorter HCs have relatively more such cases than longer chains.

2) Heuristic 2

Select LHCF when we expect that $f(\mathbf{x}) = 1$ when expected low units $\mathbf{x}$ contains either a relatively large or small number of positive values compare to values of 0. This is because longer HCs have a larger range of Hamming norms. The longest HC contains the datapoint with all 0s and the datapoint with all 1s.

3) Heuristic 3

Start at the bottom of each chain when we expect that $f(\mathbf{x}) = 1$ when $\mathbf{x}$ contains a relatively small number of positive values compare to values of 0. This can be paired with SHCF for a lower number of questions.

4) Heuristic 4

Start at the top of each chain when we expect that $f(\mathbf{x}) = 1$ when x contains a relatively large number of positive values compare to values of 0. This can be paired with LHCF for a lower number of questions.

5) Heuristic 5

The Default Order can be used if none of the heuristics is confirmed by the SME.

Some heuristics for the graph-based orders will be presented in the case study section.

VI. Cases Studies

The process of a case study is as follows: (1) recruit some SME (2) restore some MBF or MOF if possible.

A. Gout

This case study is on gout, a type of inflammatory arthritis where small urate crystals form around the joints [19]. The SME that was interviewed is the lead hematology lab technician for their hospital. The attributes were grouped without an overlap between groups. The three groups—attributes of synovial fluid, attributes of blood, and patient symptoms—are intended to be the clinical indicators of gout and are also treated as *super-attributes*. Without grouping these attributes, the search space is 6,400 potential questions to restore the MOF of $k_{n+1} = 5$. The target function f is to aid in the diagnosis of gout g. The classes $g = 1$, $g = 2$, $g = 3$, and $g = 4$ represent gout in increasing severity level, and the class of $g = 0$ is the absence of gout. The hierarchical model structure is as follows:

$$f(\mathbf{x}) = f(x_1,x_2,x_3) = g,$$

$$x_1 = h_1(y_1,y_2,y_3,y_4),\ x_2 = h_2(w_1,w_2,w_3),\ x_3 = h_3(z_1,z_2,z_3).$$

The attributes and k-values are listed in Table 2.

TABLE 2. ATTRIBUTES AND K-VALUES OF FUNCTIONS.

x_1: function h_1 (synovial fluid), $k = 5$	**x_2: function h_2 (blood), $k = 2$**	**x_3: function h_3 (patient symptoms), $k = 5$**
(y_1) Monosodium urate crystals, $k = 5$ (y_2) Calcium pyrophosphate crystals, $k = 5$ (y_3) Intracellular crystals, $k = 2$ (y_4) Infection, $k = 2$	(w_1) High white blood cell count, $k = 2$ (w_2) High red blood cell count, $k = 2$ (w_3) High uric acid, $k = 2$	(z_1) Gout attack, $k = 2$ (z_2) Interval gout attack, $k = 2$ (z_3) Swollen, painful, stiff joints, $k = 2$

MOFs were restored with a total of 93 questions, 1.45% of the total n-D points without grouping. This case study was done without any order heuristics except that y_1 must be equal to 1 or higher for $f(x_1)$ to be 1 or higher, and SHCF was used. The model is given below in Tables 3 through 5. In clauses like $(x_1 \geq 1)(x_3 \geq 2)$ the operator & is omitted, $(x_1 \geq 1)\&(x_3 \geq 2)$.

TABLE 3. GOUT MOF OUTER FUNCTIONS.

Value of g	**MOF $f(\mathbf{x})$**
1	$(x_1 \geq 1)$
2	$(x_1 \geq 2) \vee (x_1 \geq 1)(x_3 \geq 2)$
3	$(x_1 \geq 3) \vee (x_1 \geq 1)(x_3 \geq 3)$
4	$(x_1 \geq 4) \vee (x_1 \geq 2)(x_3 \geq 4)$

TABLE 4. GOUT MOF INNER FUNCTIONS FOR X_1 (SYNOVIAL FLUID).

Value of x_1	**MOF $h_1(x_1)$**
1	$(y_1 \geq 1)$
2	$(y_1 \geq 1)(y_2 \geq 3)(y_3) \vee (y_1 \geq 2)(y_2 \geq 2)(y_3) \vee (y_1 \geq 2)(y_2 \geq 3) \vee (y \geq 1)(y_4)$
3	$(y_1 \geq 2)(y_2 \geq 2)(y_4) \vee (y_1 \geq 2)(y_3)(y_4) \vee (y_1 \geq 3)(y_2 \geq 1) \vee (y_1 \geq 3)(y_3)$
4	$(y_1 \geq 3)(y_2 \geq 3)(y_3) \vee (y_1 \geq 3)(y_2 \geq 1)(y_4) \vee (y_1 \geq 4)$

TABLE 5. GOUT MOF INNER FUNCTIONS FOR X_3 (PATIENT SYMPTOMS).

Value of x_3	**MOF $h_3(x_3)$**
1	N/A
2	z_1
3	z_2
4	N/A

The resultant functions seem similar to others that exist in the literature [24] in the fact that the attributes directly related to clinical testing (x_1, attributes of synovial fluid) were most important, but the presence of patient symptoms could sometimes override the clinical test results, and the patient would be diagnosed with a harsher level gout. This can be seen in the fact that attribute x_1 and its sub-attribute y_1 are universally present in every clause of the function, but when there are more attributes in the clause, the value for x_1 and y_1 are lower. Also, note that the x_2, attributes of blood, were not present at all in the resultant function even though those attributes can "predict" gout. This is an example of the difference between replicating expert knowledge and building a classification/prediction model. In this study we focused on replicating a medical expert's knowledge rather than predict.

Furthermore, Figure 1 below is an MDF visualization of inner function $h_1(x_1)$. The only difference between this visualization method and the visualizations in [22] is the addition of multiple ordinal classes instead of only Boolean classes. There is a clear separation of the 4 values of x_1. MDF can be used for all MBF and MOF.

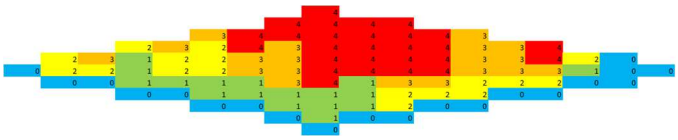

FIGURE 1. MDF VISUALIZATION OF GOUT MOF INNER FUNCTIONS FOR X_1.

Due to difficulties in acquiring real data from the hospital, this model was not tested on real data. However, the hematologist was confident that the model matched clinical practice.

B. Diabetes

The second case study is on type 2 diabetes. The SME is the same hematology lab technician. The target function f is to aid in the diagnosis of diabetes d. There are only two classes (initially): $d = 0$ for the absence of diabetes and $d = 1$ for diabetes. In this study, the attributes were not chosen through the SME but rather agreed with a Kaggle dataset [25]. The "gender" attribute was dropped because the SME said it did not fit a diagnostic function.

The dataset uses real numbers for some attributes, so they were mapped to ordinal values based on quartiles ($k = 4$), which the SME agreed to. The "smoking history" attribute has 4 original values: never smoked, former smoker, current smoker, and no info/unknown, but this was generalized to never smoked/no info/unknown, former smoker, and current smoker for $k = 3$. The search space is 3,072 potential questions. Table 6 shows the attributes as well as the real-to-ordinal mappings when applicable.

TABLE 6. DIABETES ATTRIBUTES AND *K*-VALUES.

$f(\mathbf{x}) = f(x_1,x_2,x_3,x_4,x_5,x_6,x_7) = d$, $k = 2$
(x_1) Age, $k = 4$ ($x_1 < 24$; $24 \le x_1 < 43$; $43 \le x_1 < 63$; $63 \le x_1$)
(x_2) Hypertension, $k = 2$
(x_3) Heart disease, $k = 2$
(x_4) Smoking history, $k = 3$
(x_5) BMI, $k = 4$ ($x_5 < 23.6$; $23.6 \le x_5 < 27.3$; $27.3 \le x_5 < 29.6$; $29.6 < x_5$)
(x_6) Hemoglobin A1C, $k = 4$ ($x_6 < 4.8$, $4.8 \le x_6 < 5.8$, $5.8 \le x_6 < 6.5$, $6.5 < x_6$)
(x_7) Blood glucose, $k = 4$ ($x_7 < 100$; $100 \le x_7 < 126$; $126 \le x_7 < 159$; $159 < x_7$)

The MBF was restored with 225 questions, 7.33% of the total *n*-D points. Heuristics 2 and 4 were used (LHCF starting from the top of each chain) as well as true attributes for x_6 and x_7 since those need to have a value of 3 and 2, respectively, in order to match clinical practice. The MBF was:

$$d = 1 = (x_6 \ge 3)(x_7 \ge 2),$$

which is quite simplistic compared to the gout model. The other attributes were not used because they seem not important for a strict diagnosis of diabetes. In hindsight, the attributes $x_{1\text{-}5}$ and $x_{6\text{-}7}$ could have been grouped together for a significant reduction in questions, and it's debatable if MOEKA is useful in this case due to the simplicity of the situation. However, the accuracy of the expert model for $d = 1$ was 94.8%, which is quite surprising, considering only two attributes were used to build the model. This brings up the question of the necessity of building advanced ML models on some medical data with very large dimensions. It also stresses the importance of expert knowledge in building useable models, when a few very strong attributes are present.

Furthermore, the SME considers some of the attributes that were not present in $d = 1$ to be important for additional treatment. We expand our MBF to an MOF with $k_{n+1} = 3$, where $d = 2$ is diabetes with additional treatment due to potential complications. The MOF was

$$d = 2 = (x_2)(x_6 \ge 3)(x_7 \ge 2) \vee (x_3)(x_6 \ge 3)(x_7 \ge 2) \vee (x_4 \ge 1)(x_6 \ge 3)(x_7 \ge 2).$$

The MOF for $d = 2$ was restored with 314 questions, 10.2% of the total *n*-D points. This was done with the same order heuristics to be consistent with $d = 1$. The accuracy for $d = 2$ cannot be confirmed because the target attribute of the dataset we used is Boolean, but the SME was confident in the function.

C. Experimental Graph-based Techniques

The goal of this case study is to attempt to reduce the number of questions by treating the HCs as nodes in a complete graph and then order chains by applying chain ordering algorithms presented below.

The hypothesis is that if an oracle answers questions from one chain, then the next chain in the generated order of chains will likely have more expanded points based on these answers than chains that are distant in the order. The proposed order is based on some measure of similarity or **linkage between chains**. Linkage is defined as a sort of "distance" between chains. Each *linkage* is a numeric label assigned to the edge between two nodes in the graph.

The first order algorithm is **Minimal Spanning Tree Order** (MSTO) based on the Minimal Spanning Tree, a tree that spans each vertex in a graph with the minimal number of edges, so there are no cycles. Each vertex is a single chain. Linkages between chains are computed using methods described below. We randomly select a start node/chain and apply Kruskal's algorithm. The next node added to the order belongs to the next linkage with the smallest sum when added to the previous linkage(s). The process of ordering continues by adding all other nodes/chains sequentially with the smallest sum of linkages with the constraint that there are no loops, i.e., the MST is preserved.

The second order algorithm is the **Shortest Path Order** (SPO) which is the shortest path between chains. The motivation is that there may be information to pick up a particular chain as a start chain.

The third order algorithm is the **Linkage Sort Order** (LSO) where the edges are ordered by ascending linkage order, and the corresponding nodes are picked as they are encountered. This order is the first step of Kruskal's algorithm. The motivation is based on the simplicity of this method.

Furthermore, to treat each Hansel chain as a node with linkages, we need to compute linkages between each chain in a weighted complete graph.

Hamming Chain Linkage (HCL) is computed as a sum of the Hamming Distances between pairs of points that are in the same relative position between the two chains of equal length, e.g., index location 0 in chain *A* will be paired with index location 0 in chain *B*. If the chains are of unequal length, then the matched pair of points is determined by equal Hamming norm of the points.

The next linkage used is **Hausdorff distance**. The informal definition of Hausdorff distance is that it is the greatest distance from a point in one set to the closest point in another set. This can also be described as the maximum of the minimum distance between pairs of points.

The third linkage is **Group Average Distance**, which is the summation of the Hamming distance between each pair of points between two chains. The sum is then divided by the product of the lengths of the chains. Full details of this distance can be found in [23]. However, this distance not necessary will be a distance for HCs.

Next, two functions were tested. The first function is

$$x_1x_3x_5 \vee x_2x_3x_5 \vee x_4x_5 \quad (1)$$

The Boolean search space is 32 discrete points for $n = 5$. The second function is a continuous function is

$$f(\mathbf{x}) = \sum_{i=0}^{n} x_i \quad (2)$$

for $n = 7$, and each attribute x_i has three values. The search space is 2187 discrete points. Each point is sorted according to the value of $f(\mathbf{x})$, and the bottom third are mapped to a value of 0, the next third are mapped to 1, and the last third are mapped to 2.

This creates an MOF with hundreds of clauses. In practice with SMEs, this type of extremely long function does not generally appear, so the number of questions performed is significantly higher than normal. Tables 7-10 present results of these case study.

TABLE 7. FUNCTION 1 MSTO WITH SHCF CONTROL OF 14 QUESTIONS.

Linkage Metric	Number of Questions
HCHD	13
Hausdorff Distance	11
Group Average Distance	12

TABLE 8. FUNCTION 2 MSTO WITH SHCF CONTROL OF 1107 QUESTIONS.

Linkage Metric	Number of Questions
HCHD	1096
Hausdorff Distance	1076
Group Average Distance	1107

TABLE 9. FUNCTION 2 SPO WITH SHCF CONTROL OF 1107 QUESTIONS.

Linkage Metric	Number of Questions
HCHD	1104
Hausdorff Distance	1107
Group Average Distance	1107

TABLE 10: FUNCTION 2 LSO WITH SHCF CONTROL OF 1107 QUESTIONS.

Linkage Metric	Number of Questions
HCHD	1101
Hausdorff Distance	1076
Group Average Distance	1107

The best result is by using Hausdorff distance with either LSO or MSTO. Both turned out to be very similar due to the way that chains were rearranged according to the MST as edges were added. It likely is a better idea to rearrange the chains according to a depth-wise traversal of the MST. Although the chains were rearranged, when using group average distance, the chains tended to be ordered in ascending value of size (similar to SHCF), and HCHD tended to be ordered in descending value of size (similar to LHCF). This explains why some of the results were similar to the results for SHCF. Some further research is needed with the distance metrics. For example, we can measure distance by how many points can be expanded from one HC to another. Instead of using generic metrics like Hausdorff distance, it should be possible to take advantage the properties of Hansel chains.

D. Housing

In this case study we consider a situation where a set of features is defined by a potential buyer of a house. The buyer needs to identify their preferences for buying the house, which can be defined as a function by using MOEKA. This function can be used to search a database to find potential options.

Traditionally, a buyer would identify a limited set of characteristics to search manually through some set of houses. Online real estate marketplaces offer options to filter for some set of features, but it can still be difficult for the user since the filtering process is not sophisticated enough to decrease the search space since it usually consists of a function with a single clause of "&" statements. With MOEKA we can create much more sophisticated functions with multiple clauses. There are many other situations beyond housing where a search in the database can be much richer with a more sophisticated Boolean and/or k-valued function(s). Thus, this cases study is generalizable.

In this task the first author served as a "SME". The attributes that are used are taken from a Kaggle dataset that is meant for housing price classification [26]. Continuous attributes were given $k = 4$ and based on quartiles. The search space is 885,168 questions before grouping, and the groups were arbitrarily split on "room qualities" and "other qualities," though other groupings can certainly be used. The two super-attributes have $k = 3$ for "would not buy," "might buy," and "would buy." The attribute x_3 is on the same level as the two super-attributes because of its importance over the rest of the attributes. The housing price attribute is inverted because a "good price" is a "low price." The main road attribute and number of stories attribute were also inverted since this SME would prefer a small house away from a main road. The overall housing preference MOF has $k = 2$ for "would not buy" and "would buy." The hierarchical MOF model is shown below:

$$f(\mathbf{x}) = f(x_1,x_2,x_3) = d,$$

$$x_1 = h_1(y_1,y_2,y_3,y_4),\ \ x_2 = h_2(z_1,z_2,z_3,z_4,z_5,z_6).$$

Table 11 shows the attributes and their k-values.

TABLE 11: HOUSING ATTRIBUTES AND k-VALUES.

x_1: function b_1 (rooms), $k = 3$	x_2: function b_2 (other), $k = 3$
(y_1) Bedrooms, $k = 6$	(z_1) Area, $k = 4$
(y_2) Bathrooms, $k = 4$	(z_2) Main road, $k = 2$
(y_3) Guestroom, $k = 2$	(z_3) Number of stories, $k = 4$
(y_4) Basement, $k = 2$	(z_4) Hot water heating, $k = 2$
(y_5) Air conditioning, $k = 2$	(z_5) Parking spots, $k = 3$
(y_6) Furnishing status, $k = 3$	(z_6) Preferential area, $k = 2$
x_3: price, $k = 4$	

The total MOF $f(\mathbf{x})$ was restored with a total of 148 questions:

$$f(\mathbf{x}) = (x_2\geq 1)(x_3\geq 3) \vee (x_2\geq 2)(\, x_3\geq 2) \vee (x_1\geq 1)(x_2\geq 1)(x_3\geq 2).$$

The inner functions h_1 and h_2 are given in tables 11-12.

TABLE 11. HOUSING MOF INNER FUNCTIONS FOR x_1 (ROOM QUALITIES).

Value of x_1	MOF $h_1(x_1)$
1	$(y_1\geq 1)(y_2\geq 3)(y_3) \vee (y_1\geq 1)(y_2\geq 1)(y_3)(y_4) \vee (y_1\geq 1)(y_2\geq 1)(y_3)\,(y_6\geq 1) \vee (y_1\geq 1)(y_2\geq 1)(y_4)(y_6\geq 1) \vee (y_1\geq 1)(y_2\geq 2)(y_6\geq 1) \vee (y_1\geq 1)(y_2\geq 1)(y_6\geq 2) \vee (y_1\geq 1)(y_2\geq 2)(y_4) \vee (y_1\geq 1)(y_2\geq 1)(y_5) \vee (y_1\geq 2)(y_2\geq 1)(y_4) \vee (y_1\geq 2)(y_2\geq 1)(y_3) \vee (y_1\geq 3)(y_2\geq 2) \vee (y_1\geq 4)(y_2\geq 1)$
2	$(y_1\geq 1)(y_2\geq 3)(y_3)(y_6\geq 2) \vee (y_1\geq 1)(y_2\geq 2)(y_3)(y_4)(y_6\geq 2) \vee (y_1\geq 1)(y_2\geq 3)(y_4) \vee (y_1\geq 2)(y_2\geq 1)(y_4)(y_6\geq 2) \vee (y_1\geq 2)(y_2\geq 3)(y_3)(y_6\geq 1) \vee (y_1\geq 3)(y_2\geq 3)(y_6\geq 2) \vee (y_1\geq 1)(y_2\geq 2)(y_5) \vee (y_1\geq 1)(y_2\geq 1)(y_3)(y_5) \vee (y_1\geq 1)(y_2\geq 1)(y_4)(y_5) \vee y_1\geq 1)(y_2\geq 1)(y_5)(y_6\geq 1) \vee (y_1\geq 3)(y_2\geq 1)(y_5) \vee (y_1\geq 3)(y_2\geq 1)(y_4) \vee (y_1\geq 4)(y_2\geq 1)(y_3) \vee (y_1\geq 3)(y_2\geq 3)(y_3) \vee (y_1\geq 4)(y_2\geq 2)(y_6\geq 1) \vee (y_1\geq 4)(y_2\geq 3) \vee (y_1\geq 5)(y_2\geq 1)$

TABLE 12. HOUSING MOF INNER FUNCTIONS FOR x_2 (OTHER QUALITIES).

Value of x_2	MOF $h_2(x_2)$
1	$(z_1\geq 1)(z_3\geq 3)(z_4)(z_5\geq 2)(z_6) \vee (z_1\geq 1)(z_2)(z_3\geq 3)(z_4)(z_5\geq 1)(z_6) \vee (z_1\geq 1)(z_2)(z_3\geq 3)(z_5\geq 2)(z_6) \vee (z_1\geq 2)(z_3\geq 3)(z_5\geq 2)(z_6) \vee (z_1\geq 2)(z_2)(z_3\geq 3)(z_5\geq 1)(z_6)$
2	$(z_1\geq 2)(z_2)(z_3\geq 3)(z_4)(z_5\geq 2)(z_6) \vee (z_1\geq 3)(z_2)(z_3\geq 3)(z_4)(z_5\geq 1)(z_6)$

There were zero entries in the housing database of 545 entries that matched the expert model. There were 393 entries that matched inner function h_1, zero entries for inner function h_2, and 270 entries that matched the desired price. When simply ignoring h_2 in the expert model, so $f(\mathbf{x}) = (x_1\geq 1)(x_3\geq 2)$, there were 148 entries (27.1%) in the database that matched the expert model. h_2 was replaced with several simplified

functions: $h_2(\mathbf{z}) = 1 = (z_6)$ yielded just 18 entries, $h_2(\mathbf{z}) = 1 = (z_4)(z_6)$ had 1 entry, and $h_2(\mathbf{z}) = 1 = (z_1 \geq 1)(z_6)$ had 2 entries. This shows the possible benefit of MOEKA for these types of searches due to the very low hit rate for more complex MBF/MOF.

Several heuristics were used to obtain these results. SHCF from the bottom element of the chain was used for $f(\mathbf{x})$ and h_1 since the SME suspected that a low number of attributes and values were needed to restore the function for $f(\mathbf{x})$ and h_1. However, for h_2 LHCF from the top element of the chain was used since the SME suspected that a large number of attributes and valuers were needed. True attributes were also used for h_1 and h_2.

Furthermore, only 0.2% of the upper limit of questions was needed to restore the function, the best result yet. This is likely due to the grouping of attributes, the inversion of some attributes, as well as using more true attributes than some of the other case studies. One area of concern is the fact that 148 questions is a lot if the task revolves around complicated subject matter (such as medical diagnostics). In these cases, it can be worth trying to identify patterns in the questions before the interview in order to expedite the process. For example, many times one question will lead to a very similar question, so the question can be phrased more succinctly, such as "this case is the previous case except x_1 is equal to 2." It can also be worth attempting to do the interview quickly, potentially at the cost of the performance of the expert model. This model can be tested and the "correct" answers can be saved, and potentially "wrong" answers can be reexamined in another interview. The second interview will be much shorter due to saving the results from the first.

VII. Conclusion and Further Work

Due to some SMEs not being able to trust ML models, MOEKA is a promising way to solve this problem by building a trustable, qualitative interpretable model for the task with domain experts. Since such "expert models" are based on SME knowledge, we can use them to help to understand, interpret, trust, and/or improve the ML models built from data. Visualization of the model can help this as well. One constraint to MOEKA is that only MBFs and MOFs can be used. Although some study has been done on non-monotonic noise in the literature, the focus of this paper has been on what sort of orders to use with MOEKA to create expert models with some example heuristics and case studies.

Further study includes developing and evaluating more operation reduction techniques, performing new cases studies, exploring the differences between expert models and ML models built on real data, and handling non-monotonic noise. Further analysis and testing with the above reported test studies can be done as well.

VIII. References

[1] Dikmen, M., & Burns, C. (2022). The effects of domain knowledge on trust in explainable AI and task performance: A case of peer-to-peer lending. *International Journal of Human-Computer Studies*, *162*, 102792.

[2] Levy, A., Agrawal, M., Satyanarayan, A., & Sontag, D. (2021, May). Assessing the impact of automated suggestions on decision making: Domain experts mediate model errors but take less initiative. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (pp. 1-13).

[3] Zewe, A. MIT News Office, A. Z. (2020, June 30). *Building explainability into the components of machine-learning models*. MIT News, Massachusetts Institute of Technology.

[4] Lones, M. A. (2021). How to avoid machine learning pitfalls: a guide for academic researchers. *arXiv preprint arXiv:2108.02497*.

[5] Kovalerchuk, B., Vityaev, E., & Ruiz, J. F. Integrated consistent and complete "expert" and data mining, in: Medical Data Mining and Knowledge Discovery, Springer, 2001. pp. 238-281.

[6] Starkhagen, C. (2022, July 7). *Qualitative data: The Unsung Hero of Machine Learning Datasets*. Twine Blog.

[7] Tenny, S., Brannan, J. M., & Brannan, G. D. (2017). Qualitative study.

[8] Forbus, K. D. (2011). Qualitative modeling. *Wiley Interdisciplinary Reviews: Cognitive Science*, *2*(4), 374-391.

[9] Gitis, V., Derendyaev, A., Petrov, K., Yurkov, E., Pirogov, S., Sergeeva, N., ... & Kaprin, A. (2021). Monotonic Functions Method and Its Application to Staging of Patients with Prostate Cancer According to Pretreatment Data. *Applied Sciences*, *11*(9), 3836.

[10] Gupta, M., Cotter, A., Pfeifer, J., Voevodski, K., Canini, K., Mangylov, A., Van Esbroeck, A. (2016). Monotonic calibrated interpolated look-up tables. *Journal of Machine Learning Research*, *17*(109), 1-47.

[11] Milani Fard, M., Canini, K., Cotter, A., Pfeifer, J., & Gupta, M. (2016). Fast and flexible monotonic functions with ensembles of lattices. *Advances in neural information processing systems*, *29*.

[12] Potharst, R., Bioch, J. C., & Petter, T. (1997). Monotone decision trees. *Department of Computer Science, Erasmus University Rotterdam*.

[13] Zhu, H., Zhai, J., Wang, S., & Wang, X. (2014). Monotonic decision tree for interval valued data. In *Machine Learning and Cybernetics: 13th International Conference, Lanzhou, China, July 13-16, 2014. Proceedings 13* (pp. 231-240). Springer Berlin Heidelberg.

[14] Potharst, R., & Feelders, A. J. (2002). Classification trees for problems with monotonicity constraints. *ACM SIGKDD Explorations Newsletter*, *4*(1), 1-10.

[15] Ben-David, A. (1995). Monotonicity maintenance in information-theoretic machine learning algorithms. *Machine Learning*, *19*, 29-43. [16] Cano, J. R., Luengo, J., & García, S. (2019). Label noise filtering techniques to improve monotonic classification. *Neurocomputing*, *353*, 83-95.

[17] Zitikis, Y. D. R. (2005). An index of monotonicity and its estimation: a step beyond econometric applications of the Gini index. *Metron*, *63*(3), 351-372.

[18] Verbeke, W., Martens, D., & Baesens, B. (2017). RULEM: A novel heuristic rule learning approach for ordinal classification with monotonicity constraints. *Applied Soft Computing*, *60*, 858-873.

[19] U.S. Department of Health and Human Services. (2024, April 24). *Gout*. National Institute of Arthritis and Musculoskeletal and Skin Diseases. https://www.niams.nih.gov/health-topics/gout

[20] Milstein, I., Ben-David, A., & Potharst, R. (2014). Generating noisy monotone ordinal datasets. *Artif. Intell. Res.*, *3*(1), 30-37.

[21] Kovalerchuk, B., Triantaphyllou, E., Deshpande, A. S., & Vityaev, E. (1996). Interactive learning of monotone Boolean functions. *Information Sciences*, *94*(1-4), 87-118.

[22] Kovalerchuk, B., Delizy, F., Riggs, L., & Vityaev, E. (2012). Visual data mining and discovery with binarized vectors. *Data Mining: Foundations and Intelligent Paradigms: Volume 2: Statistical, Bayesian, Time Series and other Theoretical Aspects*, 135-156.

[23] Fujita, O. (2013). Metrics based on average distance between sets. *Japan Journal of Industrial and Applied Mathematics*, *30*, 1-19.

[24] Vargas-Santos, A. B., Taylor, W. J., & Neogi, T. (2016). Gout classification criteria: update and implications. *Current rheumatology reports*, *18*, 1-10.

[25] Mustafa, M. (2023, April 8). *Diabetes prediction dataset*. Kaggle.

[26] H, M. Y. (2022, January 12). *Housing prices dataset*. Kaggle.

[27] Freiesleben, Timo, and Thomas Grote. "Beyond generalization: a theory of robustness in machine learning." *Synthese* 202.4 (2023): 109.

[28] Kovalerchuk, B. Interpretable AI/ML for High-stakes Tasks with Human-in-the-loop: Critical Review and Future Trends. 2024, https://www.researchsquare.com/article/rs-3989807/latest.pdf.

[29] Delgrande JP, Glimm B, Meyer T, Truszczynski M, Wolter F. Current and Future Challenges in Knowledge Representation and Reasoning, 2023, https://arxiv.org/pdf/2308.04161