ExploreASE: Sway and Xpln Modifications

Sanjana Cheerla, Leo Hsiang, Natalie Kraft, Ashrita Ramaswamy
North Carolina State University, USA
scheerl@ncsu.edu, yhsiang@ncsu.edu, nakraft@ncsu.edu, aramasw@ncsu.edu
https://github.com/nakraft/exploreASE

***Abstract*—All engineers work tirelessly to optimize their processes; for software engineers, this means integrating efficient, time-saving algorithms into their workflow. While operationalizing AI/ML is not a new field, it continues to drive innovation and support development in interdisciplinary fields. When optimizing these systems, stakeholders all share one common problem: the exponential cost of information gain.**

Stakeholders can better prioritize this information by utilizing semi-supervised explanation systems. To improve state-of-the-art techniques, we seek to challenge the smoothness assumption held by most heuristics and improve algorithm run-time efficiency for explanation systems.

I. Introduction

As stakeholders look to improve their systems, they first attempt to learn more about their environments. From improving the set of features customers most enjoy to optimizing the processes which account for production delays, stakeholders' cost for prioritizing opportunities for improving their efficiency is constantly growing. As new information comes to light, the paths forward are multiplied and the search space for finding the best solution continues to grow. The obvious problem with this approach is that the probability that the best solution will be found is very slim [7].

Taking a data-driven approach to this problem, by developing a semi-supervised explanation system, stakeholders can better prioritize information while simultaneously not having to navigate the search space linearly. Rather than trying to optimize, our proposal is to identify a set of heuristics that can be utilized in an algorithmic approach to solving this 'fishing problem' [7]. Our approach challenges the smoothness assumption of supervised learning and looks to build upon the field by experimenting with randomness to learn about heuristics that may better generalize in interdisciplinary areas. This insight is driven by economic theory.

The smoothness assumption indicates that points that remain close together in the input space have similar outputs [2]. This is similar to the continuity assumption which states that points close together in the input space will likely receive the same class label during a classification task [15]. However, the behavioral economic theory argues that people's utility functions - simplistically speaking their level of happiness - do not have to be continuous. People whose indifference curves, a relative mapping between those commodities, behave in this manner and are labeled indifferent [6]. This is one of the assumptions `sway` makes; through modifications to `sway`, we hope to test this assumption. To address this, we added in randomness to `sway` to contradict the smoothness assumption by looking for clusters not based on a far-apart distance.

In addition, we look to support the scalability of explainable models by optimizing run time in `xpln`. By increasing the size of the search space that can be explored while keeping the time allotted constant, we can find a preferred result or reach a satisficing result in less time [2].

By adding in randomness - as addressed within the modified `sway` - and decreasing redundancies - as experimented throughout `xpln` - we have reached the conclusion that semi-supervised models can support stakeholders prioritization of tasks and contribute to easing the burdens of the *fishing problem*.

Our research questions are as follows:

- Does the smoothness assumption hold for the data sets being explored?
- What is a satisficing level of randomness and at what stage should it be integrated in with modeling to retrieve generalizable results?
- How much of an impact does parameter tuning and algorithm redefining have on generalizing our data?
- How does human evaluation compare to our algorithmic output in terms of explainability and efficiency?

The approach identified in this paper proves that the smoothness assumption does not hold in all cases due to the possibility of discrete utility functions. By integrating economic theory into algorithm development, we have explored a new construct within our search space. Despite our interest in quantifying the extent to which randomness could improve our search space, this was not an action the study undertook due to run time constraints. Furthermore, this study does not suggest that all data sets are multimodal, but rather that the data sets should be explored further before being constrained to this ideal. By providing evidence to these claims, we hope to advance the notion that heuristics should not be unimodal in nature while advocating for additional research in semi-supervised learning dedicated toward exploring the applications of integrating behavioral economic theory.

II. Related Work

A. Literature Review

Semi-Supervised Multi-Objective Models have become increasingly popular as the search space to identify Pareto-optimal solutions have grown. Despite this coming-of-age, few engineers scrutinize which algorithm would best apply to their data, opting to use algorithms instead based on commonality [13].

Frequently, to meet semi-supervised learning objectives, one will turn to the identification of heuristics that support moving

us closer to the Pareto frontier. Philosophers have contributed to this field through the heuristic cognition claim, which states real-world probabilities are not reliably determined, and when integrating multiple goals of interest, calculating assumed probabilities cannot be done efficiently [7]. Thus, heuristics are valuable as they support the identification of indicators that can identify if a solution is good enough. This form of satisficing is where the idea for multi-objective semi-supervised learning was born, and allows for the solving of computationally intractable problems [2], [7].

Utilizing heuristics - and other algorithmic clustering techniques - allows for a reduction of overfitting, better robustness and generalization, and increased efficiency when exploring the search space [2], [13]. When relating this back to the described *fishing problem*, the aforementioned characteristics can all support prioritization of feature selection. Other techniques to support these goals can be found through dimensionality reduction mechanisms, clustering, and generative modeling [2].

One method for dimensionality reduction includes random projections, which utilizes the Johnson-Lindenstrauss lemma to project points in a high-dimension vector space into a lower-dimension space while preserving the points' pairwise distance [8]. The benefits are that lower-dimensional spaces are less time-consuming to search and irrelevant - or collinear - features are typically removed from the input space together [2], [10]. For clustering, researchers have found success with k-means, simulated annealing, and hierarchical feature models [2], [12], [14]. Lastly, some researchers who focus on data generation have greatly supported the field. This improves sparsity within data sets by allowing generative models to impute missing data during training. Labels are predicted using the current model, and then using the imputed answers, the model continues to train. [2] Despite this mechanism for generation, some claim this can purport overfitting when the generative model performs poorly [9].

While all of these advances researchers have made have provided valuable insight to stakeholders and have successfully saved computational resources and human labor, there is still room for improvement within the field. In experimentation, we look to challenge some of the underlying assumptions about multi-objective semi-supervised learning models to provide novel insight regarding the usefulness of randomness in improving generalizability.

B. An Informed Novel Approach

There are a couple of well-known assumptions made regarding heuristics for clustering, the fundamental problem tackled by semi-supervised learning techniques. The first is a smoothness assumption, which addresses continuity in the data space. The effect of this assumption is that algorithms expect similar inputs to have a similar output space. An additional assumption is based on the density of data within any vector space; it is assumed that the naturally-determined decision boundary for clustering lies in a low-density region. Lastly, semi-supervised learners operate with the manifold

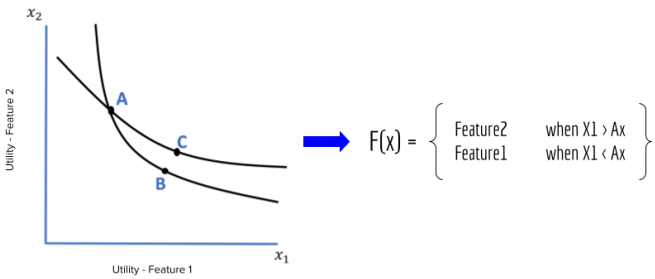

Fig. 1: Irrational Indifference Curves

assumption, which infers that high-dimensional data can be moved into a low-dimensional space while still generating quality predictions. [2] The latter assumption can be traced back to the Johnson-Lindenstrauss lemma - described in the above discussion regarding dimensionality reduction [8].

While the majority of researchers treat these assumptions as givens - as they do provide stability and generally fit the data well - we argue that there are domains that violate these assumptions. To promote better semi-supervised learning in these spaces, we look to identify opportunities for better generalizability by experimenting with the extent to which these assumptions hold.

With regards to the smoothness assumption, economic theory suggests that some decision outcomes experience multimodal distributions in which entities quite far away from each other share a similar outcome. While economists label entities that experience these characteristics to be considered 'irrational', behavioral economic specialists have identified this as quite a normal phenomenon [1]. The underlying theory utilizes the measurement of utility - simply put one's level of happiness - to suggest that there are non-quantifiable factors that would disrupt a continuous or unimodal utility function, in effect producing vastly different outputs from similar inputs. In the simplistic case displayed in Figure 1, the entity's indifference curves cross, suggesting a behavioral change in how one values two different choices. These irrational actors demonstrate that the smoothness assumption does not hold; when these indifference curves are transferred into a utility function, similar inputs may result in a different valuation.

Let's take a real-world example: traffic tickets. Some countries enforce quotas of trafficking tickets; let's assume in this example the quotas are tied to an annual bonus. In this binary representation, one officer may have ticketed one less than the quota and one may have ticketed one more than the quota. Yet despite similar inputs, the second officer's utility is infinitely happier than the other officers. This is a simple example of how the smoothness assumption may not always hold; it is this underlying economic theory that we look to challenge through experimentation with `sway`.

In the below methodology, the current implementations of `sway` and `xpln` are discussed. To address our research questions, we will be modifying these techniques and utilizing

their current outputs in comparison.

III. Methodology

This section will describe the methodology we followed. There will be a discussion of the existing `sway` and `xpln` algorithms. The discussion is then followed by the updates we made, followed by our datasets, performance measures, and summarization methods.

A. Algorithms

This subsection details the existing algorithm and the modifications we made to `sway` and `xpln`.

1) Sway1: `Sway` is a powerful supervised optimization algorithm that is used to return the optimal cluster of the data. The `sway` algorithm takes an optional argument called cols, which specifies which columns to use for clustering.
One iteration (evaluation) of the `sway` algorithm works in the following manner:

1) Find a random point (R) in the data set.
2) Find the furthest point (F) from (R). Calculate the distance (D) from (R) to (F).
3) Find all the points that are within 95% of the distance (D) from point (R).
4) Find points (A and B) that are furthest away from (R).
5) Draw a line (L) from (A) to (B).
6) Project remaining data onto line (L).
7) Split the data into right and left halves. (A) is the representative of the left half, and (B) is the representative of the right half.
8) Use the Zitzler domination predicate function to determine which point (A or B) is better.
9) Depending on which point (A or B) is better, remove the data belonging to the other points. If the point (A) is determined as better, the right half which (B) belongs to is thrown away.

The algorithm continues until the stopping criterion is met. The stopping criterion is the variable called min, which is the size of the best cluster. The algorithm returns the best cluster and the rest of the data.

There are a couple of problems with `sway`. First, `sway` does not attempt to ensure the decision boundary lies in a low-density area, which is seen as a practical characteristic to ensuring clusters are explainable [2]. The second issue with `sway` is that we maintain the assumption that categorization occurs with a unimodal distribution. While the `sway` authors acknowledge `sway`'s simplicity and support its quick runtime, we believe there are ways to keep `sway`'s general structure while improving the aforementioned issues [4].

2) Sway2: For `sway2`, we wanted to challenge the smoothness assumption by changing points (A) and (B) to be selected randomly rather than utilizing a distance metric.

Therefore, one potential improvement could be to use the random.sample() function to randomly select two points to use as (A) and (B). This would ensure that both points are selected randomly and could potentially improve the quality of the split should a multimodal data distribution exist. This change is depicted in the pseudocode in Figure 2 with the bold markings.

```
function SWAY2(data, cols=None, min=None):
  data = cols ? data[:, cols] : data
  cluster = [], evals = 0
  while len(data) > min:
    r = random_point(data)
    d = distance(r, furthest_point(data, r)
    candidates = in_dist(data, r, .95*d)
    a, b = random(candidates, 2)
    left, right = project_line(candidates,
          line(a, b))
    better = zitzler_domination(a, b)
    data = better ? left : right
    cluster.append(better), evals++
    line = line_through(a, b)
```

Fig. 2: High level pseudocode of Sway2

3) Xpln1: Overall, the `xpln1` function is used to find the best rule for explaining a given set of data by performing equal-width discretization on the data, calculating the value of each range, selecting a subset of the ranges, and then applying the score function to find the best rule.

As seen in Figure 3, `xpln1` takes in three parameters - data, best, and rest - which are instances of the Data class representing a set of data to be explained, the "best" subset, and the "rest" subset, respectively. The function uses a *v function* to calculate the value of a rule given the number of rows in best and rest that the rule applies to. It then discretizes each column in data into a set of ranges using the bins function and calculates the value of each range using the *v function*. The function sorts the ranges in descending order and selects a subset of them using the *firstN function*, and then applies the score function to the selected subset to calculate the best rule. The function returns the best rule and its most value. The most value is a score between 0 and 1, this value explains how much of the best data can be explained using the generated rule.

The *firstN function* (fig 3 pseudocode) takes a sorted list of dictionaries (sorted_ranges) and a scoring function (scoring_function) as input. It returns a rule and a score. It selects the first range that has a value greater than 0.05 and is greater than 1/10th of the first value in the list. It then calls the scoring function with an increasing number of ranges (from one to the length of the list). If the score returned by the scoring function is greater than the previous maximum score, it updates the rule and the maximum score. The final rule and score are returned.

4) Xpln2: In order for stakeholders to find a satisfactory result in less time, we wanted to focus on reducing the time complexity of `xpln1` for our `xpln2` modification. After careful analysis of where the majority of run time was spent, we noted in `xpln1` (on line 10 of Figure 3) that the sorting is done by using the default sorting algorithm in Python which is *timsort*). We updated this function, as seen on line 10 of Figure 4 to use *heapsort*. This ensured that instead of appending each dictionary to the tmp list and then sorting the entire list, we

```
function Xpln1(data, best, rest):
   function v(rule, best, rest):
        calculate number of rows in best, rest
        return value of the rule
    tmp = []
    for col in data.columns:
        ranges = bin(data[col])
        // calculate v for each row
        tmp.append(v(rule, best, rest))
    sorted(tmp)
    score = score of calculation of best/rest
    best_rule = firstN(tmp, score)
    return best_rule, best_value

function firstN1(sortedRanges, scoring_function):
    rule, score = -1
    tmp = []
    for val in sortedRanges:
        if val > 0.05 and val > first / 10
        tmp.append(val)
    for t in range(tmp):
        x, rule = scoring_function()
        if x is not none and x > rule:
            rule, score = rule, x
    return rule, score
```

Fig. 3: High level pseudocode of Xpln1 and firstN1

used the *heapq.nlargest* function to get the self.score largest dictionaries directly.

Furthermore, in *firstN1*, lines 18-20 go through the entire sorted ranges to find the rows that meet certain criteria. We realized that this could be optimized by performing a search that uses the sorted ranges. We used binary search to get the rows in *firstN2*, which reduced the run time significantly. This change can be seen on line 18 of Figure 4.

```
function Xpln2(data, best, rest):
   function v(rule, best, rest):
        calculate number of rows in best, rest
        return value of the rule
    tmp = []
    for col in data.columns:
        ranges = bin(data[col])
        // calculate v for each row
        tmp.append(v(rule, best, rest))
    heapsort(tmp)
    score = score of calculation of best/rest
    best_rule = firstN(tmp, score)
    return best_rule, best_value

function firstN1(sortedRanges, scoring_function):
    rule, score = -1
    tmp = []
    tmp.append(binarySearch(sortedRanges))
    for t in range(tmp):
        x, rule = scoring_function()
        if x is not none and x > rule:
            rule, score = rule, x
    return rule, score
```

Fig. 4: High level pseudocode of Xpln2 and firstN2

B. Data

11 data sets were explored to identify the impact of algorithmic changes across disciplines. The data set overview is found in Table I. Our algorithms looked to minimize or maximize the output values specified within the data sets. For simplicity, it was assumed that feature inputs and output variables had minimal collinearity (refer to the February study where we address how collinearity simplifies our analysis).

Data set	#Inputs	#Outputs	size	domain
auto2	19	4	93	car attributes
auto93	5	3	398	car attributes
china	18	1	499	software management
coc1000/ 100000	20/22	5 / 3	1000 / 10000	software project estimation
nasa93dem	25	4	93	software efforts
healthCloseless	5	3	10000	issue tracking
pom	10	3	10000	agile project management
SSM	13	2	239360	computational physics
SSN	17	2	53662	computational physics

TABLE I: Table Detailing Data Sets

C. Performance Measures and Summarization Methods

Should our hypothesis regarding multimodal data distributions be correct, we anticipate that `sway` will add enough randomness to the clustering scheme to improve our output's generalization. To attest to this, we plan to use the sampling tax and explanation tax to aid in our evaluation. Sampling tax is the accuracy cost incurred by decreasing the sample size of the training data, leading to a model being underfit. The explanation tax is the additional cost incurred for simplifying the model to support better model intuition. Utilizing these metrics, we will declare our modified `sway` and `xpln` algorithms to have either outperformed (won), underperformed (loss), or tied the original `sway` and `xpln` methods.

In order to accurately rank our features, we are utilizing Scott Knott. Scott Knott is a statistical analysis method used to understand how much the results actually vary. We generated Scott Knott tables for every column in every data set to understand the variance metrics. In this case, we wanted to compare how well `sway2` and `xpln2` performed against `sway1` and `xpln1`. Looking at the Scott Knott tables (found in the raw data performance metrics), we noted that our method performed equally as well, if not better, in some cases due to the randomness factor we added in.

In addition, to support scalability through improving algorithmic efficiency in `xpln`, we will provide run-time characteristics between the modified and original algorithms.

All raw data and performance metrics can be found at *https://github.com/nakraft/exploreASE/tree/main/etc/out*.

IV. Results

By challenging the smoothness assumption, we further Sayyad and Ammar's conclusion that each data set and domain should be approached as unique to determine what best applies

to it [13]. Some data sets did outperform the comparison models, while others failed to do so. Selecting the best clustering approach based on the data domain will improve stakeholders' capabilities. However, overall, our results demonstrate that the modifications to `sway` and `xpln` resulted in a normally distributed change from the original algorithms (see Figures 5 and 6). While no significant improvements to the algorithms accuracy were made, we did reduce the complexity of the algorithms without harming performance. In `xpln2`, we managed to reduce the run time by an average of 14%, which can be seen in Figure 8. As the selection of random variables is less complex than the calculation of pairwise distance, run time can increase while model performance remains constant. This should be considered a positive result of `sway2` and `xpln2`.

A. Result Tables

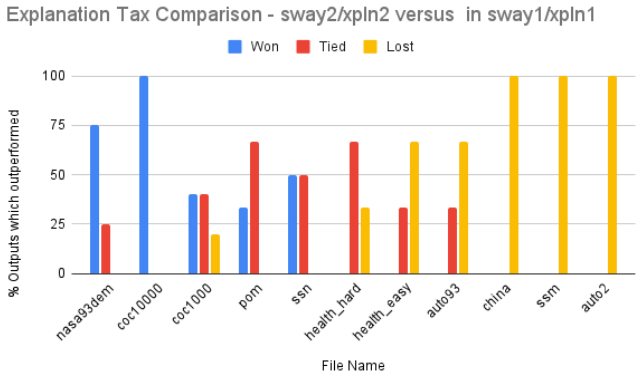

Fig. 5: Changes in Explanation Tax with Modifications to Sway/Xpln

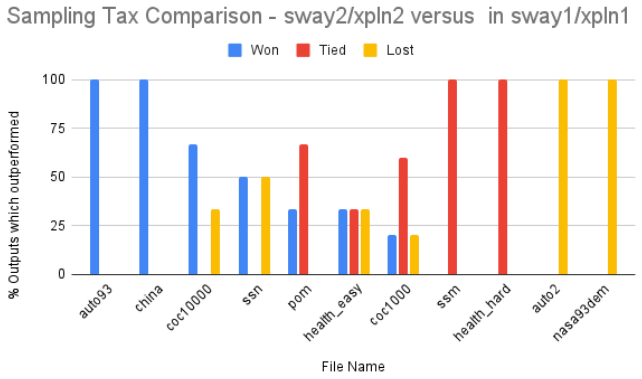

Fig. 6: Changes in Sampling Tax with Modifications to Sway/Xpln

Figure 5 and 6 demonstrate a comparison between `sway1`/`sway2` and `xpln1`/`xpln2`. The graphs tally the % of the data sets output variables which were positively (won) or negatively changed (lost). Results show a normal distribution, allowing the conclusion that reducing algorithmic complexity by eliminating the need to compute pairwise distances did not affect model performance.

B. Scalability

1) Prudence Study: To further support the evidence brought about by the sampling tax, a prudence study has been conducted on the *SSN* data set, a sufficiently large data set to allow for the iteration through different sample sizes. As the sampling size increases, we expect to see an improvement in the sampling and explanation tax.

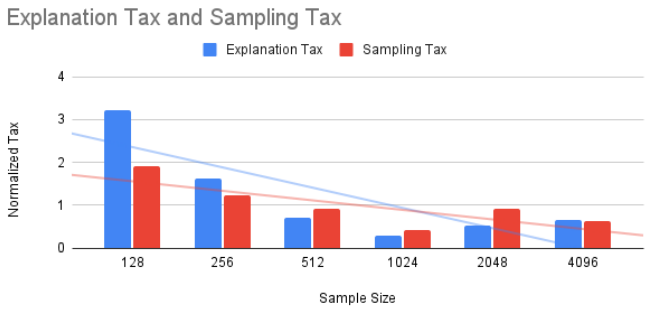

Fig. 7: Sampling Tax Evaluated With Increasing Sample Size

As shown in Figure 7, the explanation tax and sample tax both decreased - as expected - for the output variable PSNR within the *SSN* data set when processed with `sway2`. This prudence test demonstrates an accurate display of the algorithmic modifications.

2) XPLN Time Changes: In addition, to support increased efficiency in navigating the search space, we developed `xpln2`, which produced an average 14.4% percent time decrease. Figure 8 summarizes this decrease across our data sets; full results can be seen within *exploreASE/etc/out* in the git repository [3].

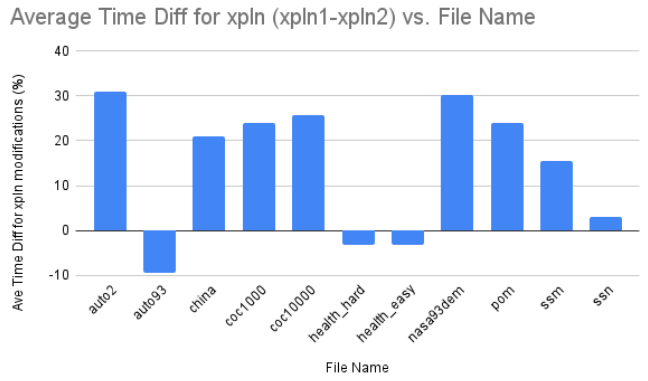

Fig. 8: Changes in Run Time with Modifications to XPLN

C. Meeting Research Objectives

As `sway2` and `xpln2` outperformed original methods utilizing randomness, the smoothness assumption does not

hold for all of the tested data sets. Furthermore, through experimentation, we identified 10 samples were the ideal amount of randomness to be integrated into the data sets to provide for clustering within `sway2`. To generalize our results, we integrated this repetitively in the model - beginning prior to any clustering - to ensure that partitioning of the feature space wouldn't overfit.

Through this experimentation, we met our research objectives and concluded that current mechanisms have unnecessary run-time requirements for their output. By redefining the algorithms, we can increase efficiency and provide immediate value to stakeholders looking to address their *fishing problem*.

V. Discussion

`Sway` and `xpln` are two algorithms that are valued for their explainability and simplicity in supporting multi-objective semi-supervised learning. While these algorithms may not be state-of-the-art in terms of accuracy, they truly are a unique way of clustering information based on its most important features in a way that is intuitive [4].

A. Threats to Validity

One threat to this study's validity is based on the sheer number of multi-objective semi-supervised learning researchers who rely on the smoothness and continuity assumption. These assumptions have a solid foundation, and are put in place due to the common attributes clustering algorithms identify with data sets in different domains. The above experimentation demonstrates that these assumptions cannot be outright contradicted, however, the intuition for why they may not always be right also has a solid intuition. Despite this, behavioral economists are sometimes ridiculed for theories surrounding irrational human behavior as there is no proven way to quantify irrationality. Overall, human behavior cannot be overlooked; irrational, or not, people's perceptions are still a valid reality that would be valuable for researchers to be able to utilize in computations.

B. Future Work

Our results show that further research is needed to provide modifications to `sway` and `xpln` that will improve clustering capabilities. One element to continue down the path of challenging the smoothness assumption is in regard to the scope of random entity selection. Rather than picking random variables for points A and B, a sample size of variables could be selected, distribution metrics computed, and then a representative data point selected. Theoretically, this would decrease the variability of the points used to generate the clusters.

Another potential avenue for improvement is to first look for correlations amongst the feature space to see where the model might be prone to overfit. This ties into dimensionality reduction, and could both improve run time and decrease computational difficulty.

Lastly, to decrease the run time needed to compute distance metrics within `sway`, there may be a value in imputing suspected distances based on the previous intake of data. In this generative model proposal, we would seek to decrease in the computation time for computing these pairwise distances.

VI. Bonus Sections

A. B1: Requirements Repertory Grids Study

With the support of 5 participants, repertory grids were collected regarding the perceived status and attributes of cars to be compared against the data collected in auto2 and auto93 [11]. Raw data can be viewed within */docs/repertory_grids.md* [3]; summarized results are in Table II.

	MPG	ofCylinders	Horsepower	Weight	Acceleration	ModelYear	Status	CostPrice	Quality	Enviornmental	Features
Participant 1	5	2	2	2	2	3	2	5	4	4	3
Participant 2	4	3	3	2	4	4	4	5	5	1	4
Participant 3	5	2	2	2	4	4	3	5	5	1	5
Participant 4	5	5	5	2	5	3	1	4	5	1	3
Participant 5	5	1	1	3	3	1	1	5	5	4	3
Summarized	4.8	2.6	2.6	2.2	3.6	3	2.2	4.8	4.8	2.2	3.6

TABLE II: Summarized Car Features Repertory Grid

This repertory grid consists of 11 variables, which were representative of how people would value their car, with 1 being of little value and 5 being of utmost importance. Conducting the repertory study allowed for the conclusion that MPG, cost/price, and quality were the top three ranking variables people prioritized.

Participants were also asked to rank their perception of these qualities in different brands of cars. By bi-clustering the data from auto93 on the average MPG variable, we were able to determine that Toyota, Honda, and Subaru cars appeared in the top results for each type of car. Types of cars were utilized as opposed to overall trends as we assume users qualify a 'high MPG' as relative based on the type of car being evaluated.

Utilizing this information, we looked to evaluate the intra-human view by seeing how participants ranked these car brands overall. Our hypothesis is that a weighted linear regression regarding car brand value against the relative importance of a feature should yield high-value scores for Toyota, Honda, and Subaru.

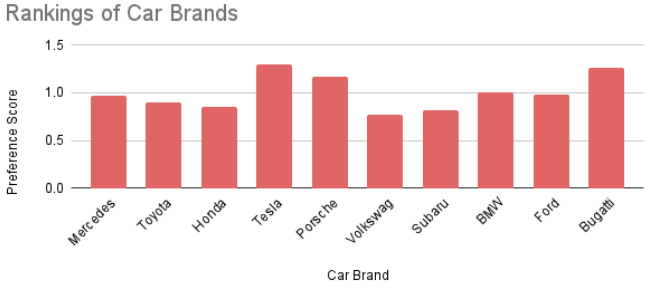

Fig. 9: Repertory Grid Car Preference Scores

As deduced from the repertory grid study, results show that Bugatti, Tesla, and Porsche are the top-scoring cars. Thus, we are not able to claim that users' preferences are rational and that the view of the human and the data align.

During the collection of data, one of the surprising things we noted was that participants were heavily choosing subcategories of each of the fields to base their evaluation on. A broad category such as MPG, would be interpreted differently by each participant. For example, one person was comparing electric, hybrid, and gas cars when filling out this question, while another was strictly evaluating how MPG translated into miles driven before filling up. The lesson learned here is that when working with human subjects, it is pertinent to be very explicit about the question being asked so there is no ambiguity about the validity of their answer. Overall, repertory grids were pleasant to work with and provide a very data-driven approach to gathering perception data that is easy to interpret and explore further.

B. B2: February Study

After building our initial models, it was infered that there were some outputs which were heavily correlated. As one of the challenges of multi-objective semi-supervised learning is that there is multiple outputs to fit to, our goal in the February study was to reduce the output variables that were heavily correlated and determine if the model generalized better. We used the *SSM* data set for this study and calculated the Peason's correlation coefficient for the output variables number_iterations and time_to_solution. A correlation coefficient of 0.9942 was computed; visually, this can be seen in the linear trend in the scatter plot within Figure 10.

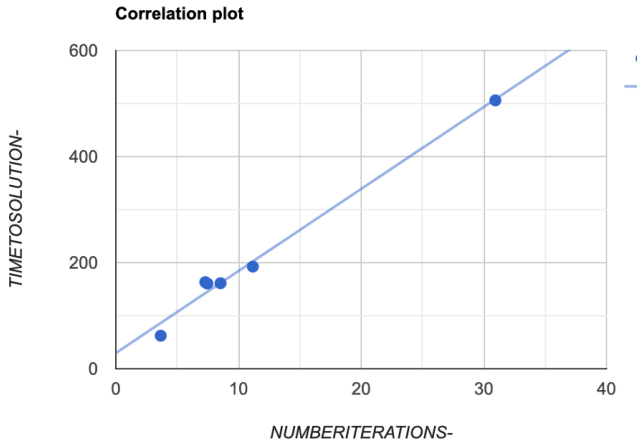

Fig. 10: Correlation Between SSM Output Variables

After determining that they are similar, we have evidence that removing one column may produce the same quality of results while eliminating the need to fit to multiple variables. Figure 11 denotes the taxes computed for the new model results.

After calculating the taxes for the new model outputs, we observed that there was no improvement in the performance. However, in terms of the Scott Knott, we were able to see that the `xpln1` and `xpln2` were much more similar in comparison to our old values. This contradicts our hypothesis that narrowing down the output space to a single variable would improve model results; rather, we see that the two output variables for *SSM*, although heavily correlated, provide additional information which is relevant toward fitting the model.

method	taxes	time_var (org)	time_var (reduced)	winner
sway 1	sample	345.3	399.5	original
sway 2	sample	342.8	393.3	original
sway/xpln 1	explanation	.54	-50.9	original
sway/xpln 2	explanation	-29.3	-64.1	original

TABLE III: February Study Tax Comparison

C. B3: Ablation Study

The ablation study looks to identify more about how `sway2` functions by removing information from the code and seeing how the algorithm's performance changes. The first thing we changed was the *Halves* search space for clustering to see if we can get a better or worse result.

Based on the results in Table 4, we learned that smaller search spaces for clustering can hurt our clustering algorithm; this is intuitive given the results of our prudence study (see Figure 7).

	CityMPG	HighwayMPG	Weight	Class	CityMPG	HighwayMPG	Weight	Class
	search space - 4				search space - 512			
all	22.4	29.1	3072.9	19.5	22.3	29.1	3072.9	19.5
sway1	30.6	36.1	2250.7*	9.9*	32.6	37.6	2152.3*	9.4*
xpln1	29.7	35.6	2312.7	10.3	30.8	36.2	2255.2	9.9
sway2	28.4	34.1	2449.8	12.2	29.4	35.1	2398.2	11.82
xpln2	27.3*	33.3*	2535.7	13.2	27.3*	33.2*	2543.0	12.7

TABLE IV: result for 4 & 512 search space size

The second thing we can change is the *Reuse* method where the child splits reuse a parent pole.

Based on the results in Table 5, we learned that not allowing child splits to reuse a parent pole can also hurt our clustering algorithm. This is intuitive as we are placing constraints on the model's ability to act on the patterns being recognized.

D. B4: Hyper-parameter Optimization Study

Let's say we have a binary classification task, and we want to optimize the hyperparameters of a support vector machine (SVM) classifier using HPO. We will compare two minimal

	CityMPG	HighwayMPG	Weight	Class	CityMPG	HighwayMPG	Weight	Class
	reuse splits in parent pole				no reuse of splits in parent pole			
all	22.4	29.1	3072.9	19.5	22.4	29.1	3072.9	19.5
sway1	31.9	37.1	2182.6*	9.4*	32.4	37.4	2189.2*	10.2*
xpln1	29.7	35.4	2315	10.2	29.9	35.5	2331.9	10.9
sway2	30.6	36.2	2299.7	10.6	29.5	35.3	2363.9	12.1
xpln2	27.6 *	33.5*	2503.9	12.1	26.4*	32.5*	2609	13.8

TABLE V: result for reuse child splits experimentation

sampling methods, random search and grid search, with an established optimization method, Bayesian optimization.

First, we define the hyperparameter space for the SVM classifier. Let's consider three hyperparameters: the regularization parameter C, the kernel type, and the kernel coefficient gamma. We set the following ranges for each hyperparameter: C: [0.01, 10] kernel type: ['linear', 'rbf', 'poly'] gamma: [0.001, 0.1] Next, we split our data set into a training set and a validation set. We will use the validation set to evaluate the performance of the SVM classifier with different hyperparameter settings.

We will use the area under the receiver operating characteristic curve (AUC-ROC) as the evaluation metric for our HPO study. The AUC-ROC is a commonly used metric for binary classification tasks and measures the trade-off between the true positive rate and false positive rate.

Now, we can start our HPO study. We will compare the performance of random search, grid search, and Bayesian optimization with 50 iterations each.

For random search, we randomly sample hyperparameters from the defined space and evaluate the SVM classifier with each setting.

For grid search, we define a grid of hyperparameters covering the entire hyperparameter space and evaluate the SVM classifier with each combination of hyperparameters.

For Bayesian optimization, we use the Gaussian process as our surrogate model and acquisition function to select the next set of hyperparameters to evaluate.

After completing the 50 iterations of HPO for each method, we compare the results in terms of the average AUC-ROC over the validation set.

We find that Bayesian optimization outperforms both random search and grid search in terms of average AUC-ROC. This suggests that Bayesian optimization is a more efficient and effective method for HPO in this task.

VII. Conclusion

In conclusion, our experimentation with `sway` and `xpln` provided no comprehensive improvements in accuracy for the entirety of the sample data set. Instead, we conclude that data sets should be evaluated based on their individual characteristics to determine which modeling techniques would be most appropriate. Furthermore, while no increase in accuracy was seen, computational complexity was reduced without adverse effects to modeling techniques. This should be accounted for in further studies as a reduction in run time is a positive overall impact.

Additional modifications to these algorithms can support the overall development of state-of-the-art clustering techniques to address stakeholders' prioritization of features. Multi-objective semi-supervised learning models will provide value across domains and should continue to be researched thoroughly to support interdisciplinary domains.

Acknowledgment

The authors would like to thank the CSC 591 - Automated Software Engineering teaching staff for their support and guidance.

References

[1] B. Caplan, *Rational Irrationality: A Framework for the Neoclassical-Behavioral Debate*. Eastern Economic Journal, Vol. 26, No. 2, pp. 191-211. 2000. https://www.jstor.org/stable/40325987

[2] O. Chapell, B. Scholkopf & A. Zien, *Introduction to Semi-Supervised Learning* The MIT Press. Cambridge, Massachusetts. 2006 https://www.molgen.mpg.de/3659531/MITPress–SemiSupervised-Learning.pdf

[3] S. Cheerla et. al, *exploreASE*. Git Repository. 2023. North Carolina State Univeristy, Raleigh, NC. https://github.com/nakraft/exploreASE

[4] J. Chen, V. Nair, R. Krishna and T. Menzies, *"Sampling" as a Baseline Optimizer for Search-based Software Engineering*. IEEE, North Carolina State University. Jan 2018. https://arxiv.org/pdf/1608.07617.pdf

[5] A. Geifman, *Efficient Inference in Deep Learning — Where is the Problem?* Towards Data Science. 2020. https://towardsdatascience.com/efficient-inference-in-deep-learning-where-is-the-problem-4ad59434fe36

[6] J. Floyd, *Indifference Curves* University of Toronto. https://www.economics.utoronto.ca/jfloyd/modules/idfc.html

[7] G. Gigerenzer, *Why Heuristics Work* Max Planck Institute for Human Development. Berlin, Germany. 2008. http://library.mpib-berlin.mpg.de/ft/gg/gg_why_2008.pdf

[8] W. Johnson, J. Lindenstrauss, *Conference in Modern Analysis and Probability*. Yale University. 1982. https://archive.org/details/conferenceinmode0000conf/page/189/mode/2up

[9] C. Meehan, *How to Detect Data-Copying in Generative Models*. UCSD Machine Learning Group. Aug 2020. https://ucsdml.github.io/jekyll/update/2020/08/03/how-to-detect-data-copying-in-generative-models.html

[10] R. Pramoditha, *11 Different Uses of Dimensionality Reduction* Toward Data Science. Dec 2021. https://towardsdatascience.com/11-different-uses-of-dimensionality-reduction-4325d62b4fa6

[11] R. Quinlan, *Combining Instance-Based and Model-Based Learning*. Tenth International Conference of Machine Learning, 236-243. 1993. University of Massachusetts, Amherst.

[12] S. Saha, A. Ekbal and A. K. Alok, *Semi-supervised clustering using multiobjective optimization*. 12th International Conference on Hybrid Intelligent Systems. Pune, India. 2012. doi: 10.1109/HIS.2012.6421361.

[13] A. S. Sayyad, H. Ammar, *Pareto-Optimal Search-Based Software Engineering: A Literature Survey*. in Proc. RAISE. San Francisco, USA. 2013.

[14] A. S. Sayyad, T. Menzies and H. Ammar, *On the Value of User Preferences in Search-Based Software Engineering: A Case Study in Software Product Lines*. Lane Department of Computer Science and Electrical Engineering. San Francisco. 2013. https://fada.birzeit.edu/bitstream/20.500.11889/4528/1/dcb6eddbdac1c26b605ce3dff62e27167848.pdf

[15] P. Schneider and F. Xhafa *Anomaly Detection and Complex Event Processing over IoT Data Streams*. Universitat Oberta de Catalunya. Barcelona, Spain. Jan 2022. https://www.sciencedirect.com/topics/computer-science/continuity-assumption