CSC591 Project Report

Hrushabhsingh Chouhan
Email: hychouha@ncsu.edu

Vasu Agrawal
Email: vagrawa5@ncsu.edu

Apoorva Chauhan
Email: achauha5@ncsu.edu

***Abstract*—"Explanation Systems" is one of the many applications of Machine Learning. Our "multi objective semi supervised explanation system" tackles two problems: first, extracting relevant information from a multi-variate dataset, and second, applying the grammar created on a related dataset for the correct selection of qualified candidates.**

In our study, we have explored different ways to implement sway, which is a optimizer and xpln, which is an explanation algorithm. Finally we implemented sway2, an improved version of sway which works on the idea of running sway on sub-regions of a decision space and again on the consolidated result of the previous operation. Similarly, we have implemented xpln2 such that it uses the concept of fast and frugal decision tree and provides an easy-to-visualize and a comprehensible model using a Decision Tree classifier.

Finally we have used statistical methods such as bootstrap, cliffs-delta and Scott-Knott to perform significance testing, effect size testing and performance-based ranking of algorithms respectively.

I. Introduction

Search Based Software Engineering, or SBSE is a technique for solving problems that present trade-offs between goals that are competing with each other. SBSE method requires a model, which generates a variable output depending on the input and an optimizer which finds such inputs that lead to an optimal output. An algorithm that optimizes a given problem does not necessarily optimize another. Hence there is a need to have a baseline method which gives a basic level of performance which can be further built upon through various optimization techniques. The baseline model used by us is sway and xpln[1].

SWAY assumes that the optimal value lies in a small subset of the decision space[3]. Therefore it used certain metrics to prune the sub-optimal decision space. In this case, it halved the decision space by comparing candidates from each half and proceeding with the one having better optimal values than the other. However, this assumption might fail and the optimal value for a model may lie in the region that was discarded. Explanation algorithms provide a way to do just that, by offering a high level overview of the interactions between various features. The "xpln" algorithm is a ante-hoc, semi-local explanation algorithm[2]. It uses the concept of discretization and then forms a rule by using the top ranges that are useful. The baseline xpln also uses pruning. These steps to create a rule makes this model complex and since only the best ranges are selected for the rule generation, the final rule might not be very accurate.

Hence, we propose sway2, an improvement to sway, which divides the decision spaces into different regions and applies SWAY on each region. It then collects the best results from each region and applies SWAY on it to get the optimal result. To solve the issue with the baseline xpln, we used the concept of fast and frugal trees[2].

sway2 acts on sub-regions of a decision space. After the best and the rest clusters are identified using sway2, the ranges separating best and rest are identified using "bins" (Discretization). In this study, we propose xpln2, which incorporates a Decision Tree classifier to classify the best and rest rows. We tried to implement a Fast and Frugal Tree[2], however, the module we were trying to use had issues which led us to make use of a Decision Tree[6]. In its essence, an FFT is a binary decision tree with a limited depth hence a similar approach was taken using decision trees. This approach is a simpler and easier to interpret and performs at par or better computationally than "xpln". Further details will be discussed later in this paper.

A. Structure of this paper

Our main research question was - How to decrease the sampling and explanation tax of our model. As mentioned earlier, the fact that sway discards half of the decision space in every iteration and xpln tends to be computationally expensive and complex lead us to suggest the aforementioned improvements. Also, the need to increase the accuracy of our explanation model led us to use a Decision Tree classifier as our explanation algorithm.

Here are a few caveats of the project : We did not perform hyper-parameter tuning and are currently using fixed hyper parameter values due to the fact that most of our effort was concentrated on the decision tree implementation of xpln2 as well as the sway2 implementation.We were unable to partition the decision space in sway2 using domain-specific knowledge for each dataset because we did not possess the necessary expertise.We also tried using the "fasttrees" package[6] in order to implement FFT for our xpln2 algorithm but were unable to do so due to unresolved issues in the package. Therefore, we decided to go ahead with a Decision Tree implementation with a max depth parameter, thus mimicking an FFT.

II. Related Work

A. The Base Approach

To understand the concept of designing an improved version of the Explanation System, we explored the ways of

selection and the fit criterion as genetic algorithms. In our case, we have used Sway as the optimizer algorithm, which explores a wide decision space using just O(logN) complexity and uses the Zilter's continuous domination predicate to select the better half. Zilter's predicate is typically useful in case where we have many goals(N¿=4) and Boolean domination fails to provide a distinguishable difference.
Further, understanding the whole concept behind the genetic algorithm, we also understood the importance of correctly creating mutants. If a set of selected test cases can significantly reduce the test execution time, but cannot detect most of the mutants, then such a selection is a bad choice[9].To get the best result our focus has been on reducing the sampling tax (the less we look, the more we miss important stuff)

B. Why that is not good enough

One issue with using the SWAY algorithm to find the best cluster in the data set is the increased sampling tax that arises due to the fact that it tends to favor the dominant half of the dataset based on the Zilter's predicate, leaving the less dominant half completely out of consideration. This can lead to biased clustering and potentially skewed results.

C. The second generation SWAY

Due to the aforementioned reasons, we implemented a second generation of SWAY, which is SWAY2. Sway2 divides the configurations into several groups, and then applies SWAY in each group [3]. Then the SWAY optimizer gives us the final optimal configuration for the given data. This results in better selection of the best data-set.

D. Related work while implementing Sway2

Our model is semi-supervised as while building the optimizer, we provide the labelled data and while generating the rule, we forward the unlabelled data. The concepts of dimensionality reduction and random projections are helpful since data sets can be multivariate in nature and can include sets of X values ranging from x_1 to x_n and a set of optimization goals from y_1, y_2,.. y_n. Dimensionality reduction helps us create a linear model for better segregation of data into 2 parts.

For each row, we first calculate their distance from the point A using the Euclidean distance, and then find their distance from A by projecting them on the line joining the points A and B using the equation (a**2 + c**2 - b**2)/(2*c). Sway2 is based on the same concept. Sway provides us with a biased outcome that greatly favors the best of each iteration. Sway 2 addresses this problem by using the best data from several groups before executing the final optimization and providing superior best and rest data to the explanation system.

E. Why is the baseline xpln not good enough

The baseline explanation does several operations to provide a rule. It performs discretization, by converting a set of continuous qualities into discrete ones by assigning categorical values to intervals and thereby converting quantitative data into qualitative data and then usages a useful function to determine rule. It also performs pruning if the rule for an attribute hasn't changed. The baseline explain is complicated and difficult to understand for a non technical person. We also beleive that explain cab be improved to give more accurate results.

F. The second Generation Explain

The second generation explanation, also known as xpln2, performs discretization using the combined data (rest + best). Then we used the concept of fast and frugal tree as proposed by Gigerenzer[8]. Please refer the expln2 explanation in the method section for more details.

G. Related work while implementing xpln2

The process of implementing explain 2 includes understanding the concept of formation of bins, creation of ranges which are later used for Rule formation and selection. A set of ranges is considered useful if it crosses a set threshold and if it's better than the set already selected. Finally we get a rule that generates the results which has the highest number of best rows and the least of rest.

After going through the basic concepts of Explain, we decided to test the Fast-and-frugal trees (FFTs), which are simple algorithms that facilitate efficient and accurate decisions based on limited information[4]. FFT's could be used for multi-goal reasoning, but since FFT is a relatively newer and less explored concept it's implementation was tricky and the reference[5] we found was not compatible with the python version we were using, therefore we first implemented decision tree classifier and later generated our classifier using random forest on the best range returned after discretization. Although, random forest could have lead to better results, the entire concept of explanation algorithm is to make to model more comprehensible which random forest doesn't do. Random forest is typically a lot more complex and difficult to understand as compared to decision tree and thus we at the end stuck to decision tree implementation.
We however did not alter the dominance algorithm- Zilter's continuous domination predicate and the way dimentionality reduction works in the algorithm. The selection of hyper-parameters like that of the confidence interval (0.05), rest (for selecting rest data)(4), seems correct and any changes did not make major difference to our explanation. The basic concept of how sway, the optimizer works(i.e selecting a set of cases that finds the best selection of best and rest data-set) is left as it is. Sway2 however works with slight modification. We also did not alter the ways in which bins are created and merged. Discretization, we felt was relevant. Other methods like bootstrap and the cliff-delta are used as it is from previous submissions. We used the current implementation of sway and

xpln as our state of the art implementation and used them to compare our new sway2 and xpln2.

III. Methods

A. Algorithms

Software Engineering researchers often use search-based software engineering optimization techniques to solve SE configuration problems with single or multiple objectives. In this study, we have implemented four such search based optimization techniques and have made use of the concept of over-sampling-and-pruning(OSAP). OSAP performs the oversampling and prunes the configurations, without evaluating all candidates. In this way, we are able to shorten the decision space significantly with the help of just a few comparisons.

1) SWAY: SWAY is based on a simple assumption that the best configuration for a software engineering problem only appears in a small region of the decision space. Using this assumption, the SWAY Algorithm further develops a separator (the half function) that bi-clusters the configurations. It prunes one cluster amongst them based on the evaluations of two representatives, one in each cluster. This pruning process is performed recursively until a small region is found.

To separate the decision space, we make use of the half function. The half function basically finds a random row A in the given data set and then gets another distant row B. We then find the distance of all the other rows from A and sort them based on their distance from A. Finally we add the first half of the rows to the set "left" and the second half to the set "right" and return these two sections back to the calling function.

To evaluate the two cluster representatives, we make use of the Zitzler's continuous domination predicate. Continuous domination judges the domination status of a pair of individuals by running a "what-if" query which checks the situation when we jump from one individual to another, and back again. Specifically:

- For the forward jump, we compute $s_1 = -\sum_i e^{w_i(a_i - b_i)/n}$
- For the backward jump, we compute $s_2 = -\sum_i e^{w_i(b_i - a_i)/n}$

where a_i and b_i are the values on the same index from two individuals, n is the number of goals and w_i is the weight which is -1 if we are minimizing the goal or 1 if maximizing. According to Zitzler, one example is preferred over the another if we lost the least upon jumping to it; i.e. $s_1 < s_2$.

2) SWAY2: SWAY2 is an improved version of SWAY. It first requires us to divide the decision space. For example, in the requirement engineering problems, such division can be the size/scale of the proposed configuration, measured by the number of features to be developed, etc. In our case, we have fixed the division to 4 subsets. We chose this value based on the trial and error approach. After dividing the decision space, the SWAY optimizer algorithm is executed on each sub-decision space. The optimal/desired configuration is the union set of best rows coming from each sub decision space, on top of which we then execute another round of SWAY giving us the final optimal configuration in the given data set. Our understanding behind this approach was inspired by the work of CHEN JIANFENG[3]. In his work he has mentioned that dividing the decision space into multiple halves based on expert domain knowledge could lead to more informative clusters. Implementing the subdivision deduced by the expert domain knowledge would be one of our future works.

3) XPLN: Explanation algorithms offer a high-level picture of how model features influence each other. As part of the current baseline explanation algorithm, we, first of all, discretize our attribute values X and then find the ranges/bins that distinguish the best cluster from some random sample of the rest. We then explore those ranges further looking for rules that select the best cluster[2]. This rule generator does not evaluate any extra y values and the generated bins are developed such that we are able to find some constraint that selects for lots of best and not much of the rest.

The entire idea behind the implementation of Explain algorithm is that we are trying to find the rule that would help us get the best possible set of rows within a given data set. We initially start with the best and rest clusters received via the sway optimizer. On these clusters, the bin function is executed which gives us a list of bins/ranges of attribute values X. Typically in the case of categorical data, the ranges are defined based on the different categories that we have while in the case of numeric data, the bins are first created and later merged together based on how diverse the bins are from each other.

Once we have the final ranges coming in from the bin function, we then sort these ranges based on how well they select for best values using "probability x support" parameter; i.e. b^2/(b+r). After that, we try various combinations of these ranges such as first range, the first 2 ranges, the first 3 ranges and so on to come up with rules that would give us the highest number of best and least possible number of rest.

4) XPLN2: As part of our improvement to the current explain algorithm, we propose to use the Fast and Frugal Tree(FFT) approach as approved by Gigerenzer and implemented by Phillips et al[8]. An FFT is a binary decision tree of limited depth. At each level, there is one sub-tree and one leaf node that selects for either one of the two classes. Hence, there are two choices at each level and at bottom level there are two leaves, one for each class. We followed the steps provided by Chen et al[9] in their research paper's Table 4 for generating FFT for a multi-objective class problem. We used Decision Tree instead of FFT tree since there wasn't

an available running library for generating FFT and when compared to the Random Forest Classifier, it performed better and provided a greater level of understandability. We initially performed the discretization operation on the X attributes and later order the ranges to get the best possible rule as done in the previous approach. Once we do that, we then divide the data based on the best range/rule that we got. We then applied our Decision Tree Classifier that generated training data based on the X attributes of the best and rest rows. We also encoded the categorical data using the LabelEncoder. Our classifier was based on whether the row came under the best or the rest cluster represented by 0 and 1 respectively. We trained our model using the aforementioned training data and then applied this to all the rows present in the data thus generating our final best rows data set.

B. Data

Talking about the data, as part of this project, we worked on a total of 11 datasets each having its unique properties with regards to the number of goals to be optimized, the number of rows and the information pertaining to each of these datasets. Lets start by exploring each of them briefly:-

1) auto93.csv: The auto93.csv dataset has a total of 398 rows along with 8 columns out of which there are 3 optimization goals namely Lbs, Acc and Mpg. This dataset comprises of details of different cars with information regarding the number of cylinders, origin, model, etc and we try to find the optimal cars amongst them that have the highest mileage, lowest weight and highest acceleration.

2) auto2.csv: The auto2.csv is an advanced version of auto93.csv with a much greater dimensionality and an increased number of objective goals. The auto2.csv dataset has a total of just 93 rows along with 23 columns out of which there are 4 optimization goals namely Weight, Class, HighwayMPG and CityMPG. The dataset is again composed of details of different cars with information regarding the number of cylinders, maker, RPM, Horsepower etc and we try to find the optimal cars amongst them that have the highest city and highway mileage, lowest weight and lowest class.

3) china.csv: The china.csv dataset has a total of 499 rows along with 19 columns and just a single optimization goal of reducing the Effort. The dataset contains details about the software effort estimation details where we have attributes such as resources, dev type, input, etc and we need to reduce the development effort as much as possible.

4) coc1000.csv: The coc1000.csv dataset has a total of 1000 rows along with 25 columns consisting of 5 optimization goals namely LOC, AEXP, PLEX, RISK and EFFORT. The dataset again contains details with regard to the software effort estimation where we have attributes such as resource used, team, time, tool etc and we need to reduce the development efforts, risk, etc.

5) coc10000.csv: The coc10000.csv dataset has a total of 10000 rows along with 25 columns consisting of 5 optimization goals namely LOC, AEXP, PLEX, RISK and EFFORT. The dataset contains details about the software effort estimation and is basically an incremented version of coc1000.csv dataset with greater number of rows for processing.

6) nasa93dem.csv: The nasa93dem.csv dataset has a total of 93 rows along with 29 columns consisting of 4 optimization goals namely Kloc, Effort, Defects and Months. The dataset contains details with regard to the software effort and detect estimation where we have attributes such as resource used, team, time, tool etc and we need to reduce the development efforts, defects, development months, etc.

7) healthCloseIsses12mths0001-hard.csv: This dataset has a total of 10000 rows along with 8 columns consisting of 3 optimization goals namely MRE, ACC and PRED40. The dataset contains details with regard to the issue close time where we have attributes such as min sample leaves, max depth etc and we need to reduce mag. of relative error, maximize accuracy and PRED40.

8) healthCloseIsses12mths0011-easy.csv: This dataset has a total of 10000 rows along with 8 columns consisting of 3 optimization goals namely MRE, ACC and PRED40. The dataset contains details with regard to the issue close time and is similar to the previous dataset with comparatively easier values.

9) pom.csv: This dataset has a total of 10000 rows along with 13 columns consisting of 3 optimization goals namely Cost, Completion and Idle. The dataset contains details about an agile project management estimator where we have attributes such as Team Size, Plan, Criticality etc and we need to reduce the cost and idle time and maximize the completion of the change.

10) SSM.csv: This dataset has a total of 239360 rows along with 15 columns consisting of 2 optimization goals namely NUMBERITERATIONS and TIMETOSOLUTION. The dataset contains details with regard to computational physics where we are trying to find the fastest configurations using flash and have attributes such as alpha, beta, PRESMOOTHING etc and we need to reduce the number of iterations and time to solution.

11) SSN.csv: This dataset has a total of 53662 rows along with 19 columns consisting of 2 optimization goals namely PSNR and Energy. The dataset contains details about computational physics where we are trying to find the fastest configurations using flash and have attributes such as Bbias, Threads, Bframes etc and we need to reduce the energy and PSNR.

C. Performance Measures

To rank the datasets, we made use of Zitzlter's predicate. Zitzlter's predicate is basically used in order to judge the domination status of pair of individuals by running a "what-if" query which checks the situation when we jump from one individual to another, and back again. Zitzlter's predicate is typically used in the case of optimization algorithms to check which half of the dataset is better and then iterate in that particular half.

For all four of our algorithms—Sway, Sway2, Xpln, and Xpln1—we started off with a budget of 5 (let's call it B1). Budget offers a simple, efficient approach to swiftly sample a big area. After running our models for B1 iterations, we raised our budget to 20 (B2) iterations. The output of our initial budget B1 (which are first hypotheses) and the actual performance over B2 are later compared. For each of the datasets in our table, this method was repeated. Changing the B2 did not, significantly alter our predictions from the B1.

As part of the project, we ran our algorithms on each of the 11 datasets for a total of 20 iterations and identified the top B (best) items within that dataset. We used Scott-knott to rank our algorithms and compared them based on the sample tax, explanation tax and explanation variance. Further, we checked our results based on the data coming from our sway, sway2, xpln and xpln2 algorithms and compared them with that of all and top.

D. Summarization Methods

As part of our summarization methods, we tried to compare the results obtained by various algorithms and ordered them based on the lower sum of B(best) values. With regards to our statistical data, we made use of cliffs delta and Bootstrap and checked whether the results that we obtained from our algorithms were significant and also whether the difference between them is significant.

We further made Scott-Knott plots to rank these algorithms for each of the Objective values y. Under the Scott-Knott plot, the algorithm first identifies the range of the entire data that needs to be plotted. We then plot individual values of each of the RX rows that we generate and send the tiles function. The tiles function plots these values on the horizontal x-axis with the values being printed based on their rank coming in from the Scott-Knott function. The tables generated by Scott-Knott for each objective of the aforementioned datasets is present in the "out" folder of our GitHub repository[11]

IV. Results

A. RQ1:Does Sway2 perform better than the current state-of-the-art?

As per the tables mentioned below, sway2 in terms of performance has invariably beaten the state-of-the-art sway implementation in the majority of the datasets. This is something that we were expecting as well since sway2 is an advanced version of sway wherein we strategically divided the dataset into several different halves and later performed sway on them individually thus getting the best cluster in each of the halves.

B. RQ2:Why does sway2 perform better than sway?

Sway2 is an improved version of sway. It first requires expert knowledge to divide the decision space. For example, in requirement engineering problems, such division can be the size/scale of the proposed configuration, measured by the number of features to be developed, etc. After dividing the decision space from expert knowledge, the SWAY is executed in each sub-decision space. The optimal/desired configuration is the union set in each sub-decision space.

C. RQ4: Why is XPLN2 better as compared to XPLN?

The XPLN2 algorithm uses decision trees as part of it's explanation algorithm which makes it inherently intuitive making them easier to understand for non-technical stakeholders. Further, explanation algorithm using decision trees are transparent in nature, making it easier to identify biases or errors in the decision-making process.

Trees can be scaled easily to handle large datasets and complex decision-making processes. As a result, the explanation algorithm can be applied to larger datasets and more complex systems, providing a better understanding of the decision-making process. Finally, decision trees can be generated quickly and efficiently, making the explanation algorithm faster and more responsive. This can be clearly interpreted from the results that we have attained. Our XPLN2 is performing significantly better than XPLN in almost all the tables when comparing their results with all and amongst each other for various objective values.

D. RQ5: Are there any limitation to using FFTs as part of the explaination algorithm?

There are certain limitations to using FFTs as part of an explanation algorithm. Decision trees may not be suitable for capturing complex interactions between different features, which could limit the accuracy of the explanation. Additionally, FFTs are prone to overfitting, which could result in inaccurate explanations. In some cases, more complex decision-making algorithms, such as neural networks or support vector machines, may be able to provide better accuracy or more nuanced decision-making.

Algorithms	CityMPG+	HighwayMPG+	Weight-	Class-
all	24.64	32.01	3380.19	21.45
sway	26.26	31.93	2679.82	14.05
xpln	24.055	30.535	2887.6	16.87
sway2	25.08	32.245	3179.94	20.255
xpln2	24.79	32.63	3323.12	20.955
top	24.805	30.83	2789.25	14.815

TABLE I: auto2.csv results

Algorithms	CityMPG+	HighwayMPG+	Weight-	Class-
all to all	=	=	=	=
all to sway	≠	≠	≠	≠
all to sway2	≠	≠	≠	≠
sway to sway2	≠	≠	≠	≠
sway to xpln	≠	≠	≠	≠
sway2 to xpln2	≠	≠	≠	≠
sway to top	≠	≠	≠	≠

TABLE II: auto2.csv comparison

Algorithms	Lbs-	Acc+	Mpg+
all	3267.44	17.16	26.18
sway	2834.59	16.13	19.965
xpln	2743.62	15.82	25.42
sway2	3664.46	17.415	19.825
xpln2	3546.12	17.12	22.815
top	2861.09	15.875	19.96

TABLE III: auto93.csv results

Algorithms	Lbs-	Acc+	Mpg+
all to all	=	=	=
all to sway	≠	≠	=
all to sway2	≠	≠	=
sway to sway2	≠	≠	≠
sway to xpln	=	≠	=
sway2 to xpln2	≠	≠	≠
sway to top	=	=	=

TABLE IV: auto93.csv comparison

Algorithms	N_effort-
all	5133.12
sway	965.93
xpln	1911.04
sway2	2057.71
xpln2	4088.94
top	1021.61

TABLE V: china.csv results

Algorithms	N_effort-
all to all	=
all to sway	≠
all to sway2	≠
sway to sway2	≠
sway to xpln	≠
sway2 to xpln2	≠
sway to top	=

TABLE VI: china.csv comparison

Algorithms	LOC+	AEXP-	PLEX-	RISK-	EFFORT-
all	1266.25	3.75	3.75	8.375	38509.4
sway	970.385	2.935	2.825	5.29	26848.4
xpln	998.77	2.94	3.055	6.68	30564.9
sway2	1204.48	3.605	3.7	7.8	36224
xpln2	1261.51	3.725	3.8	8.49	37631.5
top	945.905	2.99	2.9	5.485	26861.9

TABLE VII: coc1000.csv results

Algorithms	LOC+	AEXP-	PLEX-	RISK-	EFFORT-
all to all	=	=	=	=	=
all to sway	≠	≠	≠	≠	≠
all to sway2	≠	≠	≠	≠	≠
sway to sway2	≠	≠	≠	≠	≠
sway to xpln	≠	≠	≠	≠	≠
sway2 to xpln2	≠	≠	≠	≠	≠
sway to top	=	=	=	=	=

TABLE VIII: coc1000.csv comparison

Algorithms	Loc+	Risk-	Effort-
all	1362.15	8.91	41183.6
sway	965.95	5.16	23640.3
xpln	1121.84	6.07	30920
sway2	1393.26	7.035	32633.7
xpln2	1360.6	9.055	39822.9
top	954.415	5.22	23712.8

TABLE IX: coc10000.csv results

Algorithms	Loc+	Risk-	Effort-
all to all	=	=	=
all to sway	≠	≠	≠
all to sway2	≠	≠	≠
sway to sway2	=	≠	≠
sway to xpln	≠	≠	≠
sway2 to xpln2	≠	≠	≠
sway to top	=	=	=

TABLE X: coc10000.csv comparison

Algorithms	MRE-	ACC+	PRED40+
all	82.3	5.2	22.1
sway	73.965	7.655	19.435
xpln	73.715	7.655	18.78
sway2	78.185	6.34	20.34
xpln2	83.835	4.74	22.675
top	74.055	7.59	19.335

TABLE XI: hc1.csv results

Algorithms	MRE-	ACC+	PRED40+
all to all	=	=	=
all to sway	=	=	≠
all to sway2	≠	≠	≠
sway to sway2	≠	=	≠
sway to xpln	≠	≠	≠
sway2 to xpln2	≠	≠	≠
sway to top	=	=	=

TABLE XII: hc1.csv comparison

Algorithms	MRE-	ACC+	PRED40+
all	92.3	-8.5	17.8
sway	38.945	-1.54	52.85
xpln	42.265	-1.4	49.8
sway2	40	-1.85	52.435
xpln2	90.155	-8.26	19.24
top	39.455	-1.565	52.465

TABLE XIII: hc2.csv results

Algorithms	MRE-	ACC+	PRED40+
all to all	=	=	=
all to sway	≠	≠	≠
all to sway2	≠	≠	≠
sway to sway2	≠	≠	=
sway to xpln	≠	≠	≠
sway2 to xpln2	=	=	=
sway to top	=	=	=

TABLE XIV: hc2.csv comparison

Algorithms	Kloc+	Effort-	Defects-	Months-
all	94	624.4	3761.8	24.2
sway	57.585	259.96	2026.86	18.02
xpln	60.895	248.82	2059.29	18.025
sway2	24.035	115.78	858.885	13.98
xpln2	67.695	353.24	2444.88	19.33
top	66.69	300.705	2344.91	19.495

TABLE XV: nasa93dem.csv results

Algorithms	Kloc+	Effort-	Defects-	Months-
all to all	=	=	=	=
all to sway	≠	≠	≠	≠
all to sway2	≠	≠	≠	≠
sway to sway2	≠	≠	≠	≠
sway to xpln	≠	≠	≠	≠
sway2 to xpln2	≠	≠	≠	≠
sway to top	≠	≠	≠	≠

TABLE XVI: nasa93dem.csv comparison

Algorithms	Cost-	Completion+	Idle-
all	888	2.16	0.48
sway	309.6	0.895	0.22
xpln	337.76	0.89	0.205
sway2	702.4	2.115	0.555
xpln2	865.825	2.155	0.565
top	311.385	0.89	0.22

TABLE XVII: pom.csv results

Algorithms	Cost-	Completion+	Idle-
all to all	=	=	=
all to sway	≠	≠	≠
all to sway2	≠	≠	≠
sway to sway2	≠	=	=
sway to xpln	≠	≠	≠
sway2 to xpln2	≠	≠	≠
sway to top	=	=	=

TABLE XVIII: pom.csv comparison

Algorithms	NUMBERITERATIONS-	TIMETOSOLUTION-
all	30.9	171.36
sway	9.305	128.59
xpln	13.31	102.92
sway2	6.505	148.262
xpln2	23.885	111.73
top	9.325	65.91

TABLE XIX: SSM.csv results

Algorithms	NUMBERITERATIONS-	TIMETOSOLUTION-
all to all	=	=
all to sway	≠	≠
all to sway2	≠	≠
sway to sway2	≠	≠
sway to xpln	≠	≠
sway2 to xpln2	≠	≠
sway to top	≠	≠

TABLE XX: SSM.csv comparison

Algorithms	PSNR-	Energy-
all	44.5	1658
sway	44.52	1533.56
xpln	44.045	1358.95
sway2	44.915	1190.24
xpln2	44.47	1491.94
top	44.565	1538.9

TABLE XXI: SSN.csv results

Algorithms	PSNR-	Energy-
all to all	=	=
all to sway	≠	≠
all to sway2	≠	=
sway to sway2	≠	≠
sway to xpln	≠	≠
sway2 to xpln2	≠	≠
sway to top	=	=

TABLE XXII: SSN.csv comparison

V. Threats to validity

A. Sampling Bias

This paper shares the same sampling bias problem as every other data mining paper.Sampling bias threatens any classification experiment; what matters in one case may or may not hold in another case.For example, even though we use 11 open-source datasets in this study which come from several sources,they were all supplied by individuals[9].

B. Evaluation Bias

The paper uses various performance measures such as Zilter's predicate, the Probability x Support equation, etc to evaluate the clusters. However, there are several other quality measures that are present in the software engineering domain that can be used to evaluate the performance and effectiveness of the clusters.A comprehensive analysis using these measures maybe performed with our replication package. Additionally, other measures can easily be added to extend this replication package.

VI. Discussion

Fast and Frugal Trees were the model we first intended to use to develop the second generation explanation, but due to time restrictions, we chose another model that was compatible with the Python version we were using. We talked about the usage of both random forests and decision trees. We talked about the ideal amount of divisions to make during implementation of sway. Our discussions also involve the use of libraries like word-net and glove to get word embed-dings to include symbolic values for explanation. In the baseline method, there is no adequate metrics utilized to calculate distance between symbolic values(rather than just having 0's and 1's). We discovered a number of threats when implementing the Sway2 and Explain2 systems as discussed below.

1) By removing the less dominating cluster, Sway chooses the best cluster, however it's possible that the best cluster

isn't the best outcome.The other half might contain the optimal cluster as well.
2) Although the tree-based technique might be more accurate but is also more complex, we attempted to construct explain using Random Forest and decision trees. Our explanation could become complicated if the data set is extremely complex.
3) The numbers 1 and 0 are used to categorize the data set's symbolic values. This is not the optimal technique to project symbolic values. As a result, the rule may be biased toward numerical data and fail to take symbolic data into account when reaching a judgment.

VII. Conclusion

This study provides an alternative to create a better version of the baseline multi objective semi-supervised explanation model using the second generation Sway[3] and Explain[5].

The baseline sway selects one best cluster after recursively bi-clustering in-order to search the best rows. Continuing on with the best and rest clusters that we generated, we pass these clusters to the explain method which then generates a rule to select the best data in the entire data set by performing discretization and selection.

Sway2 builds upon the success of its predecessor, sway, by applying the sway algorithm on sub-regions of a data-set and the consolidated result of the above operation. This enables the algorithm to identify better clusters within the data that might have otherwise been missed. On the other hand, Xpln2 incorporates discretization and selection while also utilizing a Decision Tree classifier to create a more visually appealing and comprehensible model. The results of these studies are detailed in the methods and results section that provide insights into the technicalities and practical implementation of these models.

Bootstrap and Cliffs Delta were used to check whether the results obtained via various algorithms were different and their difference was significant enough. Furthermore, Scott-Knott was used to rank these algorithms for each of the objective values. We have used the initial budget of 5 and later increased the budget to 20, to understand if increasing the budget creates difference in the results.

VIII. Future Work

SWAY, assumed that a small region in decision space can lead to best objectives. Therefore, it prunes the decision space by half after eliminating the less dominant regions[3]. Sway2 is an improved version of Sway1 wherein the decision space is first divided in certain (say x) segments, the SWAY is then executed in each sub segment. The optimal results from all these segments is combined and then final execution happens. This process runs well but there exists some scope that can be further improved.

1) A related future effort will examine how many decision spaces the data set should be partitioned into. This can depend on the size and complexity of the data.
2) Future works will explore more methods for producing mutants. for example, if the mutant production can utilize information from existing data rather than just oversampling, consider it.
3) Determine if there exists a relationship between the number of decision spaces and the hyper-parameters.

We have explored a simpler way to model an explanation system. Our model requires the Sklearn.tree to implement DecisionTreeClassifier[10]. Our initial approach was to run FFT on the data-set. There are several ways the existing way of implementing explain2 can be improved such as :-

1) Finding out what needs to be done to execute FFT's instead of Random forest and decision tree.
2) Once the implementation of fast and frugal tree is done. Use the FFT algorithms on various data-set and compare the results with respect to baseline model and also the random forest based model.
3) Figure out what hyper-parameters can be used and manipulated while using a tree based classifier
4) When the (tree-based) model is finished, it might be overly precise. Pruning these models might be necessary.
5) The depth parameter exists in tree based algorithm, but it is not examined in this work. Determining how the depth impacts the final Rule is one of the most crucial upcoming tasks.
6) For symbolic data, we have employed label encoding. Researching alternative encoding methods to label order encoding is another component of the future work.
7) This study demonstrates how Sway and Explain, as well as Sway2 and Explain2, function together. No other combination other the one mentioned has been used. Our advice is to test whether sway1 and explain2 function better together than with explain1 and sway2.

References

[1] https://arxiv.org/pdf/1608.07617
[2] https://github.com/timm/tested/blob/main/docs/onExplain.md
[3] https://shorturl.at/mvEFY
[4] http://papers.cumincad.org/data/works/att/7e68.content.pdf
[5] doi:10.1017/S1930297500006239
[6] https://github.com/dominiczy/fasttrees
[7] https://www.kdnuggets.com/2022/02/random-forest-decision-tree-key-differences.html: :text=Random
[8] doi:10.1017/S1930297500006239
[9] doi.org/10.1145/3236024.3236050
[10] https://scikit-learn.org/stable/modules/tree.html
[11] https://github.com/apoorvacha/ASE_Project