Final Report

Michigan State University

Tanishq Tanmay,
Pavan Yachamaneni,
Yena Hong,
Shuangyu Zhao

SIEMENS

MICHIGAN STATE UNIVERSITY

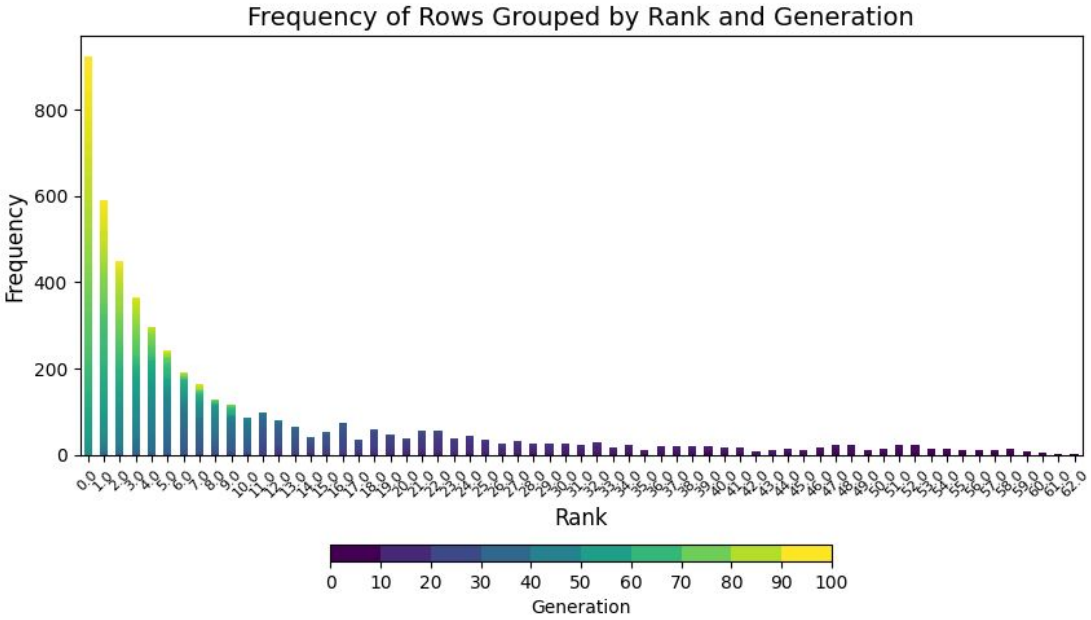
Exploratory Data Analysis

Overview of Dataset

	Unnamed: 0	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	f1	f2	gen	Rank
0	0	0.417022	0.720324	0.000114	0.302333	0.146756	0.092339	0.186260	0.345561	0.396767	0.538817	0.417022	2.482199	1	53.0
1	1	0.019367	0.678836	0.211628	0.265547	0.491573	0.053363	0.574118	0.146729	0.589306	0.699758	0.019367	4.408805	1	52.0
2	2	0.019880	0.026211	0.028306	0.246211	0.860028	0.538831	0.552822	0.842031	0.124173	0.279184	0.019880	4.198771	1	39.0
3	3	0.556240	0.136455	0.059918	0.121343	0.044552	0.107494	0.225709	0.712989	0.559717	0.012556	0.556240	1.693099	1	62.0
4	4	0.000402	0.976759	0.376580	0.973784	0.604716	0.828846	0.574712	0.628076	0.285576	0.586833	0.000402	6.783459	1	47.0

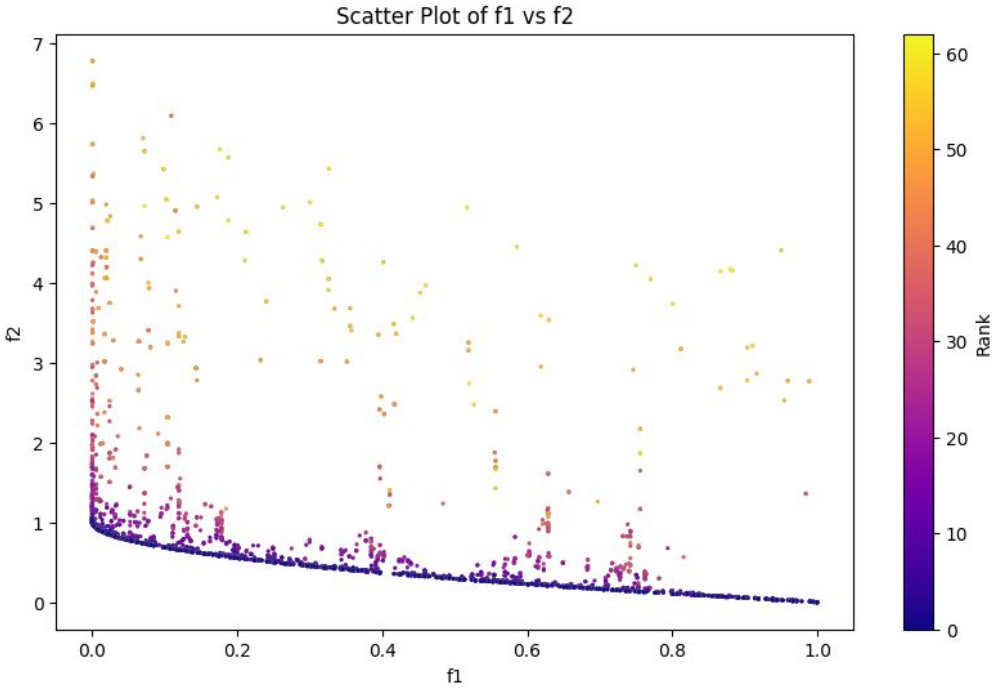
- The dataset contains 10 x variables (x1 - x10) and 2 objective functions (f1, f2) derived from these 10 x variables based on ZDT1.
- The 'gen' column represents the generation of data, ranging from 1 to 100. 'Gen 1' signifies the first generation, while 'gen 100' represents the hundredth generation.
- The 'Rank' column represents the rank of the data as defined by the Pareto rule, starting from the 0th rank.
- Each generation generates 100 data rows, thus the dataset comprises 100 * 100 = 10,000 rows.

Frequency by Rank and Generation



- As the generation of data creation increases, the rank tends to decrease (i.e., it gets closer to 0).
- In the graph, the initial generations (purple) are mostly distributed after the 20th rank, while the later generations (yellow) are distributed from the 0th to the 9th rank.

Scatter Plot of f1 vs f2



- The graph represents a scatter plot for two objective functions, f1 and f2.
- The prominent purple dots at the forefront of the graph represent high-ranked (rank 0) data, which constitute the desired Pareto set.
- However, distinguishing the Pareto set, indicated by these purple dots, from nearby lower-ranked data is challenging due to their close proximity.

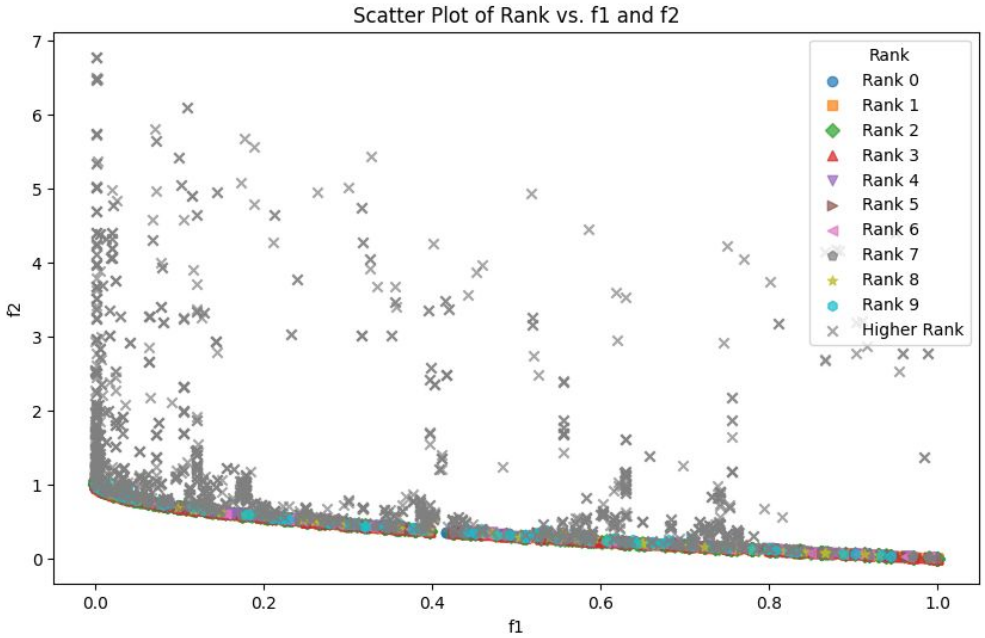
Data Preprocessing

CORE IDEA

This data preprocessing designed to classify the 'y' variable for subsequent work in predicting the Pareto set, involves three categories

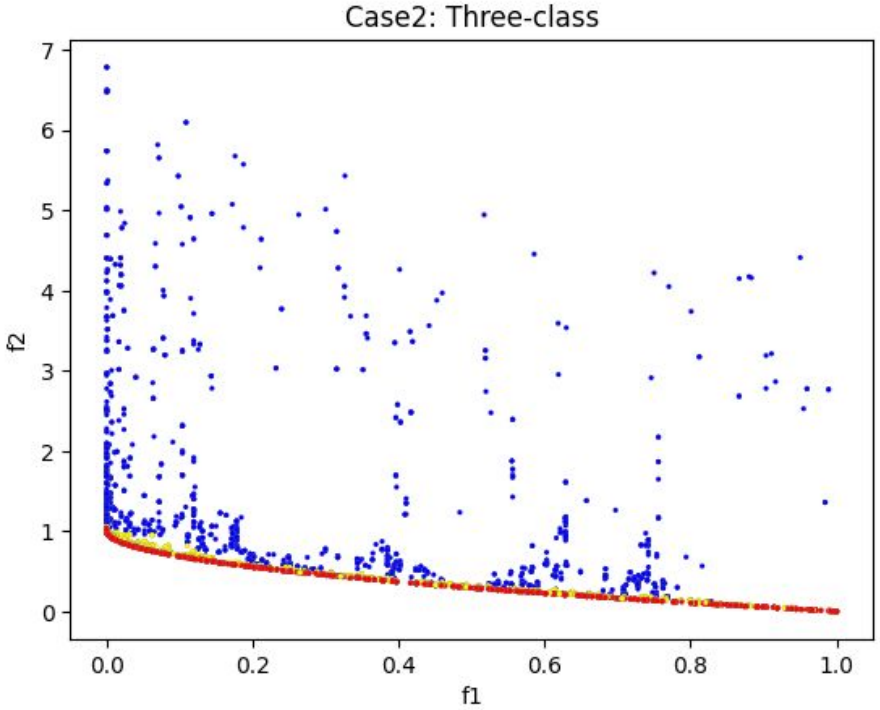
- Multi-class with ten y classes (ranks 0-9),
- Three-class with three y classes (lower, middle, high ranks from 0-62), and
- Binary class with two y classes (lower and high ranks from 0-62).

1. Multi-class with ten y classes (ranks 0-9)



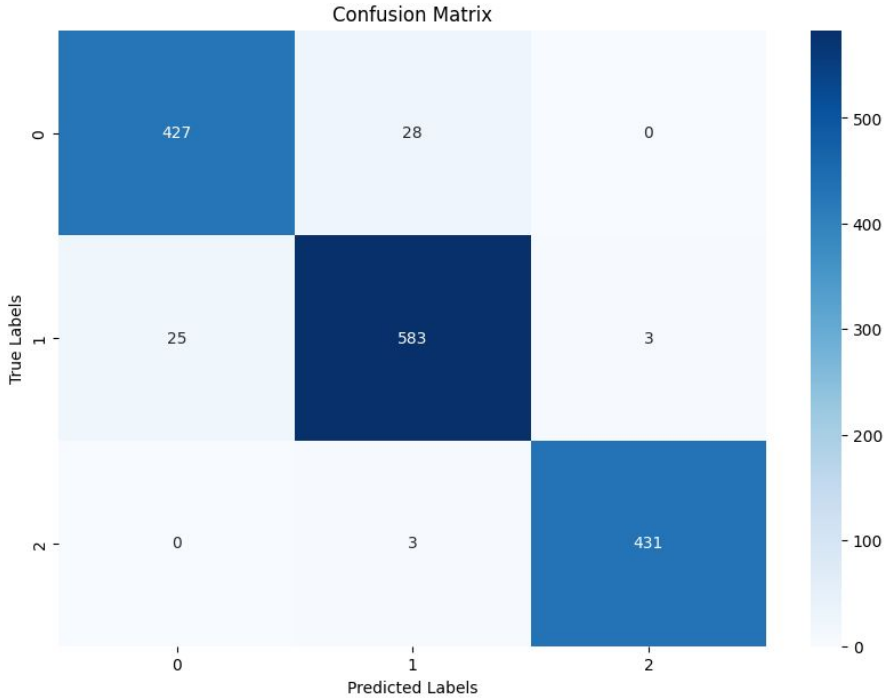
- Our initial approach was to classify the y variable into ranks from 0 to 9 and exclude ranks above 10 (rank 10 and higher) from the analysis.
- However, as shown in the graph, the data within the lower ranks (0-9) are very closely clustered, making it a challenge to develop an algorithm to predict each rank.
- Additionally, excluding data of rank 10 and above significantly reduces the amount of data available for use, making it difficult to build an accurate model.

2. Three-class with three y classes (ranks 0-62)



- We have adopted a method of dividing the y variable into three categories: low rank, medium rank, and high rank.
- This classification method encompasses the entire range of our data, including all ranks from 0 to 62.
- Such an approach helps prevent accuracy degradation that can occur due to insufficient data.
- The rank criteria defining each category can be adjusted through 'margin_1' and 'margin_2' values.
- According to the current settings, low rank is defined as 0 and 1, medium rank as 2 to 10, and high rank as 11 and above, with the number of data points in each category being 1515, 2038, and 1447 respectively, which is a well-balanced distribution.
- In the presented graph, each category is visually distinguished by color: red for low rank, yellow for medium rank, and blue for high rank.

2. Three-class with three y classes (ranks 0-62)



Apply Decision Tree Algorithm

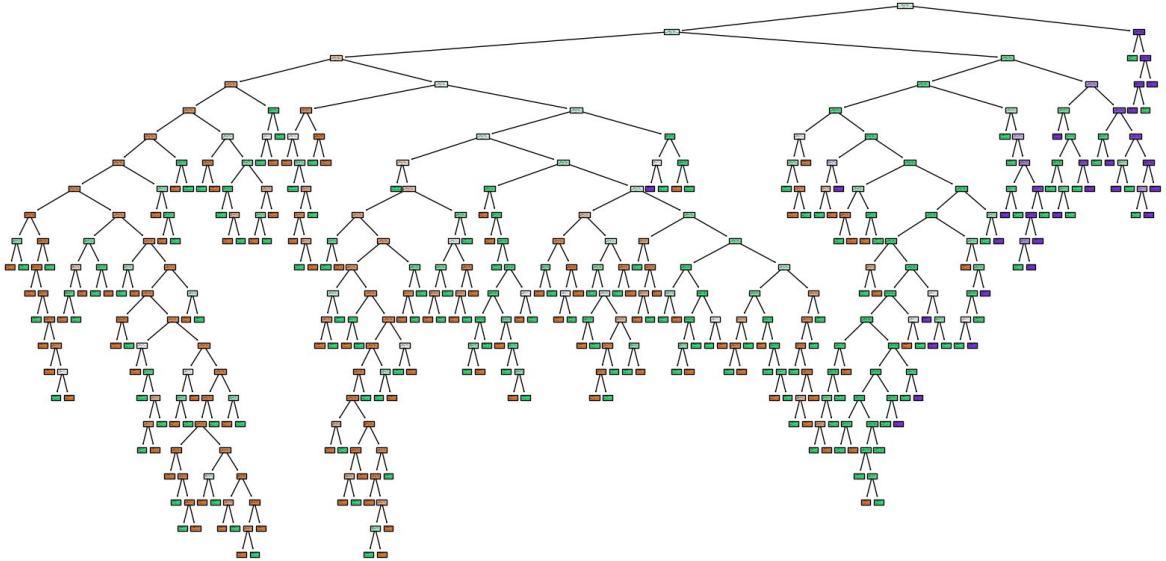
Based on the previously defined y variable, we implemented a decision tree algorithm using independent variables from x1 to x10.

The implementation process involved four main steps:

1. Creation of training and testing datasets,
2. Application of the decision tree classifier,
3. Testing of the algorithm, and
4. Analysis of accuracy.

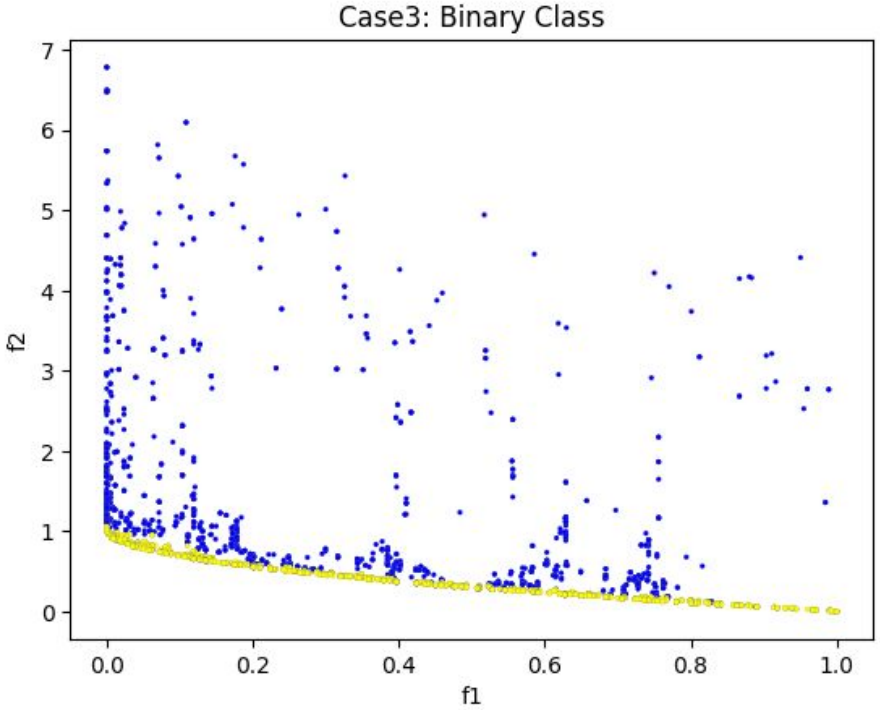
- In creating the training and testing datasets, we ensured balanced sampling across each y group.
- As indicated by the confusion matrix, the application of the decision tree resulted in high accuracy rate, recall rate, and F1 score.
- However, there were about 20 instances of misclassification observed in the lower and middle ranks.

2. Three-class with three y classes (ranks 0-62)



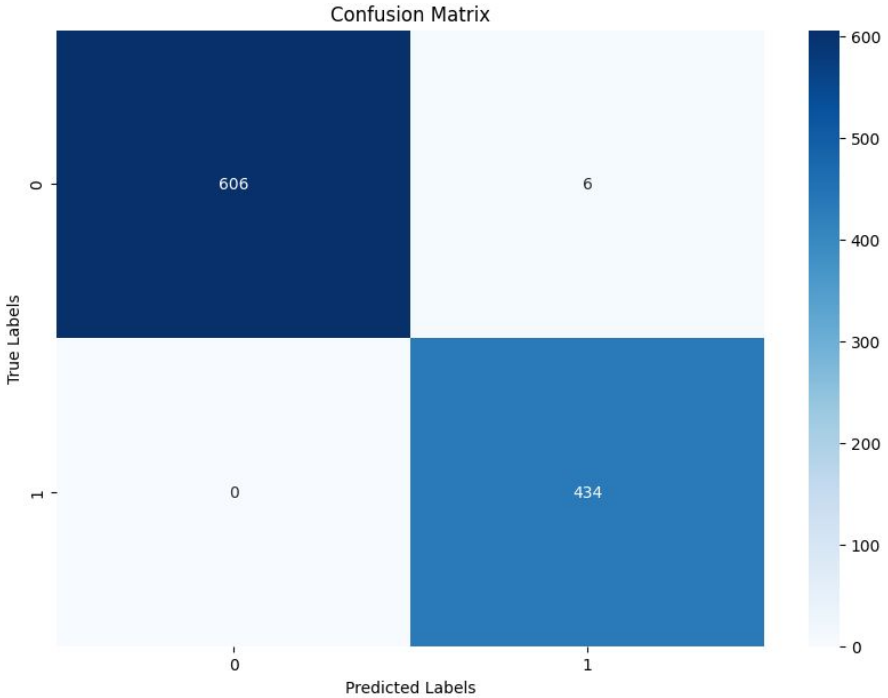
- The graph above displays a visualized decision tree algorithm.
- Due to the tree's deep depth and numerous nodes, interpreting the algorithm from the graph is quite challenging.
- In an attempt to create a more interpretable graph, we experimented with limiting the depth of the tree.
- However, this adjustment significantly reduced the accuracy of the algorithm.

3.1. Binary class with two y classes (ranks 0-62)



- Our next method involves creating binary classes for the y variable.
- As seen in the graph, the yellow dots represent the low rank, while the blue dots indicate the high rank.
- Similarly, the boundary between the groups can be adjusted through the margin.

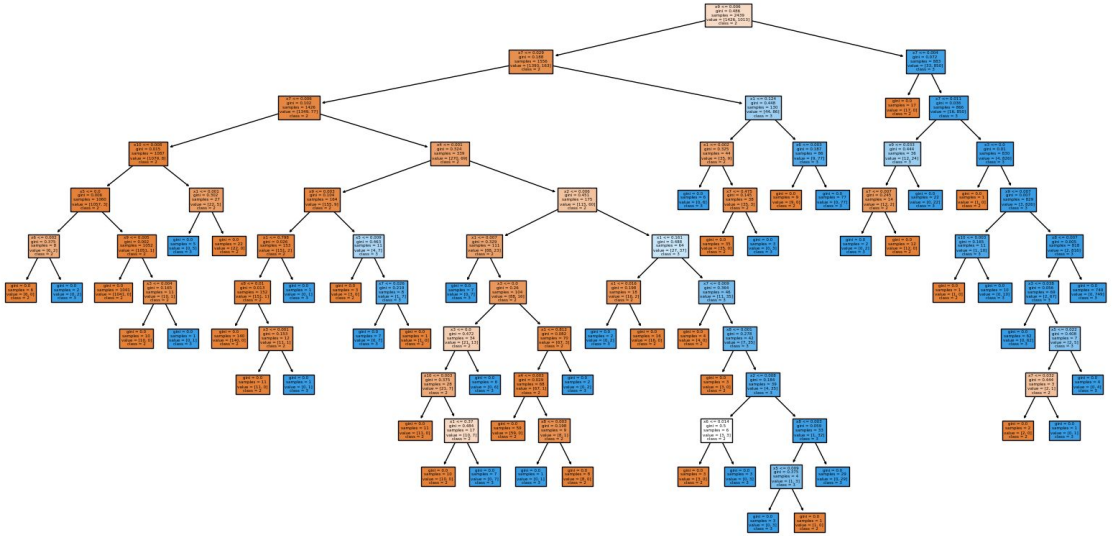
3.1. Binary class with two y classes (ranks 0-62)



Apply Decision Tree Algorithm

- After applying the decision tree algorithm to the binary classes, we analyzed its accuracy and found the results to be excellent.
- Apart from six data points, all others were accurately classified in testing phase.

3.1. Binary class with two y classes (ranks 0-62)



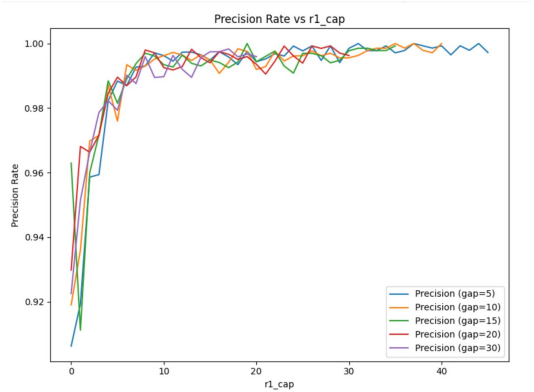
- Compared to when we divided y into three classes, the decision tree plot appears easier to interpret when y is categorized into binary classes.

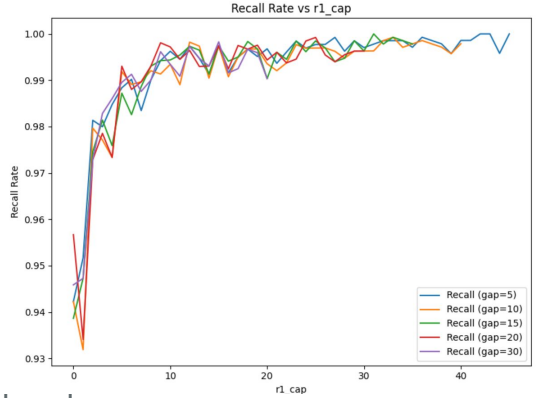
3.2. Adjust the margin for Binary Classes

```
# Define the gap list
gap_list = [5, 10, 15, 20, 30]
eval_results = {gap: [] for gap in gap_list}

# Loop over each gap value
for gap in gap_list:
    r1_min = df['Rank'].min()
    r1_max = 50 - gap

    # Loop over r1_cap
    for r1_cap in range(r1_min, r1_max + 1):
        # Create v3_class
        df['v3_class'] = np.where(df['Rank'] <= r1_cap, 1, np.where(df['Rank'] <= r1_cap + gap, 2, np.nan))

        # Drop rows with nan in v3_class
        df_clean = df.dropna(subset=['v3_class'])

        # Prepare the feature variables (X) and the target variable (y)
        X = df_clean.loc[:, 'x1':'x10']
        y = df_clean['v3_class']

        # Split the dataset
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, stratify=y, random_state=42)

        # Initialize and train the Decision Tree Classifier
        classifier = DecisionTreeClassifier(random_state=42)
        classifier.fit(X_train, y_train)

        # Predictions and evaluations
        y_pred = classifier.predict(X_test)
        precision = precision_score(y_test, y_pred)
        recall = recall_score(y_test, y_pred)
        accuracy = accuracy_score(y_test, y_pred)

        # Store the results
        eval_results[gap].append((r1_cap, precision, recall, accuracy))
```

- We have confirmed that the decision tree works very well on the binary classes.
- We will observe the accuracy of the algorithm while adjusting the margin between the two y groups and determine the optimal margin.

3.3. Results for Adjusting the Margin





- The graph displays precision, recall, and accuracy.
- The x-axis, labeled 'r1_cap', represents the upper limit of the low rank group.
    - e.g., if 'r1_cap' is 0, the low rank group consists only of data with rank 0. If 'r1_cap' is 1, it includes data with ranks 0 and 1, and if 'r1_cap' is 2, it includes data with ranks 0, 1, and 2.
- Considering our total of 62 ranks, 'r1_cap' is defined up to 50.
- Each graph contains five lines representing five different scenarios based on the 'gap'. The 'gap' refers to the margin between the low and high rank groups.
    - e.g., if 'r1_cap' is 3, the low rank group includes data with ranks 0 to 3. If the 'gap' is 5, the high rank group starts from rank 8 (3 + 5 gap), and if the 'gap' is 10, it starts from rank 13 (3 + 10 gap).
- We set the 'gap' values to 5, 10, 15, 20, and 30 to observe how the decision tree performs across the entire range of ranks.
- When examining the graphs for precision, recall, and accuracy, we notice that the values stabilize when the upper limit of the low rank group surpasses around 10.

Summary

- At this phase, we can draw the following conclusion: the predictive accuracy of the algorithm increases when the y variable is categorized into binary classes.
- The effectiveness of different 'gap' values varies depending on whether we focus on precision, recall, or accuracy.
- In the next phase, we will explore various algorithms beyond the decision tree, adjusting the gap between the two y groups (both with and without the gap approach).
- This will help us in finding a model that accurately predicts the pareto and non-pareto sets.

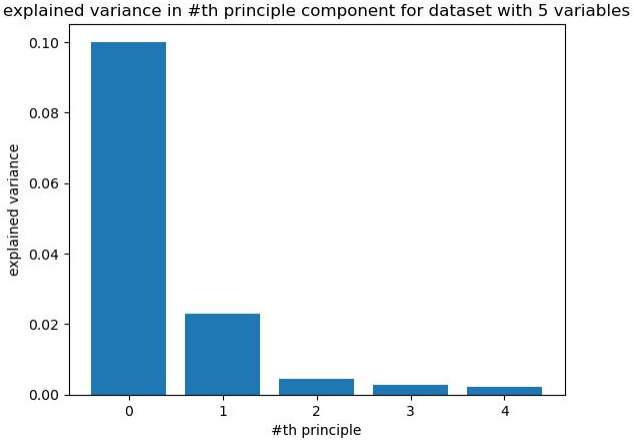
MODELS without GAP

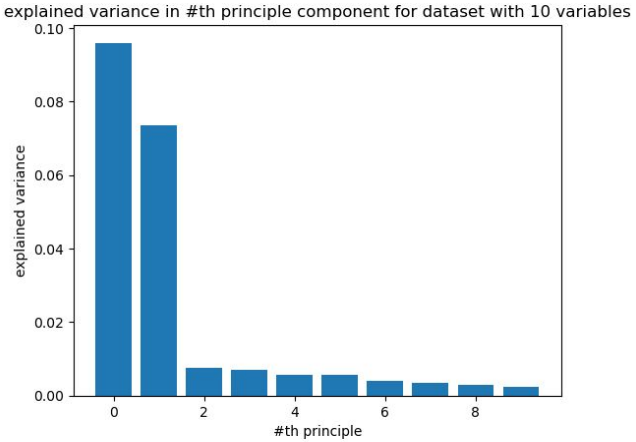
CORE IDEA

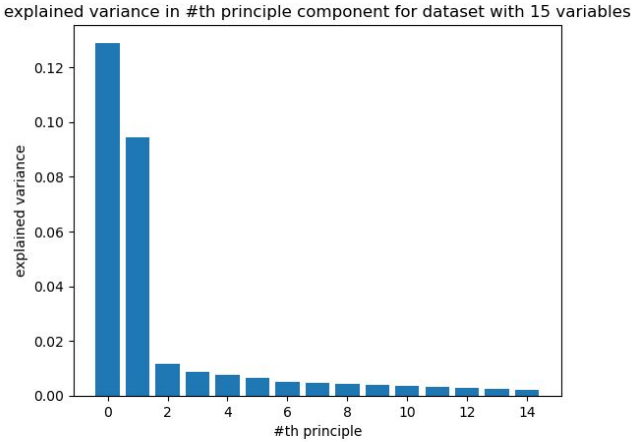
- Principal Component Analysis
- Loop - find the most optimal boundary
- F1 score - assess models
- Supervised models

Principal Component Analysis

- PCA works as feature extraction to
- In all three datasets, the first two principal component can explain most of the variance.
    - ALL datasets in this problem can reduce dimension to two.







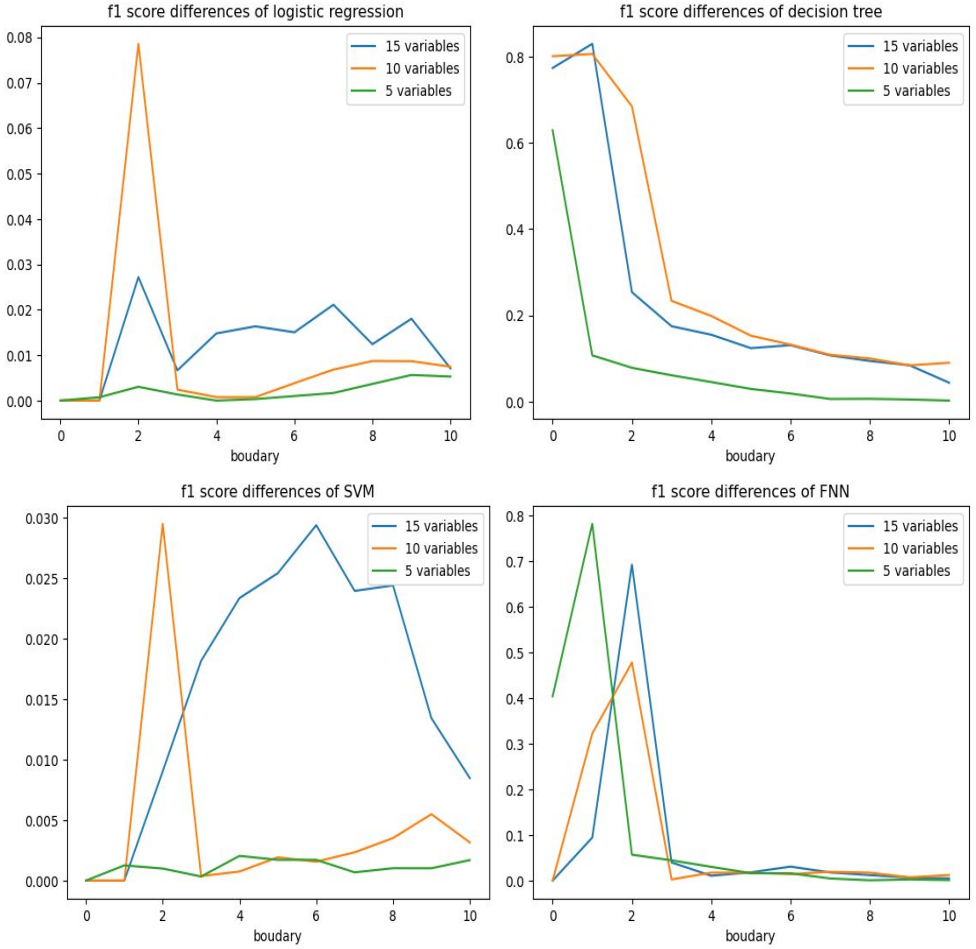
Supervised models

Compare performance differences between using pca and no using pca

F1 score differences are absolute values.

- When boundary rank >=3, the models performances' differences are pretty low.
- the original dataset dimension is lower, the difference is lower.

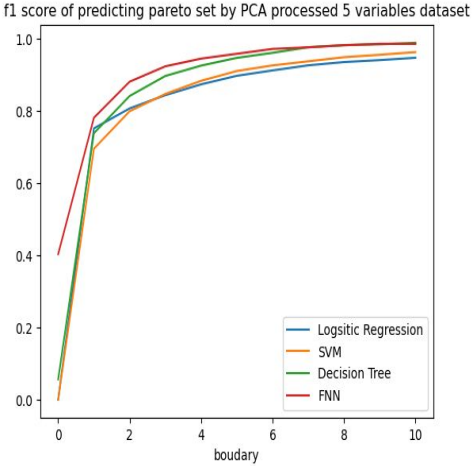
In general, when we set rank>=3 as the boundary, we definitely can use pca preprocessed dataset to train the models and compare the performances, which can help to find the generalized rules for attributes in this problem.

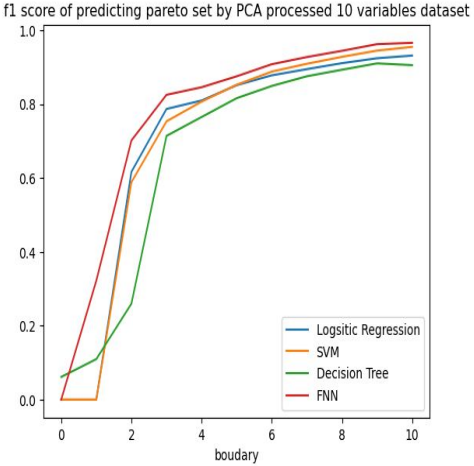


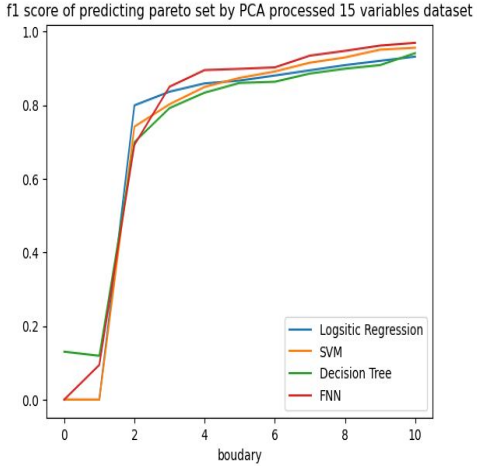
Supervised models

In these three images, the datasets used to train models were all reduced dimensions to two.

- Finding the most suitable model in this problem.
  - The differences are not significant, when predicting near-pareto set in different original dataset.
  - But in general, FNN model is relatively more ideal than others.
- Rank=3 is the optimal boundary to separating near-pareto set from others.



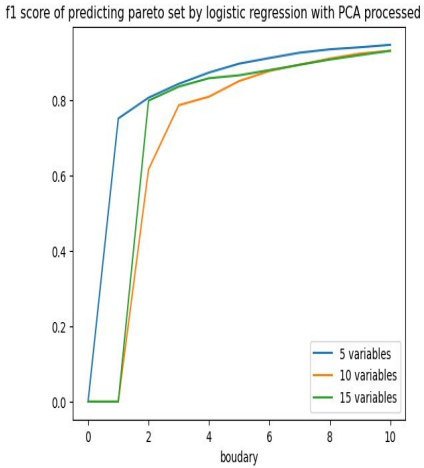


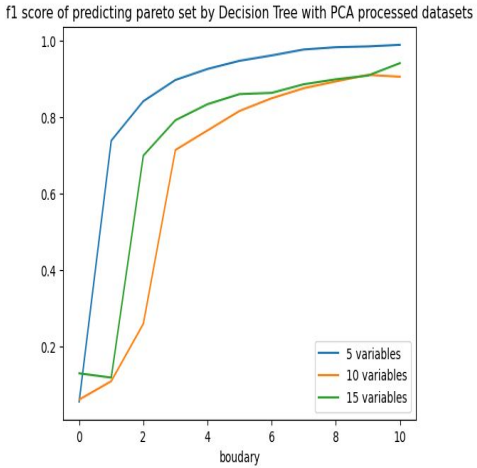


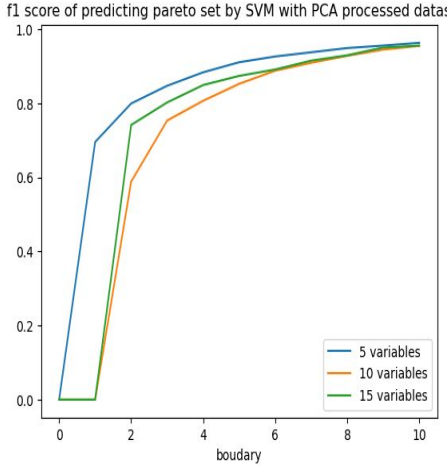
Supervised models

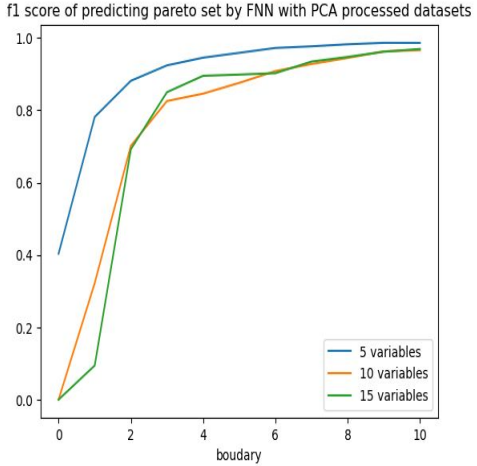
In these four images, the datasets used to train models were all reduced dimensions to two.

- The number of attributes is lower, the predicting performance is better.
- When the boundary's rank<3, the performance differences in different number attributes are high. When the rank=3 is set to be the boundary, the performance differences are low enough.









Summary

- All datasets in this problem can be reduced dimensions to two.
- The optimal boundary is the rank=3
  - The influence of various attributes number towards predicting performance is low enough.
- The models' differences in this problem are not significant, but FNN is more ideal than others generally.
- In the following project to predict attributes.
  - while training models, the boundary is better to be rank=3. Because, it can mitigate the influence of original attribute's number.
  - The model to predict 2 principal components is recommended. After predicting 2 principal components.
  - In the further study, PCA deserved to be deeply researched. The general inverse function of reducing dimension could be developed, predicting attributes based on predicted principle components.

GENERAL MODELS without GAP

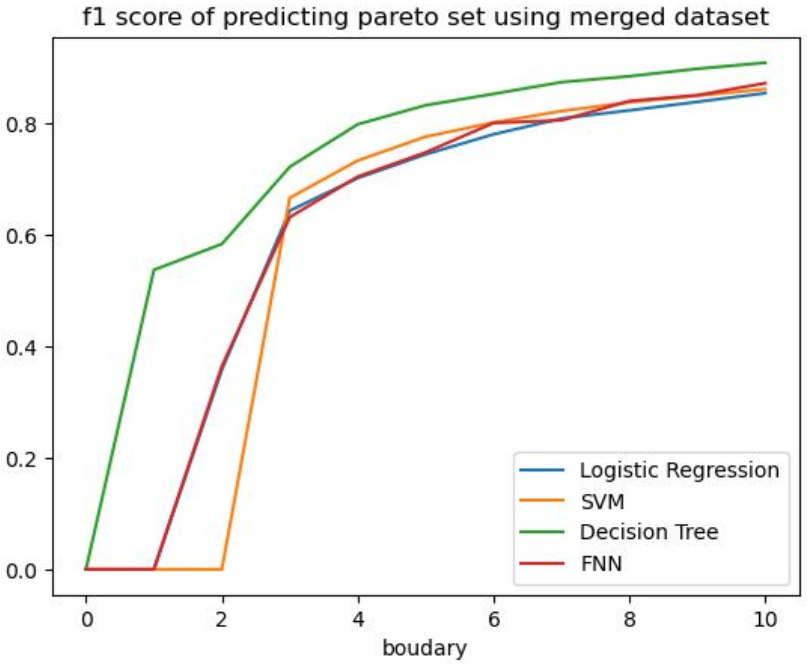
CORE IDEA

- Reducing all datasets' dimensions to ***two***
- Union the preprocessed datasets and train the generalized models

Result Analysis

Thanks to the dimension reduction by PCA, general models can be developed which can generalized the rules adapting all datasets in this problem.

- When boundary's rank >=3, the models' performances are all ideal enough.
- Decision Tree worked best.



Assumption and Further Study

- It is quite likely that the predictive performance of these models is higher when the dimension of the original dataset is lower.
- The above assumption worth to be researched. Because it will help to estimate the predicting performance of attributes of near-pareto set.

Ancestors Descendants

Idea

1. Machine Learning Integration:
    a. Incorporate machine learning techniques into the rank generation process.
    b. Leverage NSGA-2, a smart system capable of evolutionary pattern recognition.
    c. Explore the impact of data engineering on rule generation for ranks.
2. Data Partitioning:
    a. Categorize data into two distinct groups: ancestors and descendants.
    b. Train the algorithm using data from ancestors.
    c. Test the algorithm across descendants to evaluate its performance.
3. Transfer of Knowledge:
    a. Implement a knowledge transfer mechanism from ancestors to descendants.
    b. Explore how rules generated during training are incorporated into the testing phase.
4. Evolutionary Learning:
    a. Utilize NSGA-2 for its evolutionary learning capabilities.
    b. Investigate how the algorithm adapts and evolves over time with the data.
5. Pattern Recognition:
    a. Study the ability of NSGA-2 to recognize and understand patterns.
    b. Evaluate how well the system adapts to evolving data patterns.

Approach and Results

1. Metadata Specifications
   a. Dataset: ZDT 15 VAR
   b. Generations: 50
   c. Sample Size for Each GENL 50
   d. Ancestors size: 1900
   e. Descendants Size: 600
2. Assumptions:
   a. Rank <= 3 are considered to be Pareto, else design parameters are non-pareto
3. Results
   a. Enhanced Rank Generation:
      i. The integration of machine learning, particularly NSGA-2, has resulted in improved efficiency in rank generation.
      ii. The system demonstrates enhanced capabilities in recognizing patterns through evolutionary learning.
   b. Effective Knowledge Transfer:
      i. Categorizing data into ancestors and descendants proves effective, showcasing the algorithm's ability to transfer knowledge.
      ii. The success of knowledge transfer is reflected in the algorithm's performance, with transferred rules influencing the testing phase positively.

Classification Report for main_data_ans:

	precision	recall	f1-score	support
0	0.94	0.84	0.89	1419
1	0.73	0.89	0.80	696
accuracy			0.86	2115
macro avg	0.84	0.87	0.85	2115
weighted avg	0.87	0.86	0.86	2115

Classification Report for main_data_dsc:

	precision	recall	f1-score	support
0	0.96	0.98	0.97	48
1	1.00	1.00	1.00	552
accuracy			0.99	600
macro avg	0.98	0.99	0.98	600
weighted avg	1.00	0.99	1.00	600

Fig-1: Classification Report describing Training accuracies on ancestors and testing accuracies on descendants data

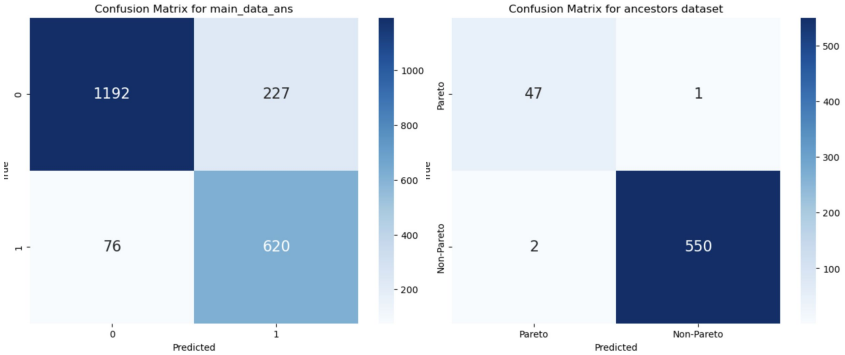


Fig-2, Fig-3: Classification matrix describing Training accuracies on ancestors and testing accuracies on descendants data

Conclusion

Conclusion

Our Project followed the following path

- Problem Understanding
- Data Preprocessing
- Supervised Learning Models
- Approach-1: Without Gap
    - Decision tree works best.
    - PCA can help to train the generalized model, but the variables are more, the accuracies are lower.
- Approach-2: With Gap
    - Creating gap always doesn't ensure to give perfect model but guarantee near perfect classification of pareto solutions.
- In leveraging machine learning, data manipulation can significantly influences rule generation for ranks using an intelligent system uncovering patterns through evolutionary processes.
- Therefore, partitioning data as ancestors, descendants and training model on ancestors data set finally validating on descendants is much like traditional approach but guarantees capturing patterns, rules that makes a large share of pareto solutions.
- For upcoming endeavors, instead of giving design parameters and asking for pareto/Not, we can navigate towards making AI algorithms to create design parameters of pareto solutions that can explore vast design space.

Thanks

SIEMENS

MICHIGAN STATE UNIVERSITY