A Comparative Study of Feature Transformations for Student Programme Classification

Wonder239
ID: 239239

***Abstract*—This report is based on a dataset of final exam results from students in different programmes and different grades. This study aims to identify effective feature transformations for predicting students' programme labels based on exam scores. We perform various feature transformations, including Z-score, PCA, and feature expansion. After that, we use different types of unsupervised learning methods for clustering and supervised learning methods for data classification.**

***Index Terms*—unsupervised learning, supervised learning, feature observation**

I. Introduction

In this report, our goal is to try at least three transformations of the data and obtain at least three sets of features. Then we need to use unsupervised learning and supervised learning methods to predict the student's program based on these three sets of features and analyze which set of features is better than the others.

The dataset we used is *training_set.xlsx*, which contains *grade*, *gender*, *Q1-Q5 scores* and *programme* of the students.

A. Feature Observation

In this section, we use feature expansion, z-score, and principal component analysis (PCA) to transform the features and select three sets of features, then compare them. We also use the t-Distributed Stochastic Neighbor Embedding (t-SNE) to visualize the distribution of features. We compare these three feature sets using intra–inter cluster distance ratio.

B. Unsupervised learning

In this section, we use an unsupervised learning method called Hierarchical Clustering [1] for clustering in three sets of features.

C. Supervised learning

In this section, we use three different supervised learning methods: Decision Tree, K-Neighbors, and NaiveBayes for classification of programmes in three sets of features. We apply an ensembled classifier as well.

II. Task 1 & 2

In these two tasks, we need to observe the distribution of raw features and perform at least 3 transforms for the raw features. After that, the features are compared via at least one metric.

A. Data Processing

Fig. 1 shows the original data distribution of the five questions. We find that there are some extreme values in the boxplot. Therefore, we applied the *replace_outliers_with_median* function to handle these outliers in the numeric data. After cleaning, the total number of datasets was reduced from 466 to 417. Fig. 2 shows the data distribution after cleaning, and extreme values are now replaced by the median.

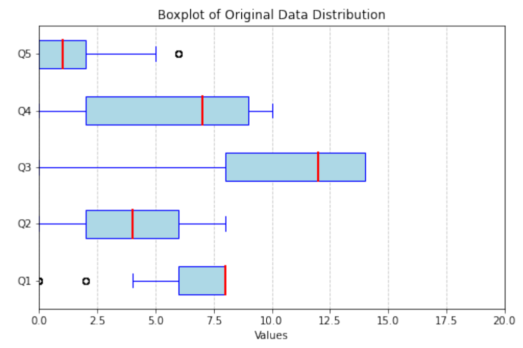


Fig. 1. Original Data Distribution.

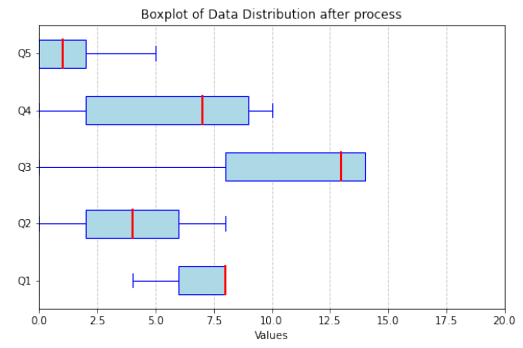


Fig. 2. Data Distribution After Process.

Fig. 3 illustrates the number of samples in each programme.

B. Feature Transformation

*1) **Feature expansion**:* We add a new feature called Total, which represents the sum of the points for questions Q1–Q5. And this is our first set of features.

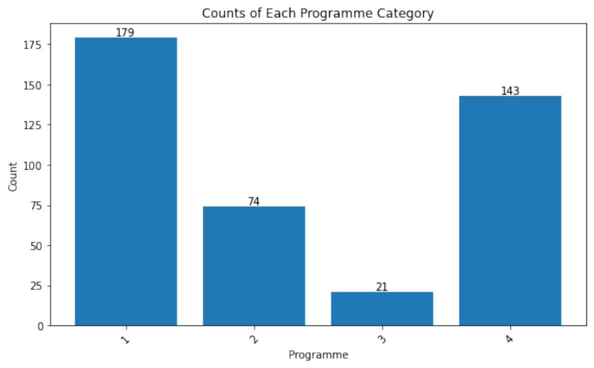


Fig. 3. Counts of Each Programme Category.

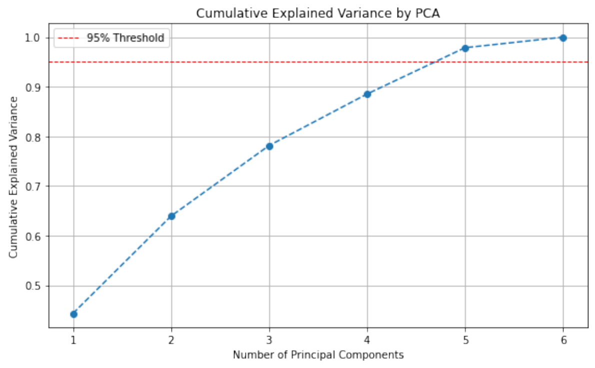


Fig. 4. Cumulative Explained Variance by Principal Components.

*2) **Z-score and Information Gain**:* We standardized all features using the z-score (standard score) method after the feature expansion on set 1. We then use information gain to calculate the mutual information [2] between each feature and the Programme label. We run several seeds to decrease the randomness of the mutual information and use the average value of the results. The results of the mutual information are shown in Table I. Grade and Total show the highest mutual information with Programme. Q2, Q4, Q5, and Gender have moderate mutual information, while Q3 and Q1 show very low mutual information (close to zero), indicating they are nearly independent of Programme. As a result, we choose to remove Q3 and Q1, while all other features are retained for further operations. And this is our second set of features.

TABLE I: Mutual Information between each feature and Programme

Feature	**Mutual Information**
Grade	0.193
Total	0.172
Q2	0.101
Q4	0.093
Q5	0.083
Gender	0.074
Q3	0.033
Q1	0.013

*3) **Dimensionality Reduction: Principal Component Analysis**:* We use PCA to analyze on the Set 2. The explained variance ratio is shown in Fig. 4.

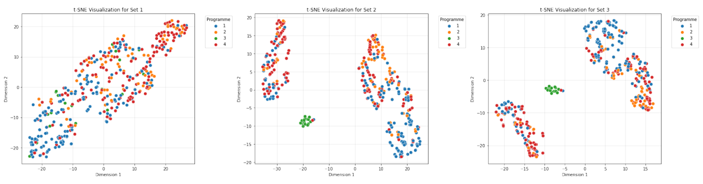
We find that retaining the first five principal components is sufficient because the first five principal components explain 95 % of the variance, so we dropped the remaining dimensions and saved the transformed data as the third set of features.

In conclusion, the three sets of features after feature transformation are:

- Set 1: Adding a new feature: Total, defined as the sum of Q1 through Q5.
- Set 2: Applying z-score normalization to the first set and then removing Q3 and Q1 based on mutual information, retaining Grade, Total, Q2, Q4, Q5, and Gender.
- Set 3: Performing PCA on the second set and keeping the first five principal components.

C. *Comparison of Three Feature Sets*

1) t-SNE Visualization of Different Feature Sets: We use the t-SNE method [3] to visualize the data features in two dimensions.



(a) Set 1 (b) Set 2 (c) Set 3

Fig. 5. Comparison of the three sets visualizations.

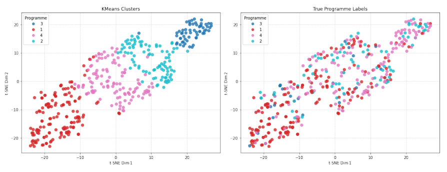
Fig. 5 shows the two-dimensional t-SNE visualizations for each of the three feature sets. We observe that, in the t-SNE plots for Set 2 and Set 3, samples from Programme 3 form a distinct cluster separated from the other programmes. However, Set 1 fails to separate those samples from the rest. This means the transformation methods enhance the feature space's ability to separate Programme 3 from the rest.

2) Intra–Inter Cluster Distance Ratio: In this method, we first assume the number of clusters is four and apply k-means [4] separately to each of the three feature sets to group the samples. We then evaluate and compare them using the ratio of average intra-cluster distance to average inter-cluster distance.

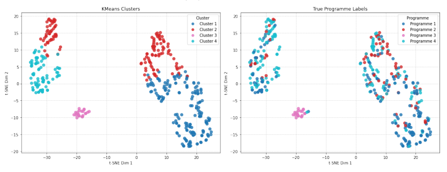
TABLE II: Intra–Inter Cluster Distance Ratios for the Three Feature Sets after k-means Clustering

Set	**Intra–Inter Cluster Distance Ratio**
Set 1	0.40
Set 2	0.51
Set 3	0.49

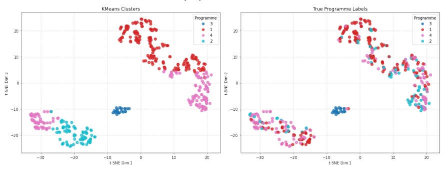
Table II shows that Set 1 achieves the lowest intra–inter cluster distance ratio, indicating the best clustering quality. Sets 2 and 3 have very similar ratios, which are a little higher than Set 1. Fig. 6 presents the 2-dimensional t-SNE projections of the three feature sets. In each figure, the left panel shows the projection coloured by the k-means cluster assignments, while the right panel is coloured by the ground-truth Programme labels.



(a) Set 1



(b) Set 2



(c) Set 3

Fig. 6. t-SNE Visualization of K-Means Clustering on Three Different Feature Sets

D. Conclusion

The comparison shows that three feature transformation methods do work. Set 2 and Set 3 have successfully separated Programme 3 from the rest. However, Set 1 achieves the lowest intra-inter cluster distance ratio, which is contrast to the t-SNE visualization result. This will be further discussed in the DISCUSSIONS AND CONCLUSIONS section.

III. Task 3

In this task, we need to cluster at least three sets of features. For each set of features, we need to try at least three clustering configurations. After that, we need to give the reasons of the experiment design and compare the best performed clustering results with the distribution of programmes.

A. Sets Chosen

In this part of the experiment, we use the same three feature sets obtained from Tasks 1 & 2.

B. Justification of Experiment Design

Since K-Means clustering method has been used in Task 1 & 2 to compare the three sets of features, we use hierarchical clustering in this task. And the number of clusters is four. In this method, we try three clustering configurations:

- Config 1: Ward linkage + Euclidean distance: Minimizes within-cluster sum of squares to produce compact, uniformly spherical clusters.
- Config 2: Complete linkage + Cityblock (Manhattan) distance: Constrains cluster diameter via maximum pairwise distance and uses L1 metric for enhanced robustness to outliers.
- Config 3: Average linkage + Euclidean distance: Trades off single and complete linkage by using average distances to build a smooth and stable hierarchical clustering.

By using these three clustering configurations, we can evaluate the impact of different linkage criteria and distance metrics on clustering performance to identify the data structure best suited to our feature sets.

C. Results & Discussion

Table III shows the results of three sets of features using Hierarchical Clustering with three different clustering configurations.

Set	Model	Linkage	Metric	Ratio	Accuracy	Weighted-F1
Set1	HC	ward	euclidean	0.459	0.556	0.486
Set1	HC	complete	cityblock	0.408	0.537	0.472
Set1	HC	average	euclidean	0.446	0.537	0.472
Set2	HC	ward	euclidean	0.520	0.583	0.510
Set2	HC	complete	cityblock	0.483	0.568	0.512
Set2	HC	average	euclidean	0.469	0.580	0.517
Set3	HC	ward	euclidean	0.496	0.573	0.497
Set3	HC	complete	cityblock	0.501	0.530	0.475
Set3	HC	average	euclidean	0.456	0.573	0.503

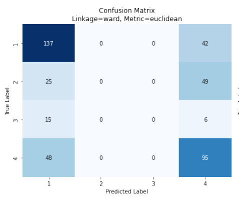
TABLE III: Hierarchical Clustering Results for Set1, Set2, and Set3 using Different Configurations

Table IV shows best Hierarchical Clustering Configurations for each feature set.

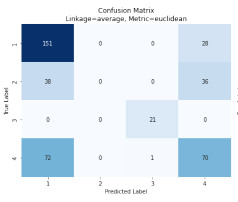
Set	Model	Linkage	Metric	Ratio	Accuracy	Weighted-F1
Set1	HC	ward	euclidean	0.459	0.556	0.486
Set2	HC	average	euclidean	0.469	0.580	0.517
Set3	HC	average	euclidean	0.456	0.573	0.503

TABLE IV: Best Hierarchical Clustering Configurations for Each Feature Set

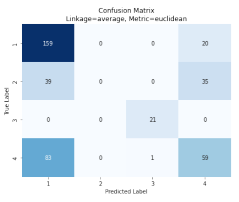
Fig. 7 shows a comparison of the confusion matrices from the best configurations for each feature set.



(a) Set 1



(b) Set 2



(c) Set 3

Fig. 7. Confusion Matrix Comparison of the Best Configurations Across Feature Sets

We find that the hierarchical clustering method successfully identifies Programme 3 in both Set 2 and Set 3, while Set 1 fails to separate Programme 3 from the others and achieves lower accuracy. This result is consistent with the conclusion drawn in Task 1 & 2. Set 2 achieves the best performance in both Accuracy (58.03%) and Weighted-F1 (51.71%), and separates Programme 1 relatively well. Set 3 shows the best clustering performance for Programme 1 but suffered from greater confusion in Programme 4, resulting in slightly lower overall performance compared to Set 2. However, the confusion matrix shows that all three sets cannot predict Programme 2, which needs further consideration.

IV. Task 4

In this task, we need to classify using at least three sets of features. For each set of features, we need to try at least three sets of hyper-parameters of the model. In addition, an ensemble classifier [5] is built with an attempt of 2 sets of different hyperparameters. After that, we need to give the reasons of the experiment design and compare the best performed classification results with the distribution of programmes.

A. Sets Chosen

In this part of the experiment, we still use the same three feature sets obtained from Tasks 1 & 2.

B. Justification of Experiment Design

We use k-nearest neighbor (KNN) classifier, NaiveBayes and DecisionTree in this task.

In KNN classifier, we try three sets of hyperparameters:

- Config 1: n_neighbors: 3, weights: uniform
- Config 2: n_neighbors: 5, weights: uniform
- Config 3: n_neighbors: 5, weights: distance

By using these hyperparameters, we want to discover how different neighbor counts and voting strategies affect the performance of the KNN classifier.

In the Decision Tree classifier, we try three sets of hyperparameters:

- Config 1: max_depth: 9
- Config 2: max_depth: 5
- Config 3: max_depth: 19

By using these hyperparameters, we want to explore how different maximum depths affect the performance and generalization ability of the Decision Tree classifier.

In Naive Bayes classifier, we try three sets of hyperparameters:

- Config 1: priors: [0.25, 0.25, 0.25, 0.25]
- Config 2: priors: [0.429, 0.178, 0.050, 0.343]
- Config 3: priors: [0.4, 0.1, 0.1, 0.4]

By using these hyperparameters, we want to explore how different assumptions about class prior probabilities affect the predictive performance of the Naive Bayes classifier. This is particularly essential in imbalanced datasets, where adjusting priors can help mitigate bias toward majority classes.

Then, we choose the feature set that yields the highest average performance and use it to build an ensemble classifier using weighted soft-voting. And in the ensemble classifier, we optimize the different models' weights by testing different weight combinations to achieve the best overall performance.

C. Results & Discussion

Table V shows the results of three sets of features using KNN classifier with three different hyperparameter settings. Table VI shows the best hyperparameter settings for each feature set. We find that the classifier perform better on Set 2 and Set 3 than Set 1, which means Set 2 and Set 3 have better features, which is similar to the comparison of Task 1& 2. The best performance of different hyperparameter settings of KNN Model shows that configuration 3 is better for Set 1 and Set 2, while configuration 1 is more suitable for Set 3. Possible reasons are that in noisy or high-dimensional spaces such as Set 1 and Set 2, larger n_neighbors plus distance weighting can help, while in clean, low-dimensional embeddings such as Set 3, smaller n_neighbors and uniform voting often work.

Set	**Model**	**n_neighbors**	**weights**	**Accuracy**	**Weighted-F1**
Set1	KNN	3	uniform	0.470	0.446
Set1	KNN	5	uniform	0.453	0.431
Set1	KNN	5	distance	0.468	0.456
Set2	KNN	3	uniform	0.549	0.527
Set2	KNN	5	uniform	0.566	0.541
Set2	KNN	5	distance	0.559	0.550
Set3	KNN	3	uniform	0.580	0.561
Set3	KNN	5	uniform	0.573	0.552
Set3	KNN	5	distance	0.566	0.559

TABLE V: Performance Summary of KNN Model on Set 1–3 with Different Hyperparameter Settings

Set	**Model**	**n_neighbors**	**weights**	**Accuracy**	**Weighted-F1**
Set1	KNN	5	distance	0.468	0.456
Set2	KNN	5	distance	0.559	0.550
Set3	KNN	3	uniform	0.580	0.561

TABLE VI: Best Performance of Hyperparameter Settings of KNN Model for Each Feature Set

Table VII shows the results of three sets of features using Decision Tree classifier with three different hyperparameter settings. Table VIII shows the best hyperparameter settings for each feature set. Fig. 8 shows the decision tree achieving the best performance under the optimal hyperparameter settings on Set 3. We can find that the classifier performs best on Set 1 and slightly worse on Sets 2 and 3. Possible reasons are that Decision tree model are insensitive to feature scaling since they only care about the ordering of values, and applying Z-score normalization or PCA projection can actually disrupt the natural threshold distributions of the original features. The suitable max_depth for Set 1 and Set 2 is 9, and the suitable max_depth for Set 3 is 5. Possible reasons are deeper trees will tend to overfit the data.

Table IX shows the results of three sets of features using Naive Bayes classifier with three different hyperparameter settings. Table X shows the best hyperparameter settings for each feature set. We can find that the classifier performs best on Set 3, then Set 2, and much better than Set 2. The possible reasons for the performance drop observed in Set 2 are that z-score normalization might have disrupted the original scale-dependent patterns that were informative for

Set	Model	max_depth	Accuracy	Weighted-F1
Set1	DecisionTree	9	0.564	0.560
Set1	DecisionTree	5	0.576	0.557
Set1	DecisionTree	19	0.544	0.542
Set2	DecisionTree	9	0.564	0.558
Set2	DecisionTree	5	0.561	0.533
Set2	DecisionTree	19	0.513	0.514
Set3	DecisionTree	9	0.566	0.553
Set3	DecisionTree	5	0.576	0.556
Set3	DecisionTree	19	0.554	0.556

TABLE VII: Performance Summary of Decision Tree Model on Set 1–3 with Different Hyperparameter Settings

Set	Model	max_depth	Accuracy	Weighted-F1
Set1	DecisionTree	9	0.564	0.560
Set2	DecisionTree	9	0.564	0.558
Set3	DecisionTree	5	0.576	0.556

TABLE VIII: Best Performance of Hyperparameter Settings of Decision Tree Model for Each Feature Set

classification, particularly for Naive Bayes, which relies on the distributional properties of features. In contrast, Set 3 uses the PCA transformation from Set 2, which projects all features into an orthogonal space that better satisfies the Naive Bayes assumption of conditional independence between features. As a result, the performance of the model works better in Set 3. Config 2 performs better on Set 3 because it uses class priors that reflect the actual class distribution in the dataset.

Set	Model	Config	Accuracy	Weighted-F1
Set1	NaiveBayes	config1	0.369	0.350
Set1	NaiveBayes	config2	0.405	0.400
Set1	NaiveBayes	config3	0.424	0.426
Set2	NaiveBayes	config1	0.317	0.281
Set2	NaiveBayes	config2	0.309	0.282
Set2	NaiveBayes	config3	0.324	0.307
Set3	NaiveBayes	config1	0.540	0.550
Set3	NaiveBayes	config2	0.580	0.574
Set3	NaiveBayes	config3	0.580	0.548

TABLE IX: Performance Summary of Naive Bayes Model on Set 1–3 with Different Hyperparameter Settings

Table XI shows the best hyperparameter configurations of

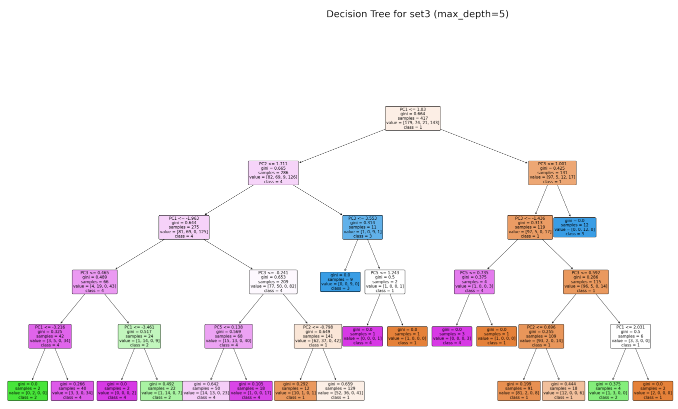


Fig. 8. Decision Tree on Set 3 (max_depth=5)

Set	Model	Config	Accuracy	Weighted-F1
Set1	NaiveBayes	config3	0.424	0.426
Set2	NaiveBayes	config3	0.324	0.307
Set3	NaiveBayes	config2	0.580	0.574

TABLE X: Best Performance of Hyperparameter Settings of Naive Bayes Model for Each Feature Set

each model for each feature set. We can find that the average performance of each model on Set 3 is better than others. So we use Set 3 to built an ensembled classifier.

Set	Model	Config	Accuracy	Weighted-F1
Set1	DecisionTree	config1	0.580	0.577
Set1	KNN	config3	0.468	0.456
Set1	NaiveBayes	config3	0.424	0.426
Set2	DecisionTree	config3	0.576	0.549
Set2	KNN	config3	0.559	0.550
Set2	NaiveBayes	config3	0.324	0.307
Set3	DecisionTree	config2	0.576	0.556
Set3	KNN	config1	0.580	0.561
Set3	NaiveBayes	config2	0.580	0.574

TABLE XI: Best Hyperparameter Configurations per Model for Each Feature Set

In the ensemble classifier, we try different weight combinations to achieve the best overall performance. The results are shown in Table XII. We find that assigning weights of 0.49 to both the Decision Tree and KNN models and 0.02 to the Naive Bayes model yields the highest accuracy (62.1%) and weighted-F1 score (60.5%). The confusion matrix of the bset weights combination of this classifer are shown in Fig. 9.

Set	Weights	Accuracy	Weighted-F1
Set3	[0.33, 0.33, 0.34]	0.600	0.581
Set3	[0.9, 0.001, 0.099]	0.578	0.558
Set3	[0.5, 0.25, 0.25]	0.592	0.575
Set3	[0.25, 0.5, 0.25]	0.600	0.584
Set3	[0.2, 0.6, 0.2]	0.600	0.585
Set3	[0.25, 0.25, 0.5]	0.612	0.596
Set3	[0.2, 0.2, 0.6]	0.619	0.606
Set3	[0.1, 0.8, 0.1]	0.590	0.578
Set3	[0.1, 0.1, 0.8]	0.616	0.606
Set3	[0.49, 0.02, 0.49]	0.621	0.605

TABLE XII: Ensemble Voting Performance on Set 3 with Different Weight Combinations

V. Discussions and Conclusions

A. *New Findings 1: Ensembling Complementary Models Enhances Performance*

1) Supporting Evidence: Tables XI and XII show that on Set 3, single models achieve their best accuracy and F1 scores of approximately 57–58%. However, when different models are ensembled, their combined performance is much better than any single classifier since their accuracy and weighted F1 scores increase about 2% to 3%. In particular, in our experiments, when using a soft-voting ensemble of the Decision Tree, KNN and Naive Bayes with weights [0.49, 0.02, 0.49], the accuracy increases to 62.1% and the weighted F1 score increases to 60.5%. Moreover, Table XII also shows that

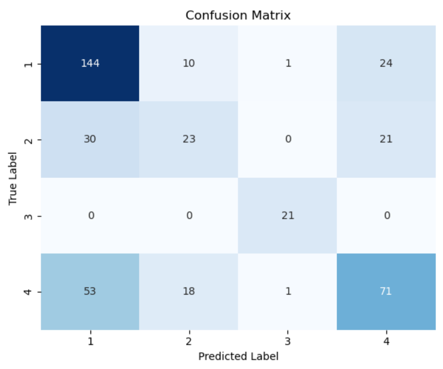


Fig. 9. Confusion Matrix for the Best Weights Combination of Ensemble Classifer.

relying too much on one model can decrease the performance, while combining different models can help. Because each model catches different patterns but misses others, so together they cover each other's blind spots.

2) Suggestions for Further Work : In the future, we can try other kinds of supervised learning methods, and use them in the ensembled classifier. To find the best weights for the combination of different models, we can apply automated optimization techniques to the weight search space in order to more systematically identify the optimal combination.

B. New Findings 2: Intra–Inter Ratio Metric May Undervalue Small Clusters

1) Supporting Evidence: In task 2, according to Table II, Set 1 has the lowest intra–inter cluster distance ratio, which is 0.40, while Set 2 and Set 3 have higher ratios. However, in the t-SNE visualization, Set 1 fails to separate Programme 3 from the other Programme, but Set 2 and Set 3 clearly isolate it. A possible reason is that the ratio metric is insensitive to improvements in tiny clusters, and Programme 3—the class we isolated, has the fewest samples of all the Programmes, which drives the ratio value higher as a result. The hierarchical clustering results in Task 3 reinforce this point: in Table IV, although Set 1 has the best ratio, its accuracy and F1 score are 0.556 and 0.486, which are lower than those of Set 2 (0.580, 0.517) and Set 3 (0.573, 0.503).

2) Suggestions for Further Work : In the future, we can use other clustering metrics for future comparison in imbalanced-cluster scenarios. In addition, we can develop weighted cluster-quality measures or probabilistic clustering models that especially focus on minority clusters, which will provide a fairer assessment of overall cluster structure.

References

[1] Xingcheng Ran, Yue Xi, Yonggang Lu, Xiangwen Wang, and Zhenyu Lu, "Comprehensive survey on hierarchical clustering algorithms and the recent developments," *Artificial Intelligence Review*, vol. 56, no. 8, pp. 8219–8264, 2023.

[2] Benedikt Gierlichs, Lejla Batina, Pim Tuyls, and Bart Preneel, "Mutual information analysis: A generic side-channel distinguisher," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2008, pp. 426–442.

[3] Laurens van der Maaten and Geoffrey Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[4] Abiodun M Ikotun, Absalom E Ezugwu, Laith Abualigah, Belal Abuhaija, and Jia Heming, "K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data," *Information Sciences*, vol. 622, pp. 178–210, 2023.

[5] Ibomoiye Domor Mienye and Yanxia Sun, "A survey of ensemble learning: Concepts, algorithms, applications, and prospects," *Ieee Access*, vol. 10, pp. 99129–99149, 2022.