Fuzzy Systems and Computational Intelligence: TSK Classification Models

Dimitrios Karatis 10775, *Electrical and Computer Engineering, AUTH, Classification.pdf*

***Abstract*—This project explores the application of Takagi–Sugeno–Kang (TSK) fuzzy models to classification problems. Two datasets from the UCI repository are employed. The first, Haberman's Survival dataset, serves as an introductory example to study model training, evaluation, and rule construction under varying partitioning strategies. The second, Epileptic Seizure Recognition dataset, presents a high-dimensional classification challenge, allowing the investigation of feature selection, clustering-based rule reduction, and hyperparameter tuning through cross-validation. Evaluation metrics include the error matrix, overall accuracy (OA), producer's accuracy (PA), user's accuracy (UA), and Cohen's kappa coefficient (κ). Results demonstrate the balance between interpretability and predictive accuracy, highlighting the trade-offs between rule complexity, class-dependent clustering, and feature dimensionality in TSK fuzzy classification.**

***Index Terms*—TSK fuzzy model, classification, fuzzy clustering, feature selection, high-dimensional data, cross validation, performance evaluation.**

Two case studies are carried out:

Haberman's Survival Dataset

The first dataset contains 306 samples with 3 features. The data are partitioned into training (60%), validation (20%), and test (20%) subsets, ensuring balanced class distributions across all partitions. Four TSK models are trained, differing in rule generation and cluster partitioning strategy: class-independent vs. class-dependent subtractive clustering, each tested under two different cluster radius values. One considered *small* and one *big*. Output functions are configured as singletons to align with categorical class outputs. Training uses a hybrid approach: membership function parameters are optimized via backpropagation, while consequent parameters are determined through least-squares estimation. Models are evaluated using classification metrics derived from the error matrix (OA, PA, UA, κ). The results are further analyzed through learning curves, fuzzy membership visualizations, and confusion matrices.

Epileptic Seizure Recognition Dataset

The second dataset contains 11,500 samples with 179 features, posing a high-dimensional classification problem. To address rule explosion, dimensionality reduction via feature selection (using Relief) and rule reduction via subtractive clustering are applied. A grid search with 5-fold cross-validation identifies optimal parameters: the number of selected features and the clustering radius. The final TSK model is trained on the optimal configuration and tested on the reserved test set. Performance is assessed with error matrices, accuracy measures, learning curves, and fuzzy set visualizations. Comparisons are also made, emphasizing the benefits of clustering-based partitioning for scalability.

This document was prepared by Dimitrios Karatis as part of the fourth assignment for the Simulation and Modeling of Dynamic Systems course at Aristotle University of Thessaloniki.

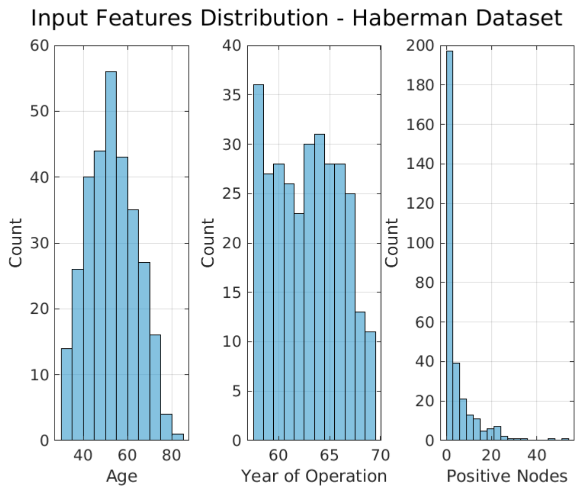


Fig. 1: Distribution for the 3 input features of the Haberman dataset

I. Application on the first dataset (smallest)

A. Methodology Analysis: Class-Independent TSK Fuzzy Classification (Models 1 and 2)

Models 1 and 2 use a **class-independent** approach, meaning that all training data are treated together without considering the class labels when generating fuzzy rules. Model 1 uses a *small* value of radius (0.2), while Model 2 uses a *big* one (0.8).

Data Exploration and Normalization

First, the data are loaded and explored using histograms and plots of each feature against sample index. Features are also plotted colored by class to give an initial idea of how separable the classes are.

Next, all input features are normalized to the range $[0, 1]$. This is important because subtractive clustering is distance-based, so normalization ensures that no feature dominates due to its numerical scale. The output class labels remain unchanged.

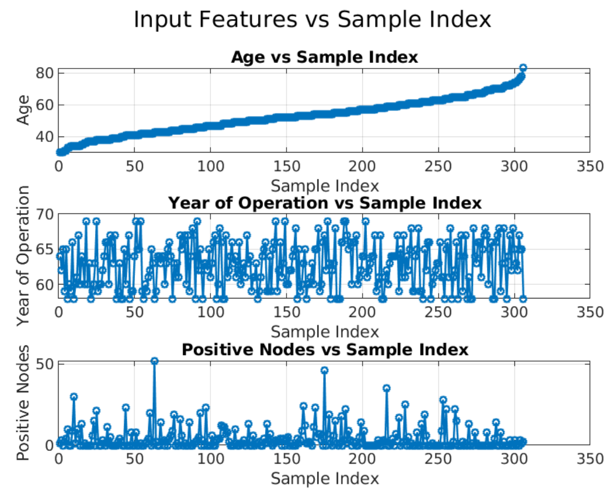


Fig. 2: Input features versus sample index plot before normalization

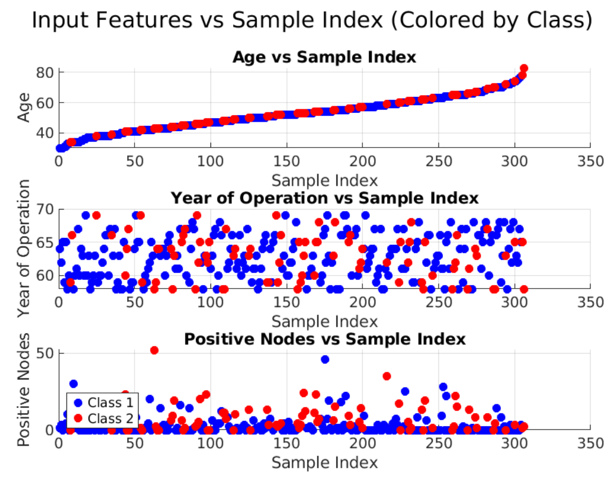


Fig. 3: Input features versus sample index plot before normalization (colored by class)

Data Splitting

To evaluate the model's performance, the dataset is split into training, validation, and test sets using stratified sampling. 60% of the data is used for training, and the remaining 40% is split evenly between validation and testing. Stratification keeps the class proportions consistent in all sets, which is important to avoid bias in learning and evaluation.

Class	Training	Validation	Testing
Class 1	135	45	45
Class 2	49	16	16

TABLE I: Number of Samples per Class and Dataset Split

Building the FIS with Subtractive Clustering

A fuzzy inference system is generated using *subtractive clustering* on the training set. This method finds cluster centers in the input space based on the density of data points. The cluster radius controls the neighborhood of influence: smaller radius means more clusters and therefore more fuzzy rules, while a larger radius results in fewer rules.

Because this is a **class-independent approach**, clustering ignores the class labels. Each cluster corresponds to a fuzzy rule, with Gaussian input membership functions and constant (singleton) outputs.

ANFIS Training

The initial FIS is trained using `ANFIS`, which combines gradient descent and least-squares estimation. Training runs for 100 epochs, using the validation set to monitor performance and prevent overfitting. During training, the shapes of the input membership functions are adjusted to better fit the data, while the output functions stay constant.

Evaluation of the Model

Once training is done, the FIS is tested on the independent test set. Continuous outputs are rounded to the nearest class, and a confusion matrix is computed. From the confusion matrix, we calculate Overall Accuracy (OA), Producer's Accuracy (PA), User's Accuracy (UA), and Kappa coefficient. These metrics give a clear picture of how well the model performs overall and for each class.

Repeating Experiments and Summary Statistics

To account for randomness in data splitting and cluster initialization, the entire process is repeated 30 times. Metrics from each run are stored, and mean and standard deviation are calculated for OA, Kappa, PA, and UA to give a robust estimate of expected model performance.

Representative Run and Visualization

The run whose accuracy is closest to the mean is selected as the representative run. For this run, I visualized membership functions before and after training, confusion matrix, prediction errors, and learning curves. These visualizations help to understand how the model learns and how the rules relate to the data.

B. Methodology Analysis: Class-Dependent TSK Fuzzy Classification (Models 3 and 4)

Models 3 and 4 implement a **class-dependent** TSK fuzzy inference system for the Haberman's Survival dataset. Model 3 uses a *small* value of radius (0.2), while Model 4 uses a *big* one (0.8). Unlike the class-independent approach used in Models 1 and 2, the main idea here is to generate fuzzy rules separately for each class. This allows the model to capture the specific patterns and distributions of each class in the input space.

Class-Dependent Subtractive Clustering

For **class-dependent** modeling, the training data are first split into the usual stratified training, validation, and test sets. Then, for each class individually, *subtractive clustering* is applied to the subset of data belonging to that class. The cluster radius again determines how many clusters are created: smaller radius values lead to more clusters, and therefore more rules.

Since clustering is done per class, the number of clusters can differ for each class, depending on the data density. This often results in a higher total number of fuzzy rules compared to the class-independent approach, as seen in our experiments below (e.g., 49 rules for radius 0.2 versus 22 rules in the independent model). Each cluster corresponds to a Gaussian input membership function, and the output for each cluster is a constant value corresponding to the class.

Building the Sugeno FIS

After clustering, a Sugeno-type FIS is built by adding:

- **Inputs:** One input for each normalized feature.
- **Outputs:** A single output representing the class, with a constant output MF for each cluster.
- **Rules:**
  - Each cluster found by the subtractive clustering step becomes a fuzzy rule.
  - The *IF* part of the rule is described by Gaussian membership functions for all three inputs. These Gaussians check how close a given input is to the cluster center: if the input is very close, the rule *fires* strongly, if it's far, the rule *fires* weakly.
  - The *THEN* part of the rule is just a constant that equals the class label of that cluster.
  - In other words, every rule says something like: ***IF*** *the inputs look like this cluster,* ***THEN*** *the sample belongs to Class 1 (or 2).*

Because rules are generated separately per class, the FIS effectively has multiple subspaces in the input domain, each tuned to one class. This allows the model to better distinguish between classes, especially if their distributions are different.

Training with ANFIS and Evaluation

Training is performed with `ANFIS` using hybrid learning (gradient descent + least squares) for 100 epochs, with a validation set to prevent overfitting. After training, the FIS is evaluated on the independent test set. Continuous outputs are rounded to the nearest class label, and the performance is measured using the same metrics as for the independent models: Overall Accuracy (OA), Producer's Accuracy (PA), User's Accuracy (UA), and Cohen's Kappa.

Multiple Runs and Representative Run

As with Models 1 and 2, 30 independent runs are performed to account for randomness in splitting and clustering. Mean and standard deviation are reported for all metrics. The run whose OA is closest to the mean is selected as the representative run, and visualizations such as membership functions, confusion matrix, prediction errors, and learning curves are produced to analyze model behavior.

Detailed code can be found in the `tsk_model_1.m`*,* `tsk_model_2.m`*,* `tsk_model_3.m` *and* `tsk_model_4.m` *matlab files.*

TSK Model 1

This Model implements a class-independent TSK FIS with **radius = 0.2**. First, we present the initial and final shapes of the membership functions for the fuzzy input variables:

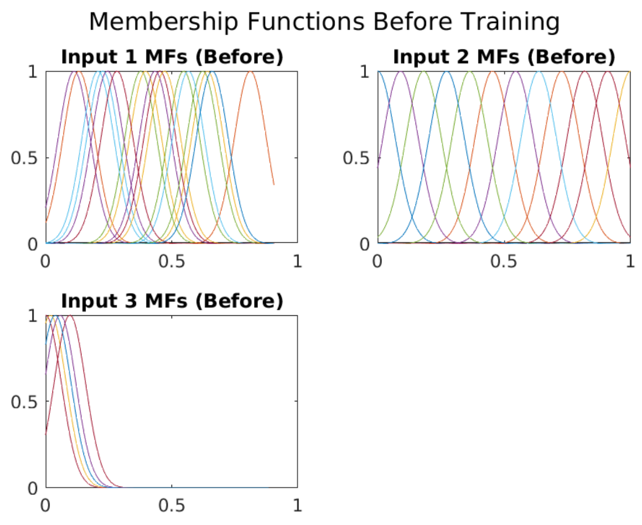


Fig. 4: MFs before training:TSK Model 1

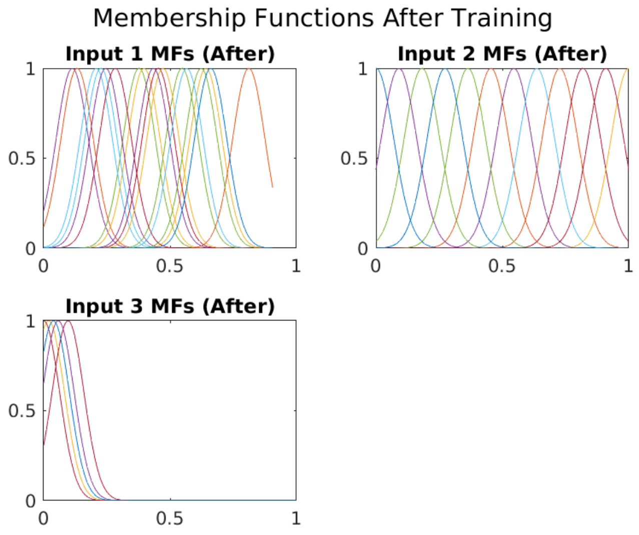


Fig. 5: MFs after training: TSK Model 1

Next, we show the learning (training) curves of the model, along with the prediction errors when applying the trained model to the testing set:

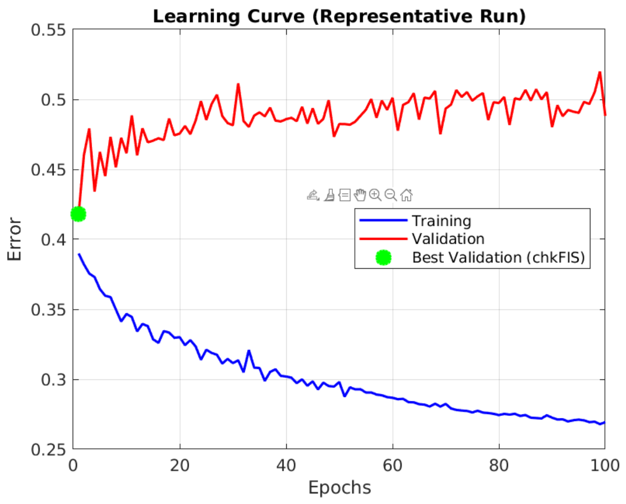


Fig. 6: Learning curves: TSK Model 1

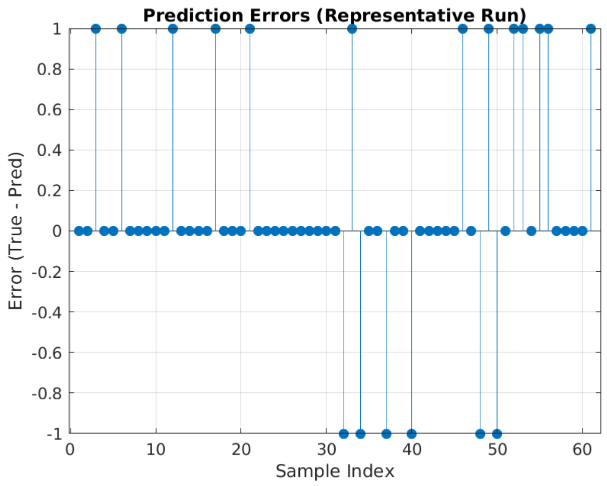


Fig. 7: Prediction errors: TSK Model 1

After that, the requested confusion matrix is presented:

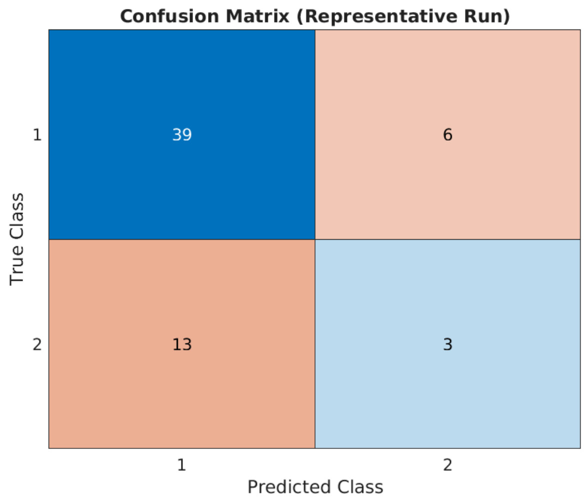
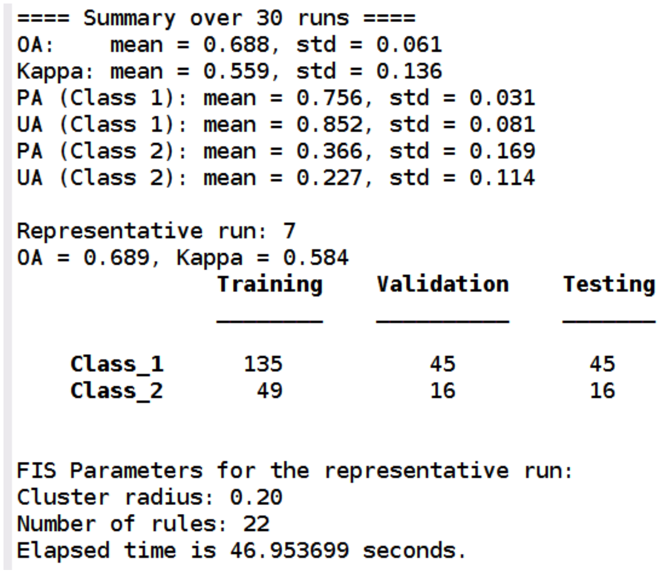


Fig. 8: Confusion Matrix: TSK Model 1

True Class	Predicted 1	Predicted 2
1	39	6
2	13	3

TABLE II: Confusion Matrix in tabular form (Representative Run)

Finally, the requested performance metrics of the model are shown below:



```
==== Summary over 30 runs ====
OA:    mean = 0.688, std = 0.061
Kappa: mean = 0.559, std = 0.136
PA (Class 1): mean = 0.756, std = 0.031
UA (Class 1): mean = 0.852, std = 0.081
PA (Class 2): mean = 0.366, std = 0.169
UA (Class 2): mean = 0.227, std = 0.114

Representative run: 7
OA = 0.689, Kappa = 0.584
             Training    Validation    Testing
             ________    __________    _______

    Class_1     135          45           45
    Class_2      49          16           16


FIS Parameters for the representative run:
Cluster radius: 0.20
Number of rules: 22
Elapsed time is 46.953699 seconds.
```

Fig. 9: Final Metrics (Mean Values): TSK Model 1

Metric	Value
Overall Accuracy (OA, mean ± std)	0.688 ± 0.061
Cohen's Kappa (mean ± std)	0.559 ± 0.136
PA (Class 1, mean ± std)	0.756 ± 0.031
PA (Class 2, mean ± std)	0.366 ± 0.169
UA (Class 1, mean ± std)	0.852 ± 0.081
UA (Class 2, mean ± std)	0.227 ± 0.114
Cluster radius	0.20
Number of rules	22
Elapsed time for 30 runs (secs)	46.95

TABLE III: Performance Summary and FIS Parameters (Mean over 30 runs)

The first model was trained using a class-independent approach with a small cluster radius of 0.2. Table III summarizes its performance, showing an average Overall Accuracy (OA) of 0.688 ± 0.061 and a Cohen's Kappa of 0.559 ± 0.136 over 30 runs. Cohen's Kappa measures how much the model's predictions agree with the true labels compared to what would be expected by random chance. A Kappa of 0.559 falls into the "moderate agreement" range, indicating that the model performs noticeably better than guessing but still leaves room for improvement. That being said, the model performed noticeably better on Class 1 than on Class 2: the Producer's Accuracy (PA) and User's Accuracy (UA) for Class 1 were 0.756 ± 0.031 and 0.852 ± 0.081, whereas for Class 2 they were much lower at 0.366 ± 0.169 and 0.227 ± 0.114, suggesting a bias toward the majority class (Class 1).

The confusion matrix in Table II further highlights this trend. Out of 45 true samples from Class 1, 39 were correctly classified while 6 were misclassified as Class 2. For Class 2, only 3 out of 16 samples were correctly identified, with 13 misclassified as Class 1. This confirms that the model struggles to capture the minority class, consistent with the lower PA and UA values for Class 2, which is expected given the strong class imbalance of the dataset.

The model generated 22 fuzzy rules under this configuration, forming a relatively compact rule base. The total training time was around 47 seconds for 30 runs, showing that the system is computationally efficient. However, the training and validation curves suggest signs of overfitting: while the training error steadily decreases, the validation error rises, with the best validation performance (`chkFIS`) occurring very early in training. Consequently, the membership functions (MFs) before training look almost identical to the MFs after training, indicating that the model barely adapted to the training data. This is likely due to the small dataset and class imbalance, which caused early stopping and limited meaningful learning, as also reflected in Fig. 7.

Table I shows the dataset split for the Haberman's Survival dataset, which is relatively small (306 samples in total). Class 1 contains 225 samples, while Class 2 has only 81 samples, **creating a significant class imbalance**. This makes the classification task challenging, as fuzzy clustering tends to generate rules that fit the dominant class better than the minority class, explaining the stronger results for Class 1 and weaker performance for Class 2. This tendency remains a major challenge for the subsequent models as well.

TSK Model 2

This Model implements a class-independent TSK FIS with **radius = 0.8**. First, we present the initial and final shapes of the membership functions for the fuzzy input variables:

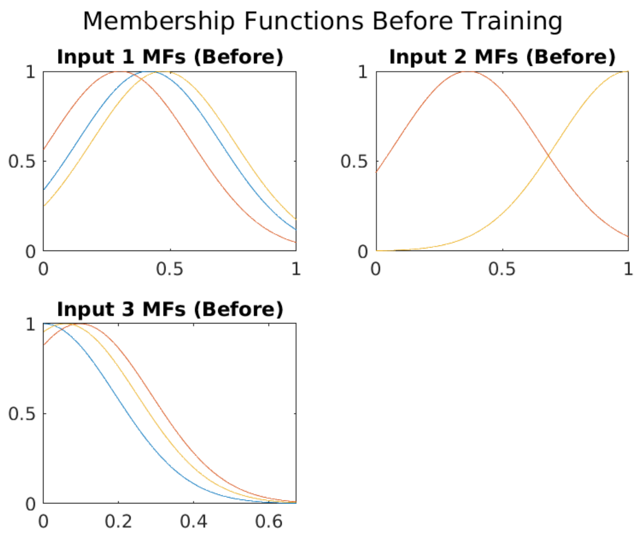


Fig. 10: MFs before training:TSK Model 2

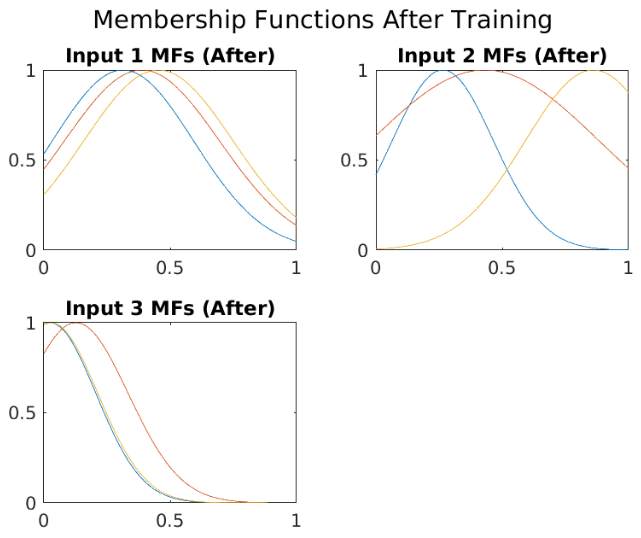


Fig. 11: MFs after training: TSK Model 2

Next, we show the learning (training) curves of the model, along with the prediction errors when applying the trained model to the testing set:

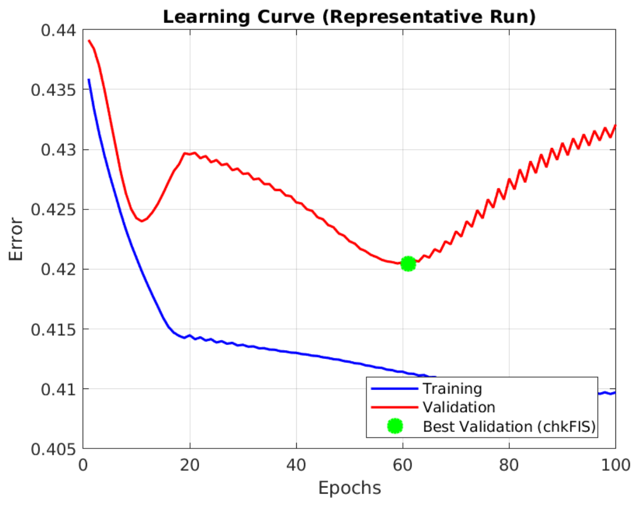


Fig. 12: Learning curves: TSK Model 2

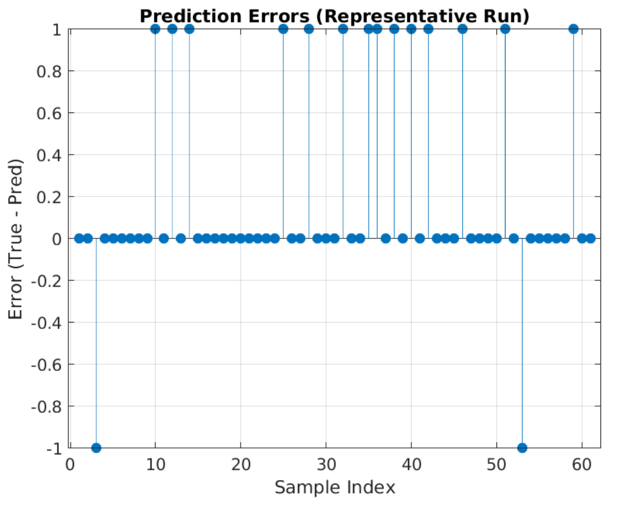


Fig. 13: Prediction errors: TSK Model 2

After that, the requested confusion matrix is presented:

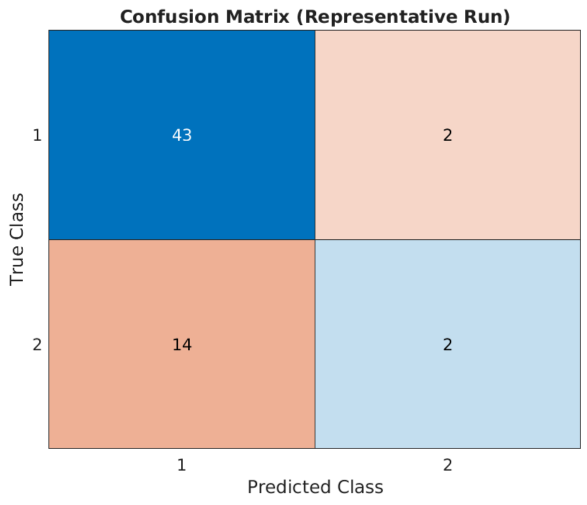
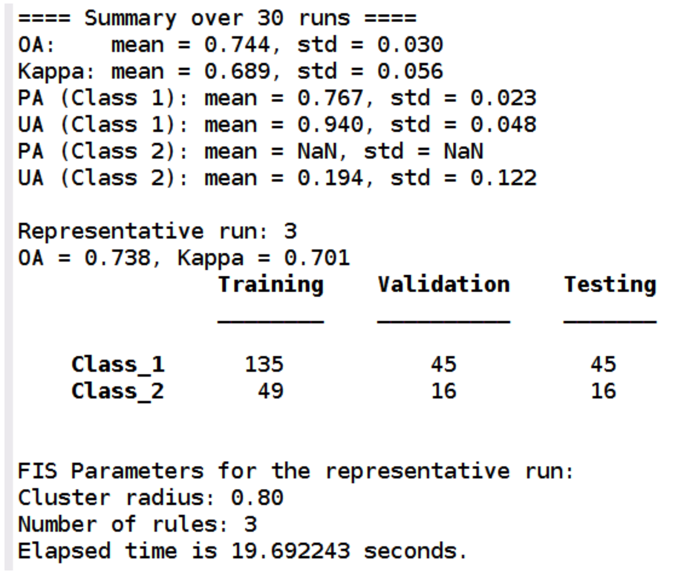


Fig. 14: Confusion Matrix: TSK Model 2

True Class	Predicted 1	Predicted 2
1	43	2
2	14	2

TABLE IV: Confusion Matrix in tabular form (Representative Run)

Finally, the requested performance metrics of the model are shown below:



```
==== Summary over 30 runs ====
OA:    mean = 0.744, std = 0.030
Kappa: mean = 0.689, std = 0.056
PA (Class 1): mean = 0.767, std = 0.023
UA (Class 1): mean = 0.940, std = 0.048
PA (Class 2): mean = NaN, std = NaN
UA (Class 2): mean = 0.194, std = 0.122

Representative run: 3
OA = 0.738, Kappa = 0.701
               Training    Validation    Testing
               ________    __________    _______

    Class_1      135           45           45
    Class_2       49           16           16


FIS Parameters for the representative run:
Cluster radius: 0.80
Number of rules: 3
Elapsed time is 19.692243 seconds.
```

Fig. 15: Final Metrics (Mean Values): TSK Model 2

Metric	Value
Overall Accuracy (OA, mean $\pm$ std)	0.744 ± 0.030
Cohen's Kappa (mean $\pm$ std)	0.689 ± 0.056
PA (Class 1, mean $\pm$ std)	0.767 ± 0.023
PA (Class 2, mean $\pm$ std)	NaN
UA (Class 1, mean $\pm$ std)	0.940 ± 0.048
UA (Class 2, mean $\pm$ std)	0.194 ± 0.122
Cluster radius	0.80
Number of rules	3
Elapsed time for 30 runs (secs)	19.69

TABLE V: Performance Summary and FIS Parameters (Mean over 30 runs) for Model 2

The second model was trained using a class-independent approach with a larger cluster radius of 0.8. Table V summarizes the performance over 30 runs, showing an average Overall Accuracy (OA) of 0.744 ± 0.030 and a Cohen's Kappa of 0.689 ± 0.056, indicating an improvement over Model 1. The model performed considerably better on Class 1 than on Class 2: Producer's Accuracy (PA) and User's Accuracy (UA) for Class 1 were 0.767 ± 0.023 and 0.940 ± 0.048. For Class 2, the Producer's Accuracy (PA) could not be calculated (NaN) because in several runs the model failed to correctly classify any samples from this class. Lastly, the User's Accuracy (UA) was very low (0.194 ± 0.122), highlighting that the model strongly favors the majority class.

The confusion matrix for a representative run (Table IV) shows that 43 out of 45 Class 1 samples were correctly classified, while 2 were misclassified as Class 2. For Class 2, only 2 out of 16 samples were correctly classified, with 14 misclassified as Class 1. This again confirms that the model struggles to capture the minority class, consistent with the class imbalance in the dataset (225 samples in Class 1 versus 81 in Class 2).

Unlike Model 1, the membership functions (MFs) in this model changed noticeably during training, showing that the system did adapt to the data. Training stopped at epoch 60 when the best validation performance was reached, after which overfitting occurred: the training error continued to decrease while the validation error rose. Interestingly, in the initial configuration before training, input 2 had only 2 MFs, whereas all other inputs had 3. This difference is due to the larger cluster radius (0.8), which caused the clustering algorithm to merge regions of input 2 that were close together, resulting in fewer initial MFs for that input. After training, input 2 also had 3 MFs, because the training process adjusted the MFs to better capture the distribution of input 2, effectively creating a new MF where the data required more detailed representation.

The model generated only 3 fuzzy rules, which is a very compact rule base. The elapsed time for 30 runs was approximately 19.7 seconds, showing high computational efficiency and improvement over model 1. Overall, the larger cluster radius allowed the MFs to adapt more meaningfully than in Model 1, but the model still struggles with the minority class due to the strong class imbalance and limited dataset size.

TSK Model 3

This Model implements a class-dependent TSK FIS with **radius = 0.2**. First, we present the initial and final shapes of the membership functions for the fuzzy input variables:

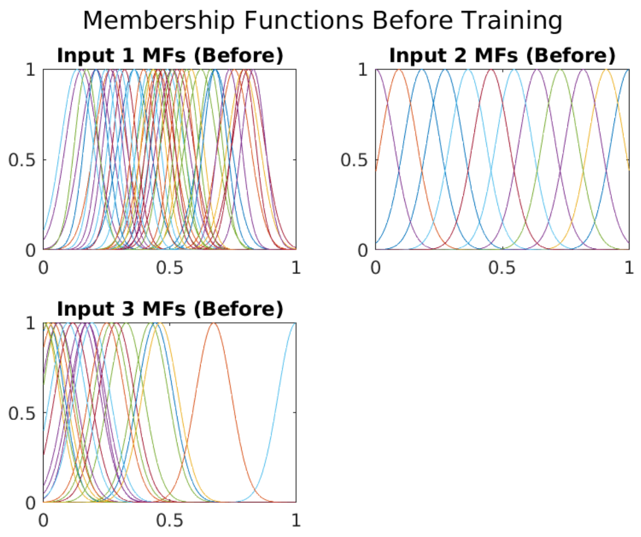


Fig. 16: MFs before training:TSK Model 3

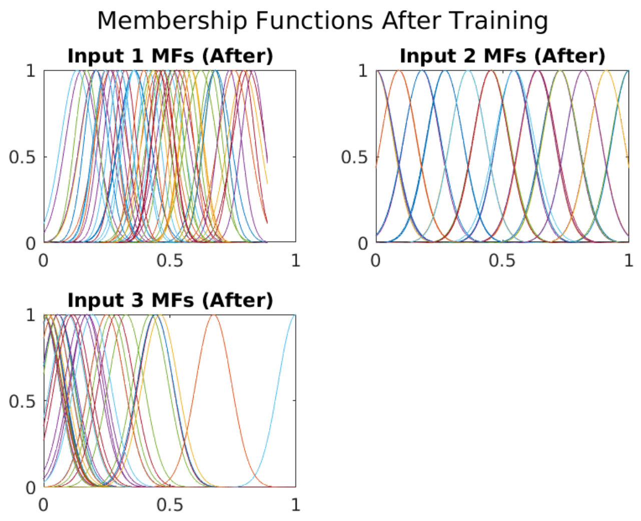


Fig. 17: MFs after training: TSK Model 3

Next, we show the learning (training) curves of the model, along with the prediction errors when applying the trained model to the testing set:

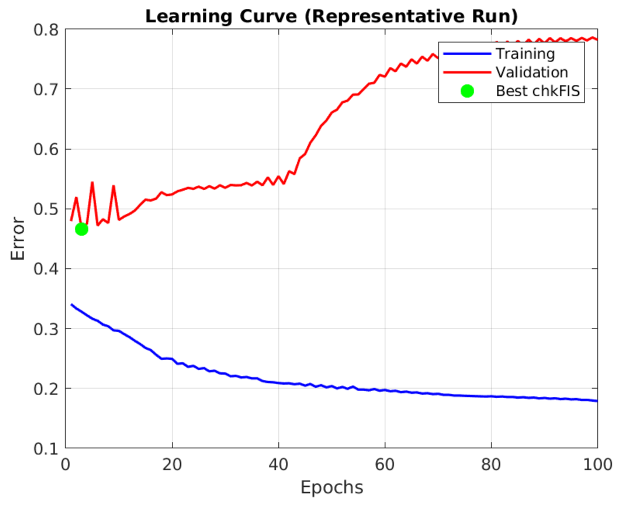


Fig. 18: Learning curves: TSK Model 3

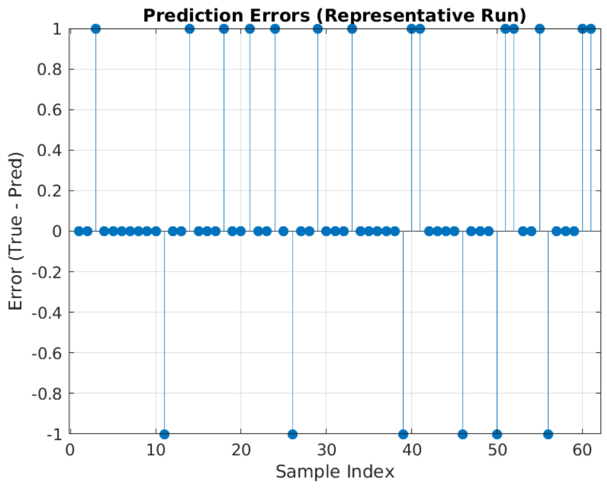


Fig. 19: Prediction errors: TSK Model 3

After that, the requested confusion matrix is presented:

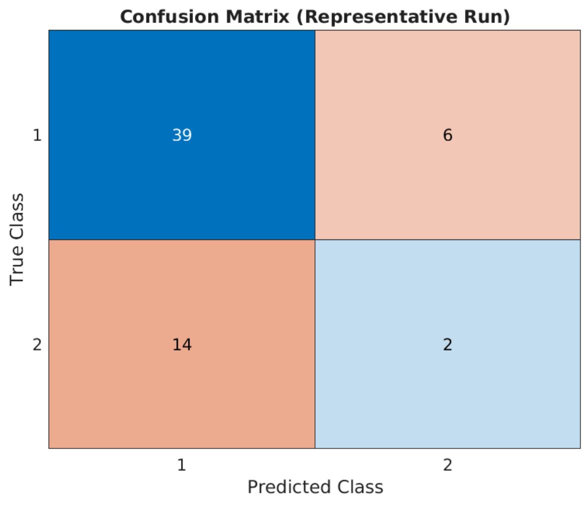
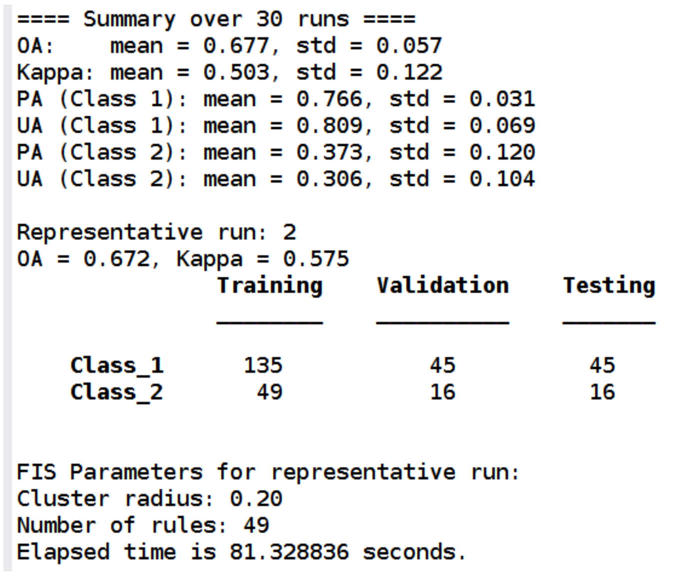


Fig. 20: Confusion Matrix: TSK Model 3

True Class	Predicted 1	Predicted 2
1	39	6
2	14	2

TABLE VI: Confusion Matrix in tabular form (Representative Run)

Finally, the requested performance metrics of the model are shown below:



```
==== Summary over 30 runs ====
OA:    mean = 0.677, std = 0.057
Kappa: mean = 0.503, std = 0.122
PA (Class 1): mean = 0.766, std = 0.031
UA (Class 1): mean = 0.809, std = 0.069
PA (Class 2): mean = 0.373, std = 0.120
UA (Class 2): mean = 0.306, std = 0.104

Representative run: 2
OA = 0.672, Kappa = 0.575
                     Training    Validation    Testing
                     ________    __________    _______

    Class_1            135           45           45
    Class_2             49           16           16


FIS Parameters for representative run:
Cluster radius: 0.20
Number of rules: 49
Elapsed time is 81.328836 seconds.
```

Fig. 21: Final Metrics (Mean Values): TSK Model 3

Metric	Value
Overall Accuracy (OA, mean ± std)	0.677 ± 0.057
Cohen's Kappa (mean ± std)	0.503 ± 0.122
PA (Class 1, mean ± std)	0.766 ± 0.031
PA (Class 2, mean ± std)	0.373 ± 0.104
UA (Class 1, mean ± std)	0.809 ± 0.069
UA (Class 2, mean ± std)	0.306 ± 0.120
Cluster radius	0.20
Number of rules	49
Elapsed time for 30 runs (secs)	81.33

TABLE VII: Performance Summary and FIS Parameters (Mean over 30 runs) for Model 3 (Class-Dependent, radius = 0.2)

The third model was trained using a class-dependent approach with a small cluster radius of 0.2. Table VII summarizes the performance over 30 runs, showing an average Overall Accuracy (OA) of 0.677 ± 0.057 and a Cohen's Kappa of 0.503 ± 0.122. Cohen's Kappa measures how much the model's predictions agree with the true labels compared to what would be expected by random chance. A Kappa of 0.503 falls into the "moderate agreement" range, indicating that the model performs better than random guessing but still leaves room for improvement. That being said, the model performed better on Class 1 than on Class 2: Producer's Accuracy (PA) and User's Accuracy (UA) for Class 1 were 0.766 ± 0.031 and 0.809 ± 0.069, whereas for Class 2 they were lower at 0.373 ± 0.104 and 0.306 ± 0.120, suggesting that the model still favors the majority class, although the bias is slightly reduced compared to Model 1.

The confusion matrix for a representative run (Table VI) highlights this trend. Out of 45 true samples from Class 1, 39 were correctly classified while 6 were misclassified as Class 2. For Class 2, only 2 out of 16 samples were correctly identified, with 14 misclassified as Class 1. This confirms that the model struggles to capture the minority class, consistent with the lower PA and UA values for Class 2, which is expected given the strong class imbalance in the dataset.

The model generated 49 fuzzy rules for this configuration, forming a much larger rule base compared to Model 1. The total training time for 30 runs was approximately 81.3 seconds, reflecting the additional computational cost of the increased number of rules. Unlike Model 1, the membership functions (MFs) in this model changed slightly during training, showing that the system did adapt to the data, albeit modestly. Training stopped at around epoch 7 when the best validation performance was reached, after which overfitting occurred: the training error continued to decrease while the validation error increased. This indicates that the model quickly fit the training data but struggled to generalize to unseen samples, likely due to the small dataset and class imbalance, as seen in Fig. 19.

Compared to Model 1, which was class-independent with the same small cluster radius, Model 3 shows a slightly lower Overall Accuracy (0.677 vs. 0.688) and Cohen's Kappa (0.503 vs. 0.559), indicating a small drop in overall agreement. However, the class-dependent approach in Model 3 allows for a larger number of fuzzy rules (49 vs. 22), giving the system more flexibility to model complex patterns. The membership functions also adapt slightly during training in Model 3, unlike in Model 1 where they remained almost unchanged. Both models exhibit overfitting: using a small cluster radius generates many rules, and because the dataset is small the models were more powerful than needed. Despite these differences, both models still struggle with the minority class, with Class 2 achieving lower PA and UA in both cases.

TSK Model 4

This Model implements a class-dependent TSK FIS with **radius = 0.8**. First, we present the initial and final shapes of the membership functions for the fuzzy input variables:

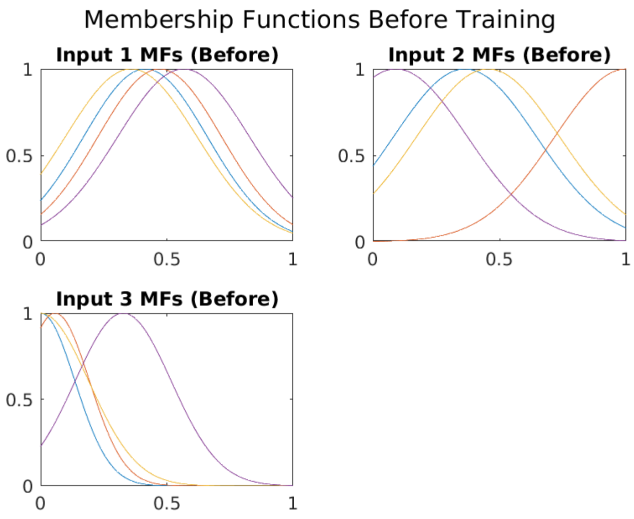


Fig. 22: MFs before training:TSK Model 4

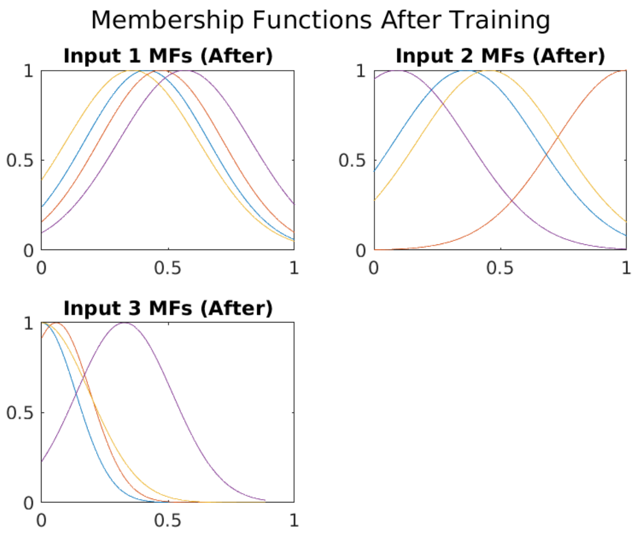


Fig. 23: MFs after training: TSK Model 4

Next, we show the learning (training) curves of the model, along with the prediction errors when applying the trained model to the testing set:

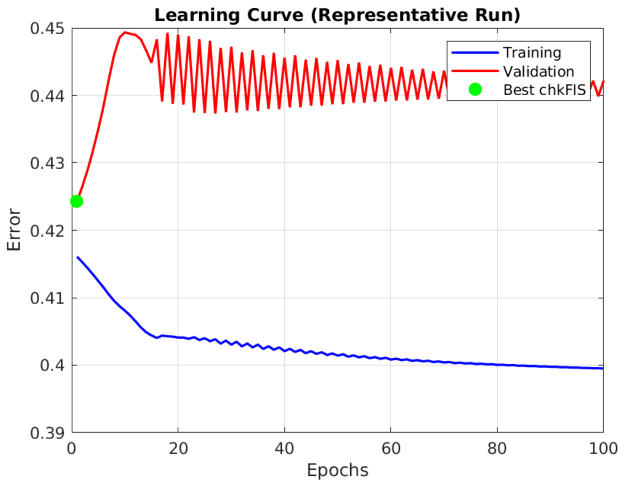


Fig. 24: Learning curves: TSK Model 4

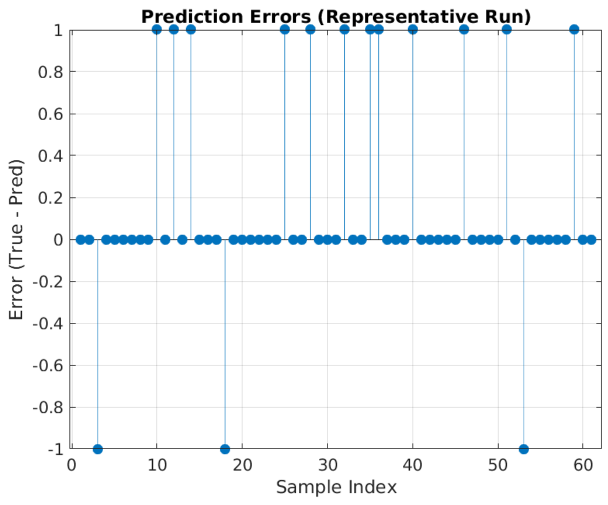


Fig. 25: Prediction errors: TSK Model 4

After that, the requested confusion matrix is presented:

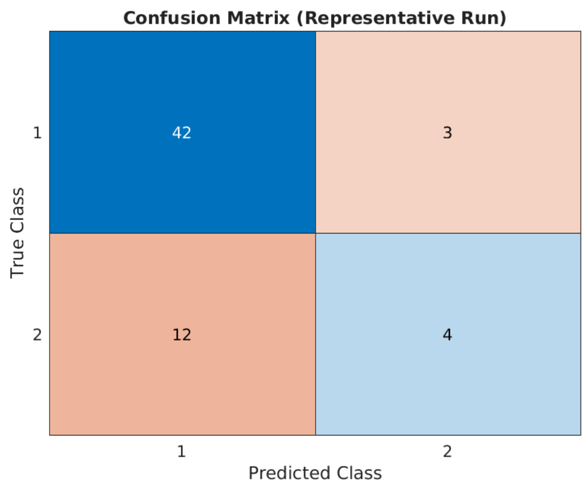
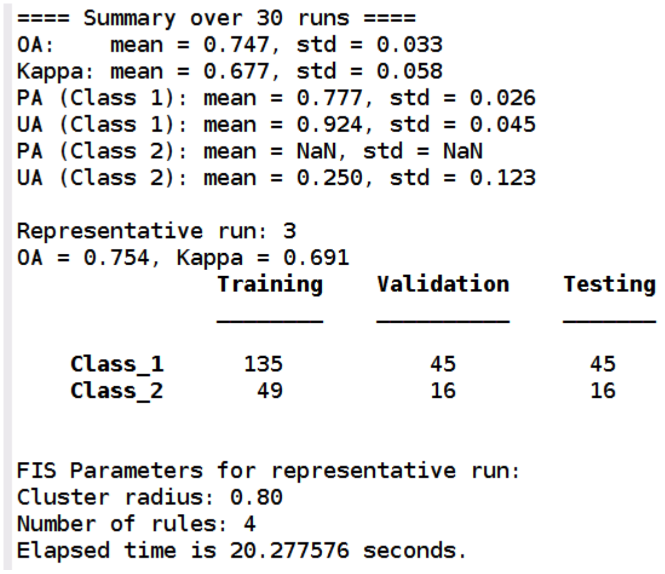


Fig. 26: Confusion Matrix: TSK Model 4

True Class	Predicted 1	Predicted 2
1	42	3
2	12	4

TABLE VIII: Confusion Matrix in tabular form (Representative Run)

Finally, the requested performance metrics of the model are shown below:



```
==== Summary over 30 runs ====
OA:    mean = 0.747, std = 0.033
Kappa: mean = 0.677, std = 0.058
PA (Class 1): mean = 0.777, std = 0.026
UA (Class 1): mean = 0.924, std = 0.045
PA (Class 2): mean = NaN, std = NaN
UA (Class 2): mean = 0.250, std = 0.123

Representative run: 3
OA = 0.754, Kappa = 0.691
                    Training    Validation    Testing
                    ________    __________    _______

    Class_1           135           45           45
    Class_2            49           16           16


FIS Parameters for representative run:
Cluster radius: 0.80
Number of rules: 4
Elapsed time is 20.277576 seconds.
```

Fig. 27: Final Metrics (Mean Values): TSK Model 4

Metric	Value
Overall Accuracy (OA, mean $\pm$ std)	0.747 ± 0.033
Cohen's Kappa (mean $\pm$ std)	0.677 ± 0.058
PA (Class 1, mean $\pm$ std)	0.777 ± 0.026
PA (Class 2, mean $\pm$ std)	NaN
UA (Class 1, mean $\pm$ std)	0.924 ± 0.045
UA (Class 2, mean $\pm$ std)	0.250 ± 0.123
Cluster radius	0.80
Number of rules	4
Elapsed time for 30 runs (secs)	20.28

TABLE IX: Performance Summary and FIS Parameters (Mean over 30 runs) for Model 4 (Class-Dependent, radius = 0.8)

The fourth model was trained using a class-dependent approach with a large cluster radius of 0.8. Table IX summarizes the performance over 30 runs, showing an average Overall Accuracy (OA) of 0.747 ± 0.033 and a Cohen's Kappa of 0.677 ± 0.058, which is comparable to Model 2. The model, again, performed considerably better on Class 1 than on Class 2: Producer's Accuracy (PA) and User's Accuracy (UA) for Class 1 were 0.777 ± 0.026 and 0.924 ± 0.045. For Class 2, the PA could not be calculated (NaN), same as model 2, because in several runs it failed to correctly classify enough samples from this class, while UA was very low (0.250 ± 0.123), highlighting that the model still favors the majority class.

The confusion matrix for a representative run (Table VIII) shows that 42 out of 45 Class 1 samples were correctly classified, while 3 were misclassified as Class 2. For Class 2, only 4 out of 16 samples were correctly classified, with 12 misclassified as Class 1. This confirms that the model struggles with the minority class, consistent with the low

PA and UA values for Class 2, which is expected given the strong class imbalance in the dataset.

The membership functions (MFs) for this model appear almost identical before and after training, indicating that the system adapted very little to the data. Training stopped very early, after which overfitting occurred: the training error continued to decrease while the validation error increased.

The elapsed time for 30 runs was approximately 20.3 seconds, showing high computational efficiency. Overall, the model is compact and fast, but like every other model, it struggles with the minority class due to the strong class imbalance.

Both Models 4 and 2 use a large cluster radius (0.8), but Model 4 is class-dependent while Model 2 is class-independent. Both models achieved similar overall performance (OA and Kappa), and both struggle to classify the minority class (Class 2), with PA undefined in some runs and low UA. The main difference is in the number of rules: Model 4 has 4 rules, slightly more than the 3 rules in Model 2, due to the class-dependent approach. Unlike Model 2, the MFs in Model 4 did not change during training, likely because the training stopped very early and the rule base quickly overfit the training data. Both models exhibit overfitting, as indicated by rising validation errors despite decreasing training errors but model 4's was worse. That extra rule compared to model 2 seems to have made the model even stronger, leading to amplified training but poor generalization abilities.

Concluding Analysis

To begin with, the performance metrics of all models are presented in tabular form:

Model	Overall Accuracy (OA)	Kappa (K)
Model 1	0.688	0.559
Model 2	0.744	0.689
Model 3	0.677	0.503
Model 4	0.747	0.677

TABLE X: Comparison of Overall Accuracy (OA) and Cohen's Kappa (K) for all four models

Overall Summary of Findings

Looking at the performance metrics and analyses above, several observations can be made:

- **Model 1 (Class-Independent, radius 0.2, "small"):**
  - Achieved moderate performance with an Overall Accuracy (OA) of 0.688 ± 0.061 and Cohen's Kappa of 0.559 ± 0.136, indicating moderate agreement that is beyond chance.
  - Performance was noticeably better for Class 1 than Class 2, highlighting bias toward the majority class.
  - Generated 22 fuzzy rules and training time was short (46.95 s), making the model computationally efficient.
  - Overfitting was evident: training error decreased while validation error rose, and the membership functions (MFs) barely changed from before to after training due to early stopping.
- **Model 2 (Class-Independent, radius 0.8, "big"):**
  - Improved OA (0.744 ± 0.030) and Kappa (0.689 ± 0.056) compared to Model 1, showing that a larger cluster radius allowed the model to adapt slightly better.
  - Strong bias toward Class 1 remained: PA for Class 2 could not be calculated (NaN) in several runs, and UA for Class 2 was very low (0.194 ± 0.122).
  - Membership functions changed noticeably during training, especially input 2 which went from 2 to 3 MFs, reflecting adaptation to the data.
  - Model generated only 3 fuzzy rules, with very short training time (19.69 s).
  - Overfitting occurred: training error decreased while validation error rose after epoch 60.
- **Model 3 (Class-Dependent, radius 0.2, "small"):**
  - OA (0.677 ± 0.057) and Kappa (0.503 ± 0.122) were slightly lower than Model 1, but the class-dependent approach allowed for 49 rules and slightly adaptive membership functions.
  - Performance for Class 2 improved slightly (PA 0.373 ± 0.104, UA 0.306 ± 0.120), although Class 1 still dominated.
  - Training time was longer (81.33 s) due to the larger number of rules.
  - Overfitting was evident: best validation performance reached at epoch 7, followed by rising validation error.
  - Compared to Model 1, the class-dependent approach provides more flexibility with rules and slight MF adaptation, but the overall accuracy is similar and the minority class is still challenging.
- **Model 4 (Class-Dependent, radius 0.8, "big"):**
  - OA (0.747 ± 0.033) and Kappa (0.677 ± 0.058) were similar to Model 2, showing that the large radius supports strong overall performance.
  - PA for Class 2 could not be calculated (NaN) in several runs, and UA remained low (0.250 ± 0.123), confirming continued bias toward Class 1.
  - Membership functions remained almost unchanged before and after training, indicating minimal adaptation due to very early stopping.
  - The model generated 4 rules with short training time (20.28 s).
  - Overfitting occurred, with training error decreasing and validation error increasing; the extra rule compared to Model 2 clearly has amplified overfitting effects.
  - Compared to Model 2, Model 4 shows similar OA and Kappa, slightly more rules, and faster

convergence, but still struggles with the minority class and major overfitting.

- **General Conclusions:**
  - **Cluster radius matters:** Smaller radius (0.2) produced many rules, leading to stronger fitting and overfitting, while larger radius (0.8) produced fewer rules, faster training, and slightly better overall adaptation for some inputs.
  - **Class-dependency improves flexibility:** Class-dependent models (3 and 4) allow more rules and better modeling of complex patterns, though overall OA and Kappa were similar to class-independent models.
  - **Minority class remains challenging:** Across all models, Class 2 consistently had low PA and UA due to the strong class imbalance (225 vs. 81 samples). Overfitting amplifies this effect.
  - **Trade-off between complexity and efficiency:** Models with many rules (Models 1 and 3) take longer to train, and have a tendency to overfit in a small dataset like this, whereas models with fewer rules (Models 2 and 4) are faster with a better generalization ability.

Observations on Membership Function Overlaps and Rule Activation

The amount of overlap between the membership functions (MFs) for each input has a direct impact on rule activation and classifier performance. When MFs overlap, multiple rules can fire for the same input, which helps smooth the output and can improve generalization. On the other hand, too much overlap may make the rules less distinct and reduce the classifier's sensitivity to subtle differences in the data, which is especially problematic for the minority class. Very little overlap can make the system rigid, where each input activates only a few rules, potentially causing overfitting when the dataset is small and the number of rules is high, as we saw in Models 1 and 3. In our models, there was quite a lot of overlap, especially in Models 2 and 4, which had the larger cluster radius, but also noticeable overlaps were shown in Models 1 and 3.

To improve the design of the input membership functions in the FIS, one approach is to optimize the cluster radius and the initial MFs to match the data distribution better. A balanced overlap, large enough to allow multiple rules to fire but small enough to keep rules distinct, can improve classifier performance. Adaptive clustering methods, like data-driven tuning of MFs, could also help adjust both the number and shape of the MFs according to the data. Finally, techniques such as oversampling the minority class could help create more representative MFs for underrepresented classes, reducing bias and improving overall classifier performance.

Concluding Observations and Comments

Overall, the experiments show that class-dependent rules and cluster radius selection significantly impact FIS performance. Class-dependent models (Models 3 and 4) allow for more flexible rule bases and slightly better adaptation of the membership functions, while class-independent models (Models 1 and 2) produce fewer rules and sometimes show limited MF adaptation. Models 2 and 4, which use a larger cluster radius, achieve slightly higher overall accuracy and Kappa compared to the small-radius models, but all models struggle with minority class classification due to the strong class imbalance (Class 2). Overfitting is common, especially when many rules are generated on a small dataset, as in Models 1 and 3. To reduce bias toward the majority class, techniques such as oversampling the minority class etc could be applied.

II. Application on the largest (second) dataset

Methodology Analysis

Data Preprocessing

For this part of the project the **Epileptic Seizure Recognition dataset**, which has 11,500 samples described by 179 features was used. Since the first column of the dataset was just an ID, it was removed. All other features were normalized to the range $[0, 1]$ so that they were on a common scale, while the class labels were kept unchanged. This helped prevent features with larger numeric ranges from dominating the fuzzy system.

The dataset was then split into three subsets in a stratified way, so that the class distribution remains consistent across all subsets. The split followed the 60%–20%–20% rule: 60% of the data for training, 20% for validation, and 20% for testing. The resulting class distribution is shown in the Table below.

Class	Training	Validation	Testing
1	1380	460	460
2	1380	460	460
3	1380	460	460
4	1380	460	460
5	1380	460	460

TABLE XI: Stratified data split across training, validation, and testing sets.

We can see that the split keeps the classes evenly distributed, so each subset is a good representation of the whole dataset.

Feature Selection

Because the dataset is high-dimensional, using all 178 features (the last column/feature is the output) would lead to rule explosion (an exponential increase in fuzzy rules). To avoid this, I used the `ReliefF` algorithm to rank features according to their importance for classification.

For this I chose a neighborhood size of $k = 50$ for `ReliefF`. The reason was that this value is large enough to capture meaningful dependencies between features and labels without being too noisy or unstable. From the ranking returned by `ReliefF`, different numbers of top features were later selected during grid search.

Fuzzy System Design

The fuzzy inference system was designed using **class-dependent subtractive clustering**. Instead of clustering all samples together (class-independent), I performed clustering separately for each class. This has the advantage of creating **cleaner rules**, since each cluster is associated with only one class, improving both interpretability and classification performance but also creating much more rules, as shown in the results further below.

Each cluster center produced Gaussian membership functions on the input features, and the output membership functions were set as constant singletons, representing the class labels. This is important in classification tasks, since the target output is categorical and not continuous.

Grid Search and Cross-Validation

The system has two main hyperparameters:

- The **number of selected features** (from ReliefF ranking).
- The **cluster radius** of the subtractive clustering.

To find the best configuration, I performed a **grid search** combined with 5-fold cross-validation on the training set, each for 250 epochs. The grid included:

$$\text{Number of features: } \{4, 8, 12, 16, 20\},$$

$$\text{Cluster radius: } \{0.4, 0.5, 0.6, 0.7, 0.8\}$$

During initial trials, I also tested smaller radii such as 0.3 and below. However, when the number of features increased, the number of rules exploded dramatically, leading to extremely slow training times (in some cases my laptop could not handle it). For this reason, the final grid search was limited to radii between 0.4 and 0.8.

For each grid combination, a fuzzy system was trained and validated using 5-fold cross-validation, and the average error was recorded. This way, we can estimate how well each parameter set generalized to unseen data.

Final Model Training and Evaluation

After completing the grid search, the best parameter set was selected as the one with the lowest mean cross-validation error. Using these parameters, I trained a final fuzzy inference system on the combined training and validation sets for 500 epochs.

The model was then tested on the independent test set. To evaluate its performance, I computed:

- **Overall Accuracy (OA)** – the percentage of correctly classified samples.
- **Producer's Accuracy (PA)** and **User's Accuracy (UA)** – precision and recall for each class.
- **Cohen's Kappa coefficient** – which measures agreement beyond chance.
- **Confusion Matrices** – both raw and row-normalized.

Visualizations and Analysis

To better understand the system behavior, several visualizations were generated:

- **Membership Functions** before and after training, showing how the fuzzy sets adapted.

- **Learning curves** for training and validation errors, to check for overfitting.
- **Confusion matrices**, both standard and normalized.
- **Error plots** (continuous vs. rounded predictions).
- **True vs Predicted subplots**, where predicted labels were slightly *offset* for visibility. This ensures that true (blue) and predicted (red) points do not overlap, making discrepancies between the two clearer.
- **Grid search plots** relating the number of rules, selected features, and cross-validation error.
- **t-SNE projections** to visualize how separable the classes are in both the full and reduced feature spaces.

These steps ensured that the final fuzzy model was not only optimized for accuracy, but also interpretable and robust against overfitting.

Detailed code can be found in the `case_2.m` *file.*

PERFORMANCE METRICS AND PLOTS

Next, the results of the final ANFIS model are presented:

A. Grid search results

From the **GridSearch_Results.csv** we have:

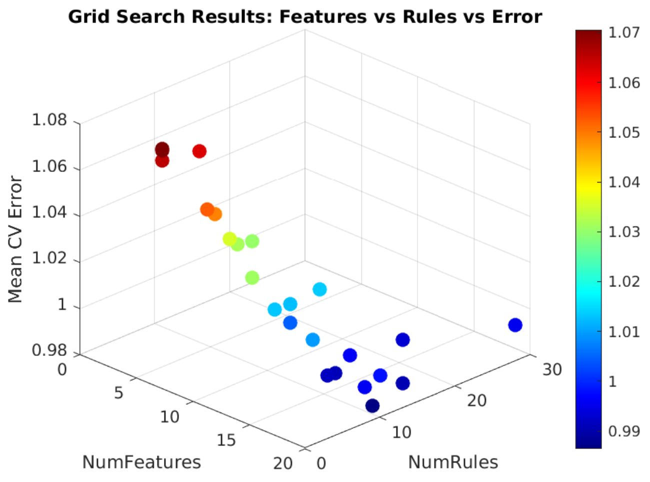


Fig. 28: Grid Search Results: Features vs Rules vs Mean CV MSE

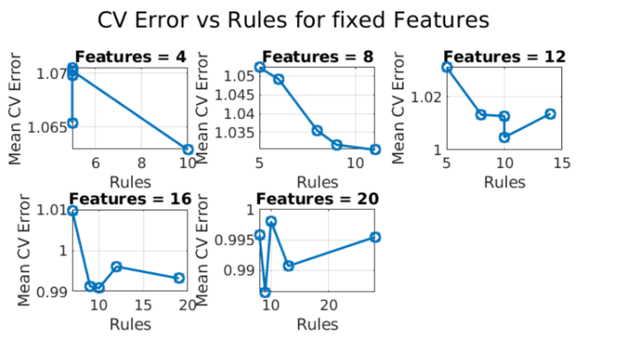


Fig. 29: Grid Search Results: Mean CV MSE vs num of Rules for fixed Features

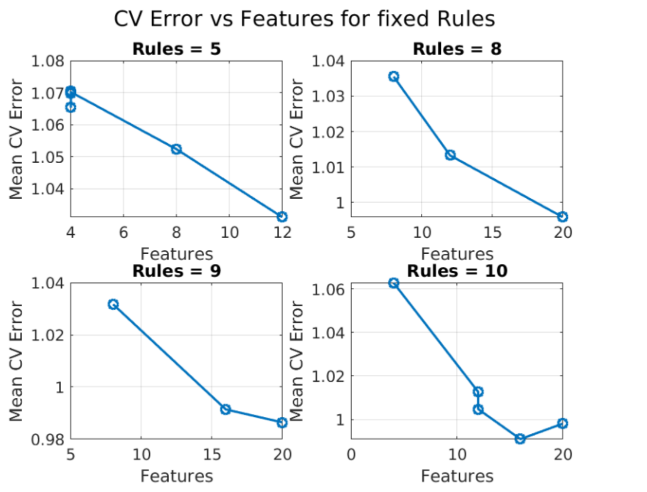


Fig. 30: Grid Search Results: Mean CV MSE vs num of Features for fixed Rules (points ≥ 2)

NumFeatures	ClusterRadius	NumRules	MeanCV_Error
4	0.4	10	1.0629
4	0.5	5	1.0654
4	0.6	5	1.0698
4	0.7	5	1.0705
4	0.8	5	1.0702
8	0.4	11	1.0304
8	0.5	9	1.0318
8	0.6	8	1.0355
8	0.7	6	1.0492
8	0.8	5	1.0523
12	0.4	14	1.0136
12	0.5	10	1.0127
12	0.6	10	1.0048
12	0.7	8	1.0133
12	0.8	5	1.0310
16	0.4	19	0.9933
16	0.5	12	0.9962
16	0.6	10	0.9910
16	0.7	9	0.9914
16	0.8	7	1.0096
20	0.4	28	0.9955
20	0.5	13	0.9907
20	0.6	10	0.9981
20	0.7	9	0.9864
20	0.8	8	0.9958

TABLE XII: Grid search results for different numbers of features and cluster influence ranges.

The grid search shows how the number of selected features and the clustering radius (r_a) together influence model complexity and error. In general, very small radii (e.g., $r_a = 0.4$) created many rules, which increased complexity without consistently improving performance. Larger radii ($r_a \geq 0.8$) reduced the number of rules substantially but often led to higher errors. The best results were obtained for moderate radii between 0.5 and 0.7, where the balance between accuracy and complexity is optimal. Among all configurations, the lowest mean cross-validation error was achieved with 20 features and $r_a = 0.7$ (MCVE ≈ 0.9864) using 9 rules. This indicates that the model performs best when using a larger subset of informative features, while keeping the clustering radius at an intermediate value to avoid both overfitting and oversimplification. From the grid search results, the following conclusions can be drawn:

- **Number of features:** Using only a small number of features (4–8) generally leads to higher cv errors. Larger subsets (12–20 features) typically achieve better performance, showing the benefit of including more informative features.
- **Clustering radius (r_a):** Very small radii ($r_a = 0.4$) create many rules without consistently improving accuracy, while very large radii ($r_a \geq 0.8$) oversimplify the model and lead to worse errors. Radii below 0.4 were also briefly tested, but the training time became exceptionally long due to the extremely large number of rules generated. For example, using 20 features with $r_a = 0.3$ produced approximately 100+ rules, and 24 features with $r_a = 0.3$ produced 800+ rules, which made these cases impractical to pursue. The best results are therefore obtained at intermediate radii ($r_a = 0.5$–0.7).
- **Best configuration:** The lowest mean cross-validation error (MCVE ≈ 0.9864) was achieved with 20 features and $r_a = 0.7$, using 9 rules. However, other settings such as 16 features with $r_a = 0.6$ (MCVE ≈ 0.9910) or 20 features with $r_a = 0.5$ (MCVE ≈ 0.9907) performed nearly as well. This indicates that there are multiple good configurations, and the choice may depend on whether simplicity (fewer features or rules) is preferred over the absolute lowest error. In fact, the trade-off is evident when comparing 16 features with 10 rules versus 20 features with 9 rules: one uses fewer features but a comparable number of rules, while the other uses more features but maintains high accuracy with slightly fewer rules. Both achieve nearly identical errors, highlighting that accuracy can often be maintained by balancing model complexity in terms of features and rules.

Overall, the results emphasize that using 16–20 informative features with an intermediate clustering radius ($r_a = 0.5$–0.7) yields the best trade-off between model complexity and accuracy.

For this project, I selected the model with **20 features and $r_a = 0.7$ (9 rules)** as the *best* configuration. The performance of this model is presented below.

B. Performance and efficiency of the final model

Below are the learning curves of the final TSK model with the optimal combination of features and rules:

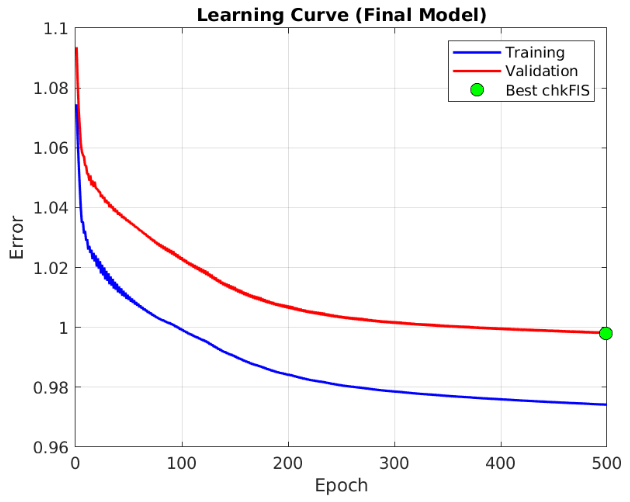


Fig. 31: Learning curves of the final TSK model

Next the requested performance metrics of the final model, as shown in **FinalModel_Metrics.csv**, are presented:

Metric	Value
Overall Accuracy (OA)	0.399
Cohen's Kappa	0.271
Producer Accuracy Class 1	0.929
User Accuracy Class 1	0.687
Producer Accuracy Class 2	0.159
User Accuracy Class 2	0.074
Producer Accuracy Class 3	0.334
User Accuracy Class 3	0.648
Producer Accuracy Class 4	0.314
User Accuracy Class 4	0.541
Producer Accuracy Class 5	0.323
User Accuracy Class 5	0.043

TABLE XIII: Performance metrics of the final model.

Regarding the performance of the final model, the prediction errors on the test set as well as the true versus predicted values are shown below. The errors can be found in two plots: the first shows the *continuous* prediction errors, which are the raw `ANFIS` outputs before rounding, giving an idea of the model's general tendency and bias. The second plot shows the *discrete* prediction errors, calculated after rounding the predictions to the nearest integer class, which reflects the actual classification performance. For the true versus predicted values, I used multiple subplots with the predicted values slightly offset, making it easier to see how well the model matches the ground truth across different parts of the test set. Both the blue and red dots ideally should align.

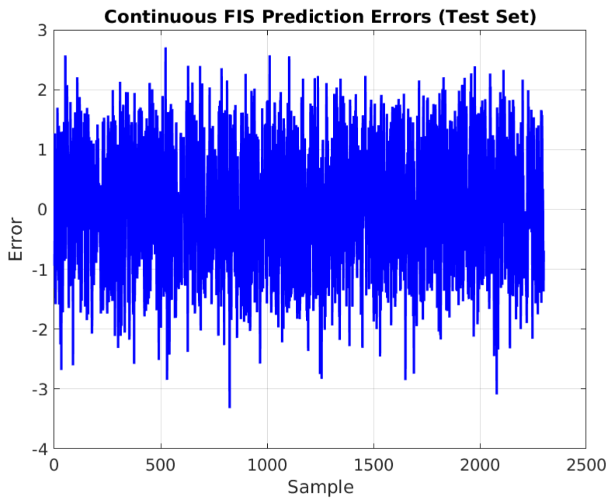


Fig. 32: Prediction errors (Continuous)

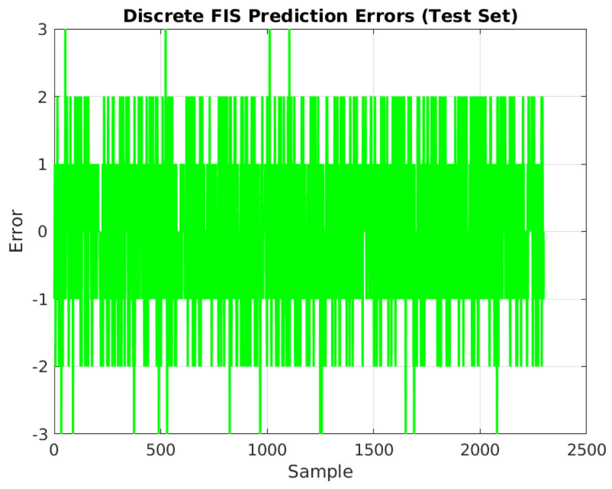


Fig. 33: Prediction errors (Discrete)

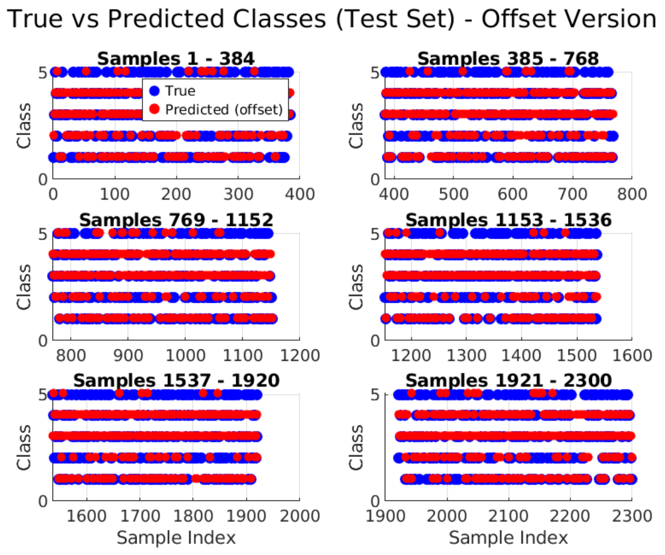


Fig. 34: True vs Predicted values (offset)

Lastly, confusion matrices (standard and normalized) are given below:

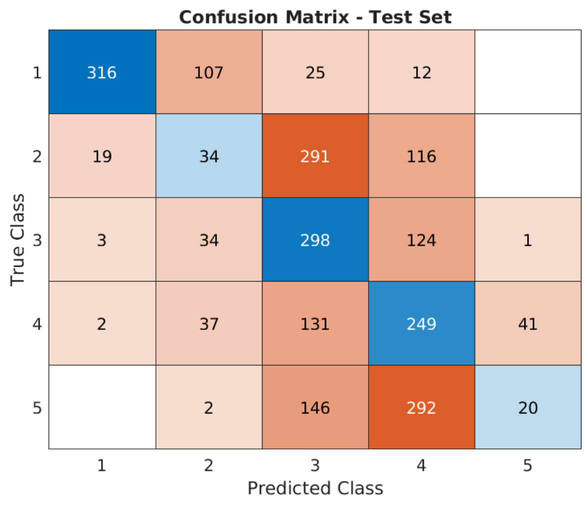


Fig. 35: Standard confusion matrix

True Class	Pred 1	Pred 2	Pred 3	Pred 4	Pred 5
1	316	107	25	12	0
2	19	34	291	116	0
3	3	34	298	124	1
4	2	37	131	249	41
5	0	2	146	292	20

TABLE XIV: Confusion Matrix of the final model.

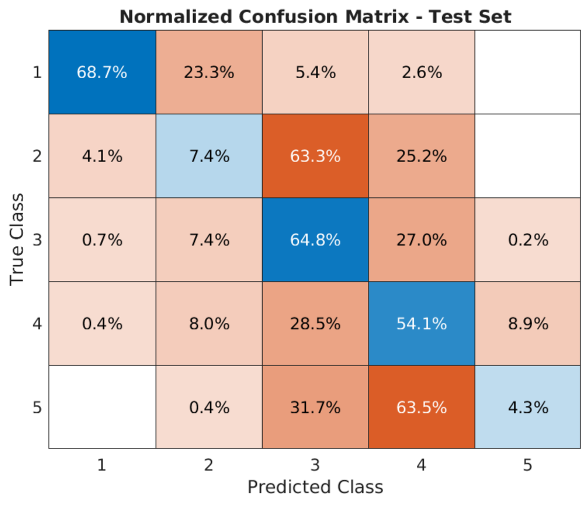


Fig. 36: Normalized confusion matrix

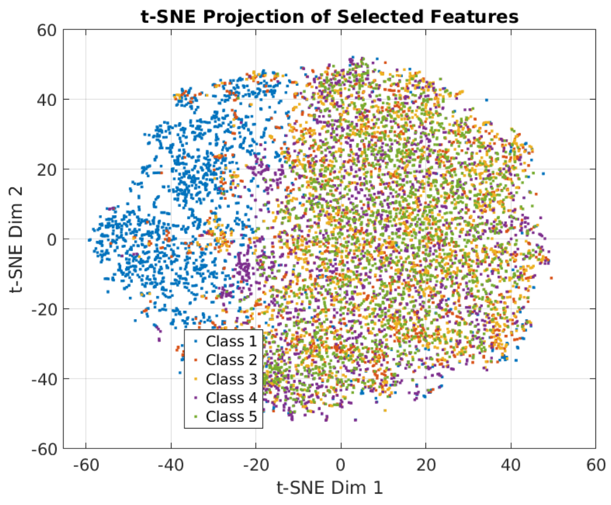


Fig. 38: t-SNE projections of the featured data

To better understand the structure of the dataset and the effect of feature selection, I visualized the samples using t-SNE. The plots show a 2D projection of the data, highlighting how well the different classes are separated in both the full feature space and the subset of selected features.

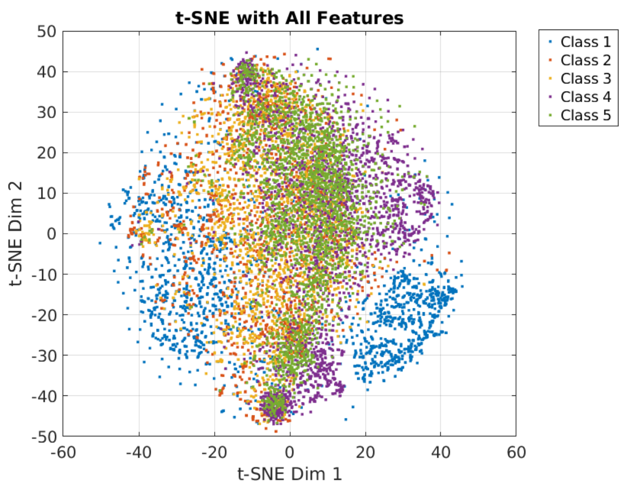


Fig. 37: t-SNE projections of the all data

Learning Curves and Generalization

The learning curve (Figure 31) gives a clear view of how the model behaved during training. The training error drops steadily and ends up below 0.98, while the validation error levels off around 1.00. The gap between the two curves shows that the model is **overfitting** the training data. In other words, it learns the training set quite well but struggles to generalize that good to unseen data. However, since the validation error does not increase sharply, this overfitting is not extreme, but it still limits the model's final performance.

Confusion Matrix and Class-wise Behavior

Looking at the confusion matrices (Figures 35 and 36), it becomes clear that **Class 1** is the easiest to recognize. This is backed up by its high producer accuracy (92.9%). The t-SNE plot (Figure 38) also shows that Class 1 forms a neat and fairly isolated cluster, which makes it easier for the fuzzy model to classify. The situation is quite different for Classes 2, 4, and 5, where there is a lot of overlap with other classes in feature space. This overlap leads to heavy confusion in the predictions. For instance, Class 2 has a very low user accuracy (7.4%), meaning the model almost never predicts it correctly. Similarly, Class 5 is mostly misclassified as Class 3 or 4, which explains its poor performance. Class 3 does somewhat better, with decent recall (64.8%), but it still gets mixed up often with Classes 2 and 4.

True vs. Predicted Behavior

The plots of true versus predicted labels give another angle on this. In some parts of the test set, the red predictions line up with the blue true labels, but for most samples there is little overlap. This shows that apart from Class 1, the model finds it difficult to produce stable predictions. It can spot one strong cluster well, but the rest of the classes are too entangled and end up confused with one another.

Quantitative Metrics

The numbers in Table XIII confirm what we see in the plots. The overall accuracy (OA) is 39.9%, which is only slightly better than random guessing in a five-class task. Cohen's Kappa is 0.271, which points to only weak agreement with the true labels. Once again, Class 1 clearly stands out with strong results, while the other classes lag far behind. The worst cases are Classes 2 and 5, which have precision values below 10%, showing that the model can barely distinguish them at all.

Overall Assessment

To sum up, the fuzzy system works well for one clearly separated class (Class 1), but it struggles badly with the rest. The t-SNE visualization makes it obvious why: Classes 2–5 are heavily mixed together in feature space, and the fuzzy rules are not powerful enough to untangle them. This leads to modest overall accuracy and low Kappa values. To improve, one could try different feature extraction methods (like PCA or neural embeddings), tune the membership functions more adaptively, or even combine fuzzy systems with other classifiers to get more expressive decision boundaries.

C. Membership Functions (MFs) of the Final Model

Below are the initial membership functions (MFs) of the 6 linguistic values (clusters) for the 30 input features of the final TSK model (6 inputs have been selected for illustration):

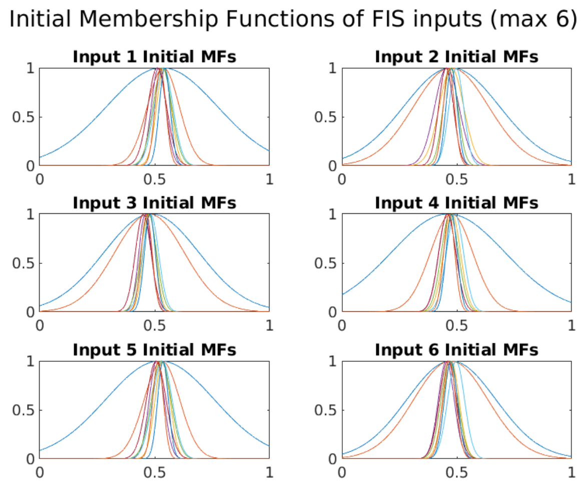


Fig. 39: Some of the Initial Membership Functions of the FIS inputs

Next, the final forms of the above membership functions (MFs) are shown:

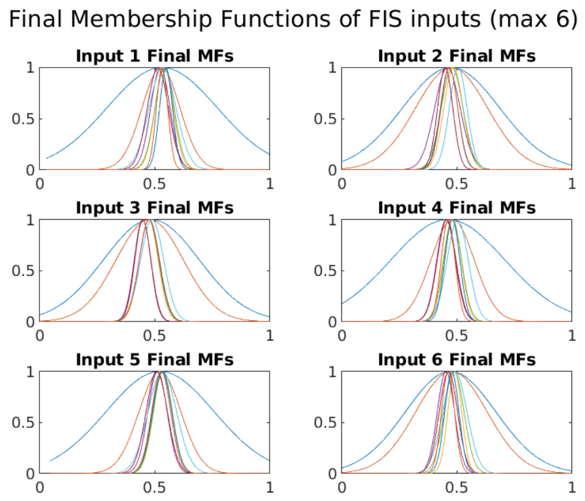


Fig. 40: Some of the Final Membership Functions of the FIS inputs

Discussion on Membership Function Overlap and Rule Activation

From the plots of the membership functions (MFs), it is clear that for each input all the Gaussians are centered around the same point (close to 0.5), and only their spreads differ. This means that the overlap between fuzzy sets is very high: in the central region almost all MFs have non-zero activation. The narrow MFs respond only to inputs very close to the center, while the wider MFs remain active across a larger part of the domain. As a result, several rules can be activated at the same time, especially for values near the center of the input space.

This high degree of overlap does not necessarily harm performance. On the contrary, it allows the system to capture both fine-grained and coarse information about the inputs, since the narrow Gaussians provide sensitivity while the wide ones ensure smoother generalization. However, it also means that the rules are not tied to clearly distinct clusters, but instead share much of the same input region. This is consistent with the relatively large clustering radius (0.7), which produced only 9 rules overall. The trade-off is that the model remains simple and smooth, but with less class separation in the fuzzy space. Still, the achieved accuracy shows that this balance between overlap and generalization was sufficient for effective classification.

Concluding Observations and Comments

For the `epileptic seizure recognition` dataset, the fuzzy classifier with 20 selected features and a clustering radius of 0.7 ended up with a very compact model of just 9 rules. The overall accuracy was 39.9% with a Cohen's Kappa of 0.271, which means the model is only slightly better than random guessing in this five-class problem. As shown in the confusion matrices and the t-SNE plots, Class 1 is recognized well, but the other classes are much more mixed and the model cannot separate them reliably.

Looking at the membership functions, most of them share the same center and only differ in spread, which means there is a lot of overlap. This overlap makes several rules fire at once, which helps smooth things out but also makes it harder to create clear boundaries between the confusing classes. The relatively large clustering radius explains why the rule base is so small. While this keeps the model efficient and interpretable, it also limits its expressiveness. Allowing more rules — either by reducing the clustering radius or by including more features — could give the system more flexibility to capture finer class boundaries and potentially improve accuracy.

It is also worth noting how effective the clustering approach is compared to a traditional grid partitioning. With 20 features, even using only 2 fuzzy sets per input would require $2^{20} = 1,048,576$ rules, while 3 fuzzy sets per input would explode to $3^{20} \approx 3.5$ billion rules. In contrast, subtractive clustering produced just 9 rules, making the problem computationally feasible and the model interpretable, even if accuracy wasn't that great.

In the end, the class-dependent fuzzy system shows both strengths and weaknesses. It can handle one well-defined class very well, but it struggles with classes that overlap strongly in feature space. With only 9 rules, the model is extremely compact, which makes it simple and efficient but also limits its ability to capture finer decision boundaries. It is possible that allowing more rules (through a smaller clustering radius or by including more features) could improve accuracy by giving the system more flexibility to distinguish the difficult classes. At the same time, the results suggest that for this kind of complex biomedical data, fuzzy rules alone may not be enough, and that combining them with other feature extraction or classification methods could give better overall performance.

Finally, for completeness, I listed the 20 most important features selected by `ReliefF` for training the final model, as shown in **Selected_Features.csv**:

44	84	101	51	43	87	42	100	143	85
83	82	35	48	99	88	102	52	50	168

TABLE XV: Indices of the 20 most important features selected by ReliefF ($K = 50$).