Fuzzy Systems and Computational Intelligence: TSK Regression Models

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

***Abstract*—This project investigates the application of Takagi–Sugeno–Kang (TSK) fuzzy models to regression problems, focusing on their ability to model multivariate nonlinear functions. Two datasets from the UCI repository are used. The first, the Airfoil Self-Noise dataset, is employed for introductory training, evaluation, and interpretation of TSK models with varying membership functions and output forms. The second, the Superconductivity dataset, provides a high-dimensional benchmark to explore feature selection, clustering-based partitioning, and hyperparameter optimization through cross-validation. Performance is evaluated using metrics such as RMSE, NMSE, NDEI, and R^2. The results highlight the trade-off between model complexity and generalization, demonstrating both the strengths and limitations of TSK fuzzy regression in real-world data modeling.**

***Index Terms*—TSK fuzzy model, regression, nonlinear systems, feature selection, clustering, cross validation, performance evaluation.**

Two case studies are carried out:

Airfoil Self-Noise Dataset

The first dataset contains 1503 samples with 6 features. The data are partitioned into training (60%), validation (20%), and test (20%) subsets. Four TSK models are trained, differing in membership function count (2 or 3 per input) and output type (singleton vs. polynomial). Membership functions are initialized with bell-shaped distributions ensuring 0.5 overlap. Model training uses a hybrid approach: membership parameters via backpropagation, and consequent parameters via least-squares estimation. Models are evaluated using RMSE, NMSE, NDEI, and R^2. The best model is selected according to validation error. Results are analyzed with learning curves, fuzzy set visualizations, and prediction error plots.

	Πλήθος συναρτήσεων συμμετοχής	Μορφή εξόδου
TSK_model_1	2	Singleton
TSK_model_2	3	Singleton
TSK_model_3	2	Polynomial
TSK_model_4	3	Polynomial

Fig. 1: Classification of models for training

Superconductivity Dataset

The second dataset contains 21263 samples with 81 features, posing a high-dimensional regression challenge. To address rule explosion, dimensionality reduction via feature selection (Relief) and rule reduction via subtractive clustering are applied. Grid search with 5-fold cross-validation identifies optimal parameters: number of selected features and cluster radius. The final TSK model is trained on optimal configurations and evaluated on the test set. Prediction accuracy, learning curves, and fuzzy partitions are presented, alongside comparisons with grid partitioning approaches. The study highlights how clustering and feature selection balance accuracy and complexity in large-scale TSK regression.

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

I. Application on the Smallest (First) Dataset

A. *Methodology Analysis*

For all models, **the first five columns** of the dataset were chosen as **input** variables because they describe the conditions affecting the airfoil, while the **sixth column** was chosen as the **output** variable to be predicted, corresponding to the sound pressure level.

The methodology used to develop and evaluate all **four TSK fuzzy models** follows a consistent workflow. First, the `Airfoil Self-Noise` dataset from the UCI repository is loaded. **Duplicate rows are removed** to avoid redundancy, and all inputs and the output are **normalized** to the range $[0, 1]$ using **min-max normalization**. This ensures that all variables are on the same scale and helps the model train more efficiently.

For each model, we perform **multiple independent runs** (30 runs) to account for the randomness in training and to obtain reliable performance statistics. In each run, a random seed is set for reproducibility, and the **dataset is shuffled** and **split into training (60%), validation (20%), and test (20%) sets**. The training data is used to create the initial fuzzy inference system (FIS) through grid partitioning, specifying the number of membership functions (MFs) for each input and the type of output function (either constant for singleton or linear for polynomial) depending on the model. Also, **bell-shaped membership functions** are used for the inputs to capture nonlinear relationships.

The FIS is then **optimized using ANFIS**, which combines gradient descent for adjusting the input MFs and least-squares estimation for the output parameters. The model is trained over a fixed number of epochs (200), while the validation set is used to monitor overfitting and select the best FIS. After training, the optimized FIS is tested on the unseen test data. To make the performance metrics interpretable in the original units of the dataset, the predicted outputs and the true outputs are **denormalized** back to their original scale before computing metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Normalized MSE (NMSE), Normalized Root MSE (NDEI), and the coefficient of determination (R^2). This allows the errors to be understood in terms of the actual noise levels of the airfoil data, rather than in the normalized $[0, 1]$ range, providing a more meaningful evaluation of model accuracy.

Finally, the **representative run** is chosen as the one whose R^2 is closest to the mean R^2 over all runs. For this run, we generate plots of the initial and trained membership functions, the learning curves of training and validation errors, and the prediction errors on the test set. These plots, along with the denormalized performance metrics, give a complete view of how well the model captures the underlying patterns in the data. Below we can see the results for each of the four models.

In the learning curve plot, the **green dot** represents the **epoch** where the ANFIS model reached the **lowest validation error**. During training, the model's performance is tracked on both the training and validation sets. While the training error typically decreases with each epoch, the validation error may reach a minimum before the training error fully converges. The green dot marks this point, showing where the model generalizes best to unseen data. The FIS at this epoch, stored in `chkFIS` in my code, represents the model **before overfitting starts**. `ANFIS` does this automatically by keeping track of the validation error and saving the FIS with the smallest value. Using `chkFIS` for testing makes sure we evaluate the model that performs best on new data, while minimizing the risk of overfitting.

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 has **2** Membership Functions with a **Singleton** output. First, we present the initial and final shapes of the membership functions for the fuzzy input variables:

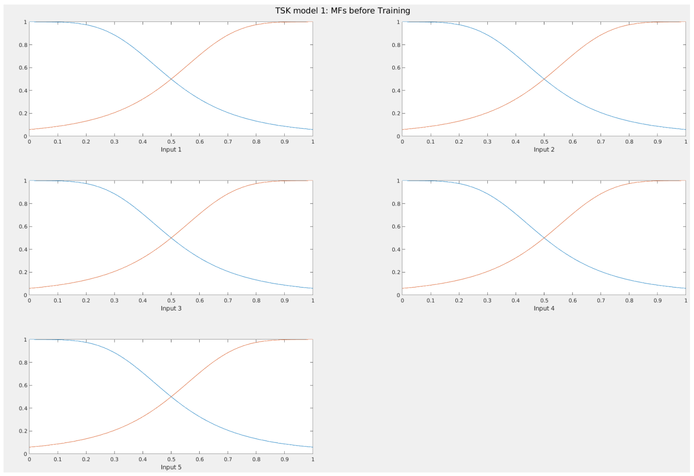


Fig. 2: MFs **BEFORE** training:TSK Model 1

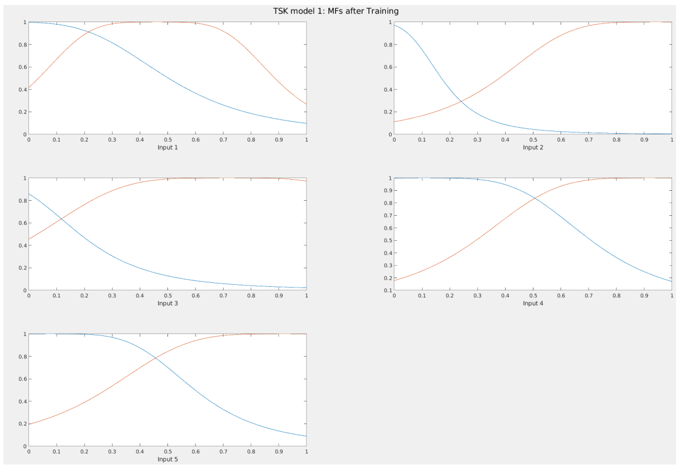


Fig. 3: MFs **AFTER** training: TSK Model 1

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

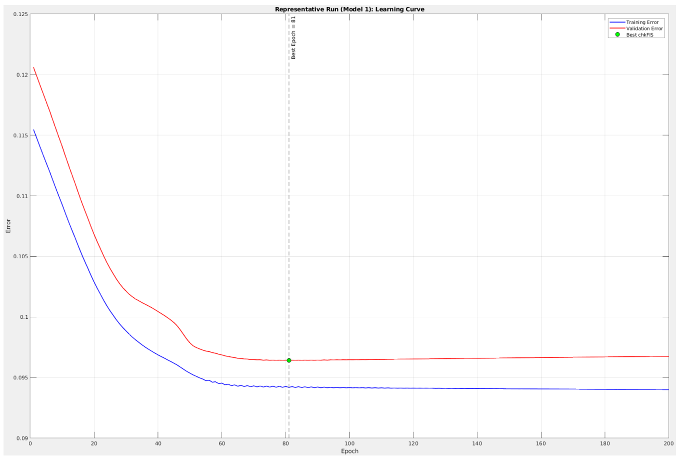


Fig. 4: Learning curves: TSK Model 1

Finally, the requested performance metrics of the model are presented:

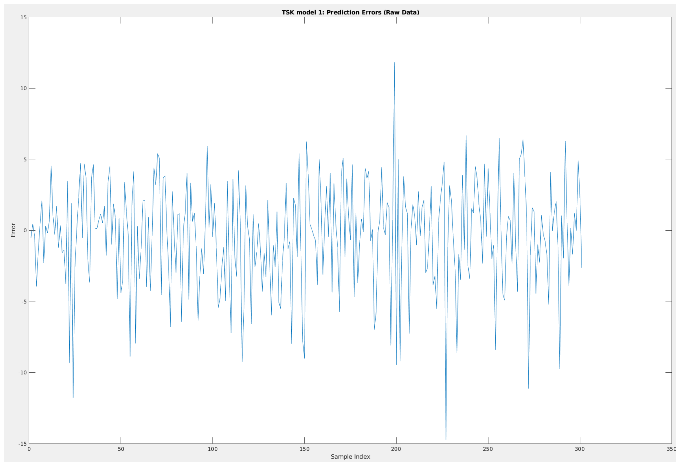
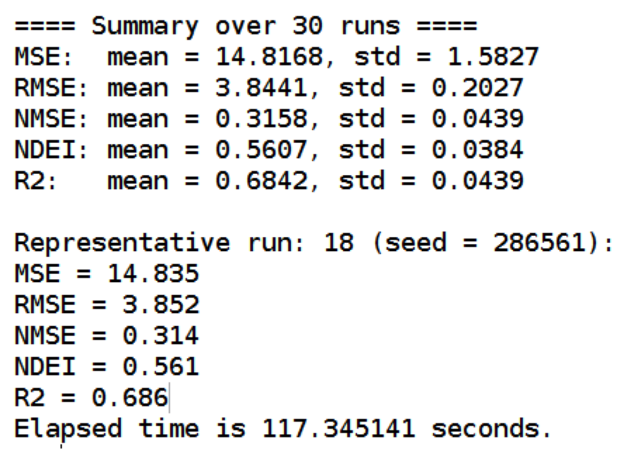


Fig. 5: Prediction errors: TSK Model 1



```
==== Summary over 30 runs ====
MSE:  mean = 14.8168, std = 1.5827
RMSE: mean = 3.8441, std = 0.2027
NMSE: mean = 0.3158, std = 0.0439
NDEI: mean = 0.5607, std = 0.0384
R2:   mean = 0.6842, std = 0.0439

Representative run: 18 (seed = 286561):
MSE = 14.835
RMSE = 3.852
NMSE = 0.314
NDEI = 0.561
R2 = 0.686
Elapsed time is 117.345141 seconds.
```

Fig. 6: Final Metrics (Mean Values): TSK Model 1

Metric	**Mean**	**Standard Deviation**
MSE	14.8168	1.5827
RMSE	3.8441	0.2027
NMSE	0.3158	0.0439
NDEI	0.5607	0.0384
R^2	0.6842	0.0439
Time for 30 iterations: 117.345141(sec)		

TABLE I: Performance Metrics of the Model: TSK Model 1 in tabular form

The TSK Model 1 on the Airfoil Self-Noise dataset shows decent performance, but there is definitely room for improvement. With an R^2 of 0.6842, the model captures about 68% of the data variance, which leaves a lot unexplained. The NMSE of 0.3158 and NDEI of 0.5607 indicate that prediction errors are still quite noticeable compared to the overall variability. As shown in Fig. 5, where the errors are centered around zero and show relatively small variation considering the scale of the output, the model can make reasonable predictions. However, these results fall short of what is typically achievable on this dataset, where R^2 values around 0.85 or higher are possible with more advanced approaches. Lastly, the model doesn't seem to show any clear signs of overfitting. Overall, the current model works but could benefit from further tuning or trying alternative methods to improve accuracy.

TSK Model 2

This Model has **3** Membership Functions with a **Singleton** output. First, we present the initial and final shapes of the membership functions for the fuzzy input variables:

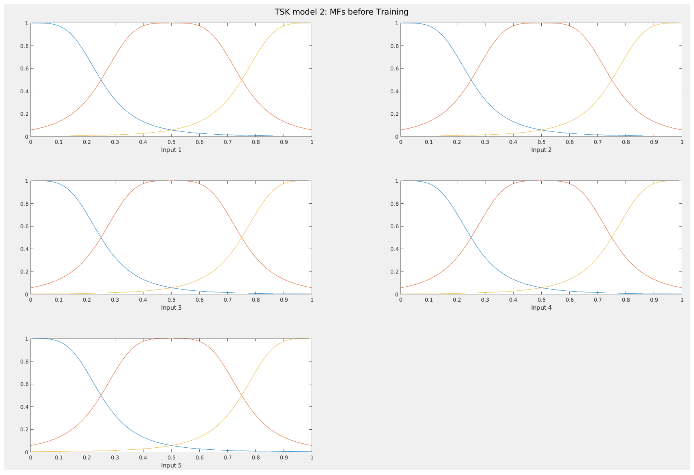


Fig. 7: MFs **BEFORE** training:TSK Model 2

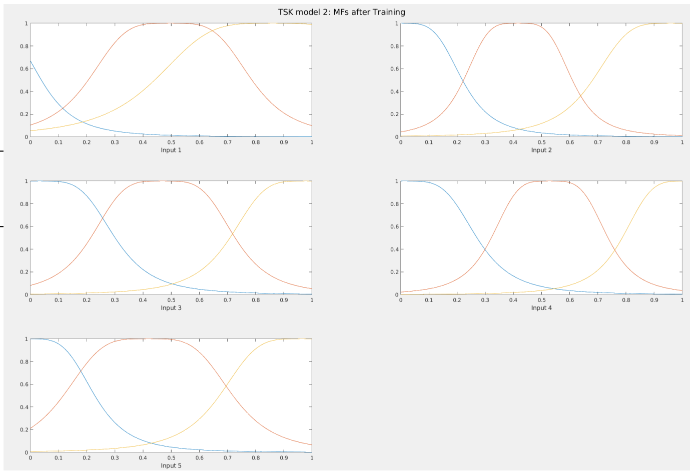


Fig. 8: MFs **AFTER** training: TSK Model 2

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

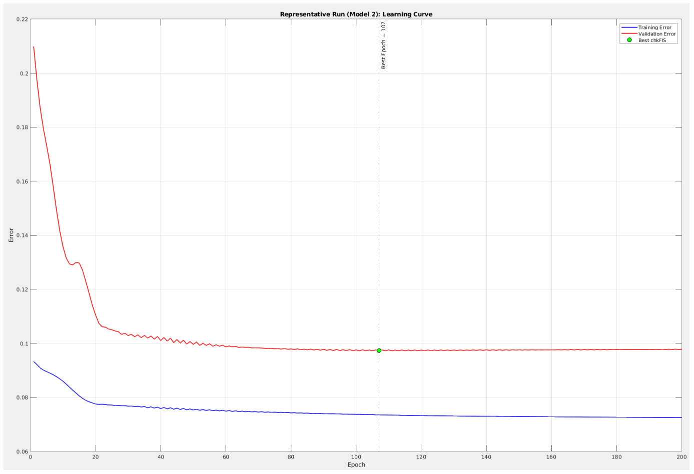


Fig. 9: Learning curves: TSK Model 2

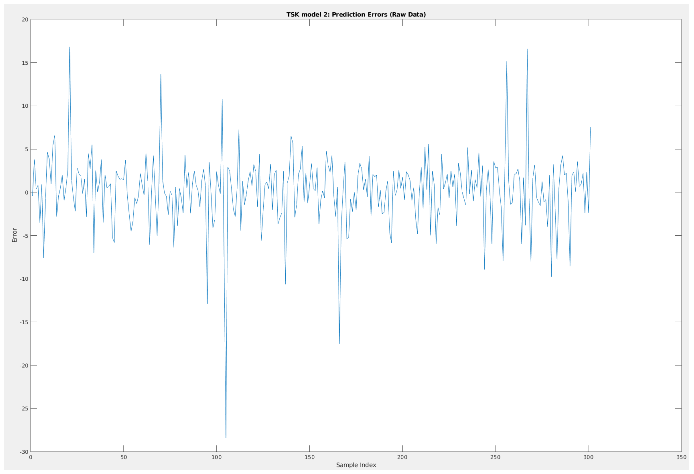
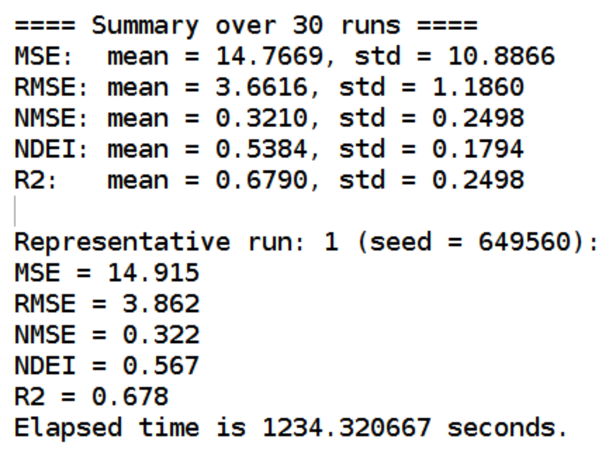


Fig. 10: Prediction errors: TSK Model 2

Finally, the requested performance metrics of the model are presented:



```
==== Summary over 30 runs ====
MSE:  mean = 14.7669, std = 10.8866
RMSE: mean = 3.6616, std = 1.1860
NMSE: mean = 0.3210, std = 0.2498
NDEI: mean = 0.5384, std = 0.1794
R2:   mean = 0.6790, std = 0.2498

Representative run: 1 (seed = 649560):
MSE = 14.915
RMSE = 3.862
NMSE = 0.322
NDEI = 0.567
R2 = 0.678
Elapsed time is 1234.320667 seconds.
```

Fig. 11: Final Metrics (Mean Values): TSK Model 2

Metric	**Mean**	**Standard Deviation**
MSE	14.7669	10.8866
RMSE	3.6616	1.1860
NMSE	0.3210	0.2498
NDEI	0.5384	0.1794
R^2	0.6790	0.2498
Time for 30 iterations: 1234.320667(sec)		

TABLE II: Performance Metrics of the Model: TSK Model 2 in tabular form

The TSK Model 2 shows performance that is broadly comparable to Model 1, but with some clear differences. The mean R^2 is 0.6790, only slightly lower than Model 1's 0.6842, which means both models capture a similar amount of variance in the data. The NMSE (0.3210 vs. 0.3158) and NDEI (0.5384 vs. 0.5607) also suggest that, on average, their predictive accuracy is about the same. The big difference, however, is in the **standard deviations**, which are much larger for Model 2. This is especially obvious for MSE (10.8866 vs. 1.5827) and R^2 (0.2498 vs. 0.0439), showing that while Model 2 can sometimes perform well, its results vary a lot more across different runs and random splits. As shown in Fig. 10, its prediction spread is clearly worse than that of Model 1. Like Model 1, there are no obvious signs of overfitting, but the training errors in Model 2 started off smaller and then decreased more slowly over the epochs, so the reduction in error (delta error) was less than in Model 1. Another point is that Model 2 needed more epochs to reach the "green dot," meaning ANFIS took longer to hit its lowest validation error compared to Model 1.

In other words, the prediction errors of Model 2 are less stable and can be much worse than those of Model 1. On top of that, Model 2 takes far longer to train (1234 seconds for 30 runs vs. 117 seconds for Model 1), which is due to the larger number of membership functions and the more complex FIS. Overall, increasing the number of MFs to 3 did not bring a consistent improvement in accuracy, and the higher variability makes Model 1 the more reliable choice for of the two, so far.

TSK MODEL 3

This Model has **2** Membership Functions with a **Polynomial** output. First, we present the initial and final shapes of the membership functions for the fuzzy input variables:

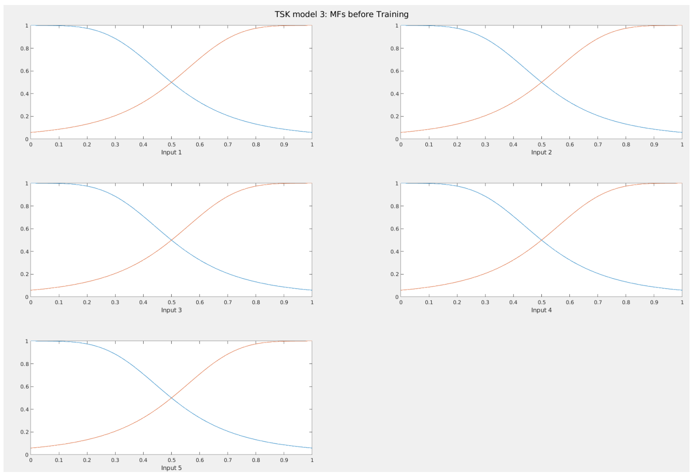


Fig. 12: MFs **BEFORE** training:TSK Model 3

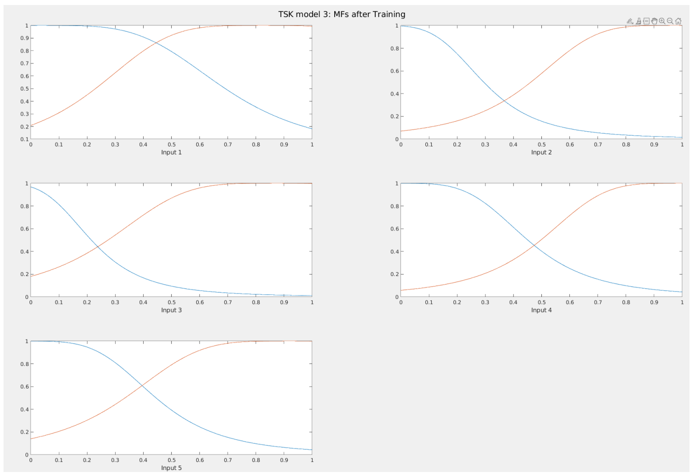


Fig. 13: MFs **AFTER** training: TSK Model 3

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

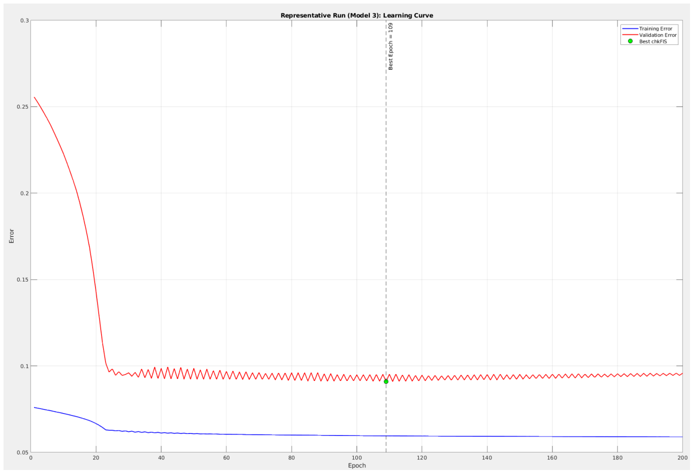


Fig. 14: Learning curves: TSK Model 3

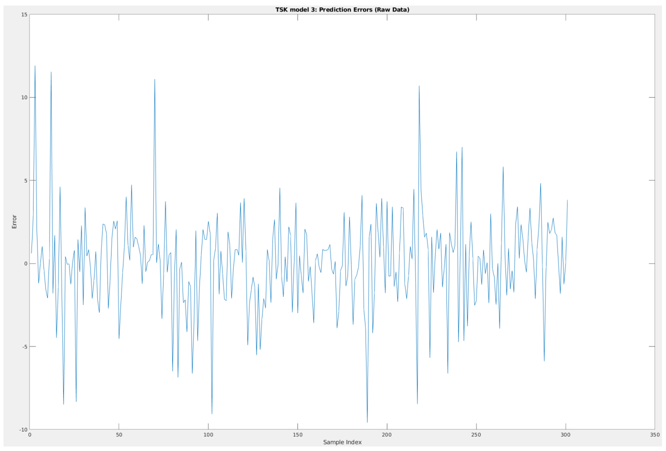
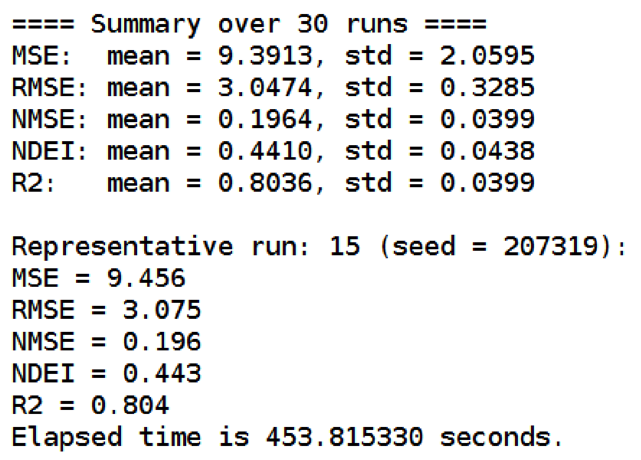


Fig. 15: Prediction errors: TSK Model 3

Finally, the requested performance metrics of the model are presented:



```
==== Summary over 30 runs ====
MSE:  mean = 9.3913, std = 2.0595
RMSE: mean = 3.0474, std = 0.3285
NMSE: mean = 0.1964, std = 0.0399
NDEI: mean = 0.4410, std = 0.0438
R2:   mean = 0.8036, std = 0.0399

Representative run: 15 (seed = 207319):
MSE = 9.456
RMSE = 3.075
NMSE = 0.196
NDEI = 0.443
R2 = 0.804
Elapsed time is 453.815330 seconds.
```

Fig. 16: Final Metrics (Mean Values): TSK Model 3

Metric	Mean	Standard Deviation
MSE	9.3913	2.0595
RMSE	3.0474	0.3285
NMSE	0.1964	0.0399
NDEI	0.4410	0.0438
R^2	0.8036	0.0399
Time for 30 iterations: 453.815330(sec)		

TABLE III: Performance Metrics of the Model: TSK Model 3 in tabular form

The TSK Model 3, which uses 2 MFs with polynomial outputs, performs noticeably better than both Model 1 and Model 2. The mean R^2 reaches 0.8036, a clear improvement over Model 1's 0.6842 and Model 2's 0.6790, showing that this model explains a larger proportion of the variance in the data. This is also reflected in the NMSE (0.1964 vs. ∼0.32 in the first two models) and NDEI (0.4410 vs. ∼0.55), which confirm that the predictions are not only more accurate on average, but also more consistent. The MSE and RMSE values are lower as well, with an average MSE of 9.3913 and RMSE of 3.0474, showing clear gains in error reduction compared to the earlier models.

Another important point is that the variability across runs is much smaller in Model 3 compared to Model 2. The standard deviation of R^2 is only 0.0399 (similar to Model 1's stability) and far below Model 2's 0.2498, which means Model 3 is both more accurate and more reliable. However, this improvement comes with a cost: the computation time is significantly higher (453.8 seconds for 30 runs), sitting between the very fast Model 1 (117 seconds) and the very slow Model 2 (1234 seconds). This trade-off seems reasonable, since Model 3 manages to combine the stability of Model 1 with a clear accuracy boost, without the extreme training cost of Model 2.

Looking at the learning curve (Fig. 14), there is no strong evidence of overfitting: after an initial drop, both training and validation errors stabilize at low values. The green dot marks the epoch where the validation error was lowest, and the chosen FIS (`chkFIS`) corresponds to this point, ensuring the best generalization. The small spikes in the validation error are normal fluctuations caused by stochastic updates during training, not signs of instability. Overall, the learning dynamics confirm that Model 3 is well-regularized and achieves a stable balance between training and validation performance.

In summary, introducing polynomial outputs in the TSK framework appears to be the most effective change so far, delivering higher accuracy and better consistency while keeping the training time manageable. Compared to simply increasing the number of MFs (as in Model 2), this approach is more efficient in terms of both training time and accuracy, ultimately leading to stronger predictive performance.

TSK MODEL 4

This Model has **3** Membership Functions with a **Polynomial** output. First, we present the initial and final shapes of the membership functions for the fuzzy input variables:

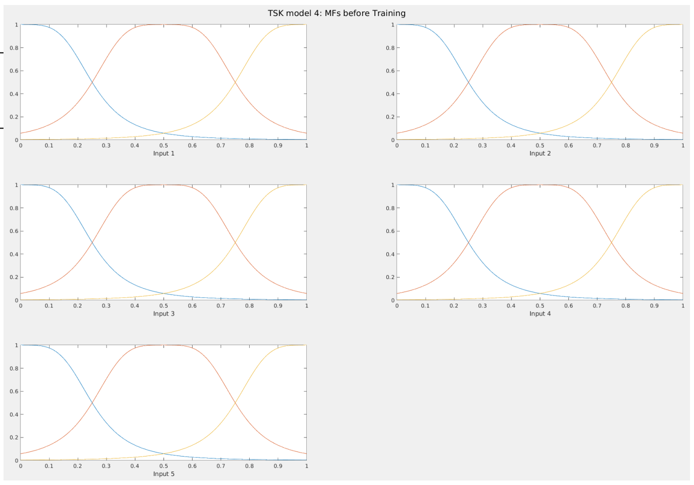


Fig. 17: MFs **BEFORE** training:TSK Model 4

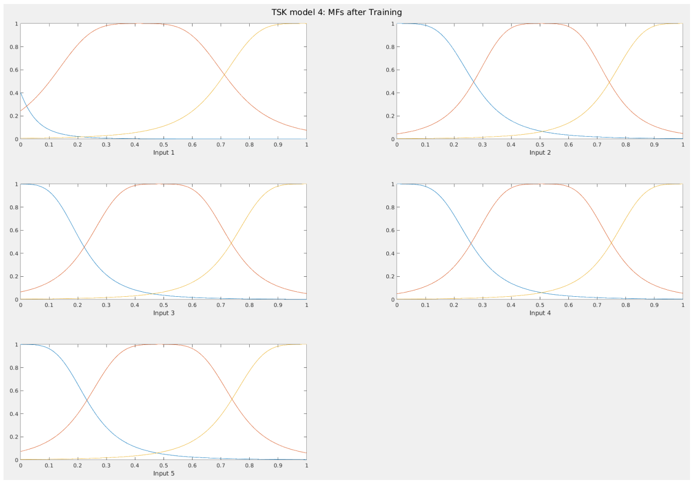


Fig. 18: MFs **AFTER** training: TSK Model 4

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

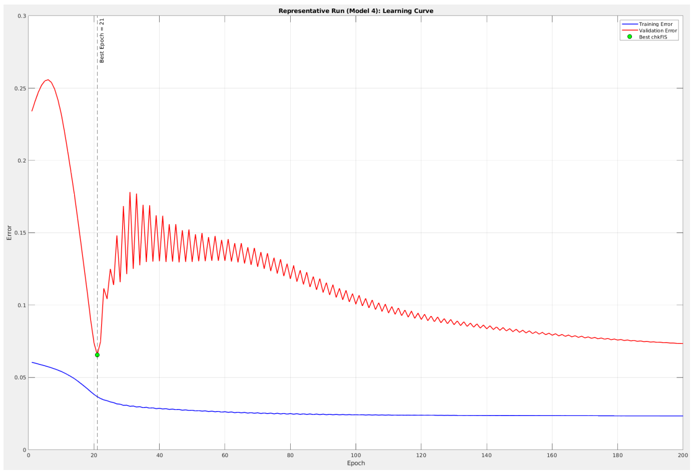


Fig. 19: Learning curves: TSK Model 4

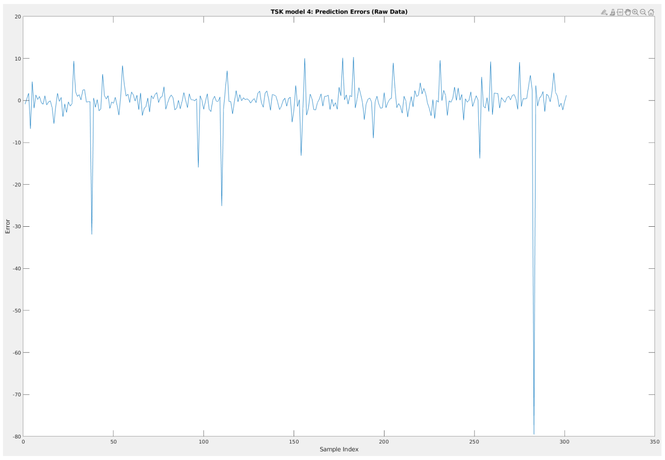
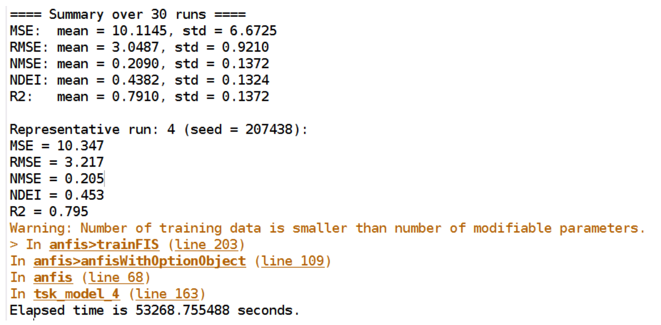


Fig. 20: Prediction errors: TSK Model 4

Finally, the requested performance metrics of the model are presented:



```
==== Summary over 30 runs ====
MSE:  mean = 10.1145, std = 6.6725
RMSE: mean = 3.0487, std = 0.9210
NMSE: mean = 0.2090, std = 0.1372
NDEI: mean = 0.4382, std = 0.1324
R2:   mean = 0.7910, std = 0.1372

Representative run: 4 (seed = 207438):
MSE = 10.347
RMSE = 3.217
NMSE = 0.205
NDEI = 0.453
R2 = 0.795
Warning: Number of training data is smaller than number of modifiable parameters.
> In anfis>trainFIS (line 203)
In anfis>anfisWithOptionObject (line 109)
In anfis (line 68)
In tsk_model_4 (line 163)
Elapsed time is 53268.755488 seconds.
```

Fig. 21: Final Metrics (Mean Values): TSK Model 4

Metric	**Mean**	**Standard Deviation**
MSE	10.1145	6.6725
RMSE	3.0487	0.9210
NMSE	0.2090	0.1372
NDEI	0.4382	0.1324
R^2	0.7910	0.1372
Time for 30 iterations: 53268.755488(sec)		

TABLE IV: Performance Metrics of the Model: TSK Model 4 in tabular form

According to the performance metrics, the model achieved a mean squared error (MSE) of 10.1145 with a standard deviation of 6.6725, corresponding to a root mean squared error (RMSE) of 3.0487. The normalized mean squared error (NMSE) was 0.2090, and the normalized degree of error index (NDEI) was 0.4382. Additionally, the coefficient of determination (R^2) reached 0.7910, which suggests that approximately 79% of the variance in the data was explained by the model. While this represents a reasonably good fit, the relatively large standard deviations in the metrics point to some instability in the model's generalization capability across different runs. Furthermore, the computational cost is very high, as the training time for 30 iterations was approximately 53268 seconds (almost 15 hours), which is significantly larger compared to the other models.

The learning curve (Figure 19) provides further insight into the training dynamics of Model 4. The training error (blue curve) decreases smoothly and stabilizes at a low level, indicating that the model learns the training set effectively. However, the validation error (red curve) exhibits oscillations after the 20th epoch, where the best validation performance was recorded. These oscillations suggest that the model struggles with stability during training, possibly due to overfitting and sensitivity to hyperparameter choices. The gap between the training and validation errors further supports the presence of overfitting: the model fits the training data well but does not generalize as effectively to unseen data.

The prediction error plot (Figure 20) illustrates the residuals across the samples. Most errors remain within a relatively small range around zero, which is consistent with the acceptable RMSE value. However, there are several large negative spikes, indicating that the model occasionally produces significant mispredictions on certain samples.

In conclusion, while Model 4 achieves competitive accuracy ($R^2 = 0.7910$) and captures the underlying data patterns effectively, it suffers from instability, overfitting tendencies, and high computational cost. Compared to the other TSK models, Model 4 does not provide the best trade-off between performance and efficiency, especially considering its training time.

Concluding Analysis

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

	MSE	RMSE	NMSE	R^2	NDEI	TIME(s)
Model 1	14.8168	3.8441	0.3158	0.6842	0.5607	117.3451
Model 2	14.7669	3.6616	0.3210	0.6790	0.5384	1234.3207
Model 3	9.3913	3.0474	0.1964	0.8036	0.4410	453.8153
Model 4	10.1145	3.0487	0.2090	0.7910	0.4382	53268.7555

TABLE V: Summary of Performance Metrics of all models (including execution time for 30 runs)

Overall Summary of Findings

Looking at the performance metrics above, a few observations can be made:

- **Model 1:**
  - Achieved moderate performance with $R^2 = 0.6842$, explaining about 68% of the variance.
  - Errors were relatively small and centered around zero, but overall accuracy fell short of state-of-the-art results on this dataset.
  - Training was very fast (117 seconds), making it efficient, but predictive power was limited.
  - No evidence of overfitting, but accuracy could be improved with more advanced approaches.
- **Model 2:**
  - Performance was very similar to Model 1 ($R^2 = 0.6790$), but with much larger variability across runs (high standard deviations).
  - Training time was significantly longer (1234 seconds), due to the more complex structure with additional membership functions.
  - Stability was worse compared to Model 1, with less consistent prediction errors.
  - No strong signs of overfitting, although the gap between validation error and training error was noticeably larger than in Model 1, suggesting weaker generalization.
  - Overall, increasing the number of MFs did not bring meaningful gains in accuracy.
- **Model 3:**
  - Clear performance improvement over Models 1 and 2, with $R^2 = 0.8036$, lower NMSE and NDEI values, and reduced prediction error.
  - Much more stable than Model 2 (low standard deviations), making it both accurate and reliable.
  - Training time (453.8 seconds) was higher than Model 1 but acceptable given the accuracy gains.
  - Learning curves showed no overfitting; both training and validation errors stabilized smoothly, although there still was a gap.
  - Polynomial outputs in the TSK framework proved to be the most effective modification, offering the best trade-off between accuracy, stability, and computation time.
- **Model 4:**
  - Achieved good accuracy ($R^2 = 0.7910$), but slightly worse than Model 3.
  - Exhibited instability and signs of overfitting, as validation error oscillated significantly after early epochs.
  - Residual plots showed most predictions were reasonable, but occasional large mispredictions occurred.
  - Computational cost was extremely high (over 53,000 seconds, nearly 15 hours for 30 runs).
  - Overall, Model 4 did not achieve a good balance between performance and efficiency.

- **General Conclusions:**
  - **Increasing the number of membership functions (MFs)** per fuzzy input variable (e.g., Model 1 vs. Model 2 and Model 3 vs. Model 4) does not necessarily guarantee better performance. Although a higher number of MFs increases model complexity and training time, it can also lead to instability and higher variance in the results. In fact, Model 2 and Model 4 demonstrated that simply adding more MFs does not consistently improve predictive accuracy and may even reduce reliability.
  - **Changing the form of the rule outputs from singleton (constant) to polynomial** (e.g., Model 1 vs. Model 3 and Model 2 vs. Model 4) leads to significantly better results in this dataset. The use of polynomial outputs allowed the models to capture more complex relationships in the data, improving R^2, reducing error metrics, and providing more stable training behavior. This effect was especially evident in Model 3, which achieved the best trade-off between accuracy, stability, and computational efficiency.
  - **Model complexity must be balanced with computational cost.**

Overall, the experiments show that **introducing polynomial outputs is a more effective strategy for improving TSK fuzzy regression** (in this particular dataset) than increasing the number of membership functions. Among the tested configurations, Model 3 achieved the most favorable balance of accuracy, reliability, and efficiency.

II. Application on the largest (second) dataset

Methodology Analysis

Data Preprocessing

The dataset has 81 input features and one target variable—the critical temperature, which is the last column of the dataset. First, I started by **removing duplicate rows** to make sure the data was clean. Then, both the inputs and the target were **normalized** to the range $[0, 1]$, since fuzzy systems are sensitive to input scaling and clustering works best when features are comparable. After that, the data was randomly **split into training (60%), validation (20%), and testing (20%) sets**, which allows for proper hyperparameter tuning and unbiased evaluation.

Feature Selection

To handle the large number of features and reduce computation, I used the **ReliefF algorithm** to rank features by how relevant they are to predicting the target. In ReliefF, the parameter k determines how many nearest neighbors are considered when estimating feature importance; I chose $k = 100$ because the dataset is large, and using more neighbors provides a more stable and reliable ranking. Instead of using all 81 features, I tested different subsets of the top-ranked ones. For the grid search discussed below, I specifically chose the number of features to evaluate as 4, 8, 12, 16, 20, and 30.

Model Initialization with Subtractive Clustering

Fuzzy rules were generated using **subtractive clustering**, which finds cluster centers directly from the data. The cluster influence range (r_a) controls how detailed the rules are: smaller values create more clusters and rules, while larger values give fewer, broader rules. Because this parameter affects both complexity and accuracy, I ran a **grid search** over different feature subsets and r_a values to find the best combination. Specifically, the values of r_a tested were 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, and 0.8, which allowed me to explore a range from fine-grained to more general rule bases.

Hyperparameter Tuning with Cross-Validation

For each combination of features and clustering radius, I performed **5-fold cross-validation** on the training data. This helps get a reliable estimate of the model's error and reduces the chance of overfitting. I also **recorded the mean cross-validation error** for each setup to see which configuration worked best.

Final Model Training

Once the best features and cluster radius were chosen, I **retrained the ANFIS model** using both the training and validation sets. During training, ANFIS adjusts the membership functions and the linear parts of the Takagi–Sugeno–Kang (TSK) fuzzy rules using a combination of least-squares and gradient descent. For the final model, I increased the number of epochs to 200 (before they were 100) to make sure it fully converged.

Evaluation and Visualization

The final model was tested on the independent test set, and predictions were converted back to the original scale of the critical temperature. Performance was measured using MSE, RMSE, R^2, NMSE, and NDEI metrics.

Visualization was an important part of the process. For this reason I **plotted the membership functions before and after training** to show how ANFIS adapts to the data. Learning curves show how training and validation errors change over epochs, while scatter plots, error plots, and 3D grid search plots help understand performance and the effect of hyperparameters. All figures were **saved as image files** for easy reporting, and important results like selected features, grid search outcomes, and f**inal model metrics were exported to CSV files** for further analysis and reproducibility.

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:

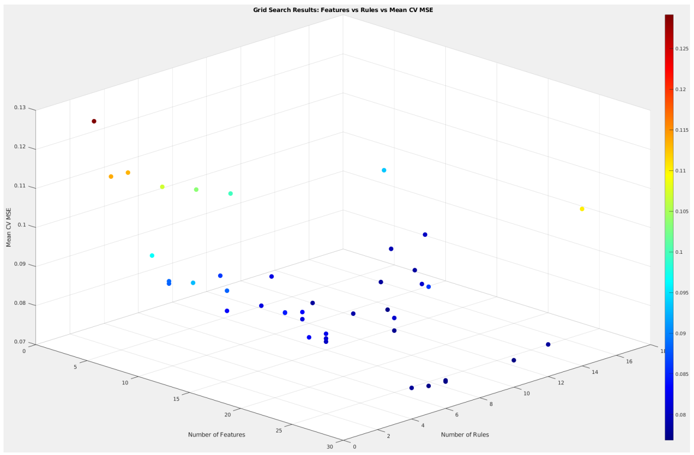


Fig. 22: Grid Search Results: Features vs Rules vs Mean CV MSE

NumFeatures	ClusterRadius	NumRules	MeanCV_MSE
4	0.2	18	0.0935
4	0.3	9	0.0997
4	0.4	7	0.1034
4	0.5	5	0.1069
4	0.6	3	0.1133
4	0.7	2	0.1136
4	0.8	1	0.1292
8	0.2	18	0.0802
8	0.3	16	0.0793
8	0.4	9	0.0818
8	0.5	6	0.0861
8	0.6	3	0.0881
8	0.7	3	0.0887
8	0.8	2	0.0966
12	0.2	15	0.0784
12	0.3	13	0.0781
12	0.4	9	0.0782
12	0.5	6	0.0816
12	0.6	4	0.0830
12	0.7	4	0.0881
12	0.8	2	0.0929
16	0.2	13	0.0809
16	0.3	11	0.0771
16	0.4	9	0.0788
16	0.5	5	0.0844
16	0.6	6	0.0814
16	0.7	6	0.0832
16	0.8	5	0.0845
20	0.2	11	0.0861
20	0.3	9	0.0809
20	0.4	9	0.0776
20	0.5	5	0.0811
20	0.6	5	0.0824
20	0.7	5	0.0803
20	0.8	4	0.0828
30	0.2	14	0.1102
30	0.3	12	0.0782
30	0.4	10	0.0769
30	0.5	6	0.0772
30	0.6	6	0.0768
30	0.7	5	0.0771
30	0.8	4	0.0780

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

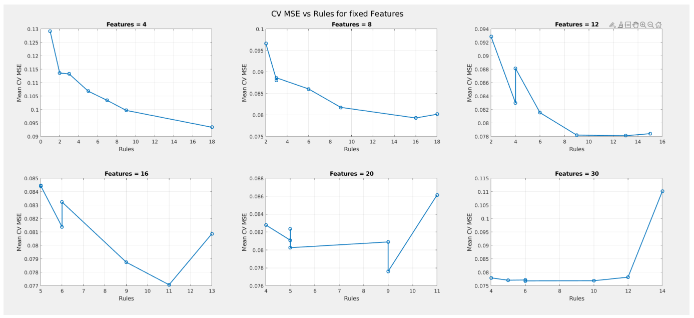


Fig. 23: Grid Search Results: Mean CV MSE vs num of Rules for fixed Features

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., $r_a = 0.2$) created many rules, which increased complexity without consistently improving performance.

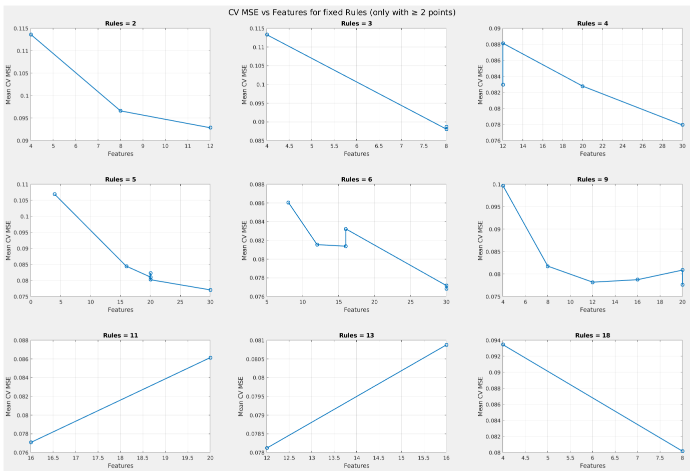


Fig. 24: Grid Search Results: Mean CV MSE vs num of Features for fixed Rules (points ≥ 2)

Larger radii ($r_a \geq 0.7$) reduced the number of rules substantially but often lead to higher errors. The best results were obtained for moderate radii between 0.3 and 0.6, where the balance between accuracy and complexity is optimal. Among all configurations, the lowest mean cross-validation MSE was achieved with 30 features and $r_a = 0.6$ (MSE ≈ 0.0768) using 6 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 errors. Larger subsets (16–30 features) typically achieve better performance, showing the benefit of including more informative features.
- **Clustering radius (r_a):** Very small radii ($r_a = 0.2$) create many rules without improving accuracy in most cases, while very large radii ($r_a \geq 0.7$) over-simplify the model and lead to worse errors. The best results are obtained at intermediate radii ($r_a = 0.3$–0.6).
- **Best configuration:** The lowest mean cross-validation MSE (≈ 0.0768) was achieved with 30 features and $r_a = 0.6$, using only 6 rules. However, other settings such as 16 features with $r_a = 0.3$ (MSE ≈ 0.0771) or 20 features with $r_a = 0.4$ (MSE ≈ 0.0776) 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 11 rules versus 30 features with 6 rules: one uses fewer features but more rules, while the other uses more features but fewer rules. Both achieve nearly identical errors,

highlighting that accuracy can often be maintained by balancing model complexity in terms of features and rules.

For this project, I selected the model with **30 features and $r_a = 0.6$ (6 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:

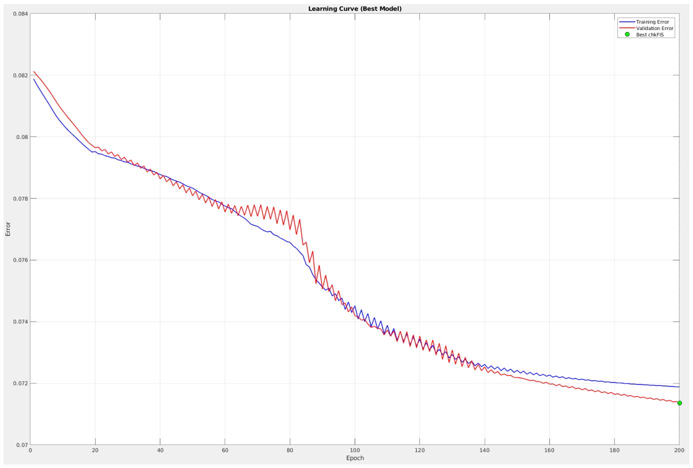


Fig. 25: Learning curves of the final TSK model

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

Metric	MSE	RMSE	R^2	NMSE	NDEI
Value	193.42	13.91	0.834	0.166	0.407

TABLE VII: Performance metrics of the final model.

Finally, regarding the performance of the final model, the plot of the model's prediction errors versus the true values on the test set as well as the true versus predicted values are shown.

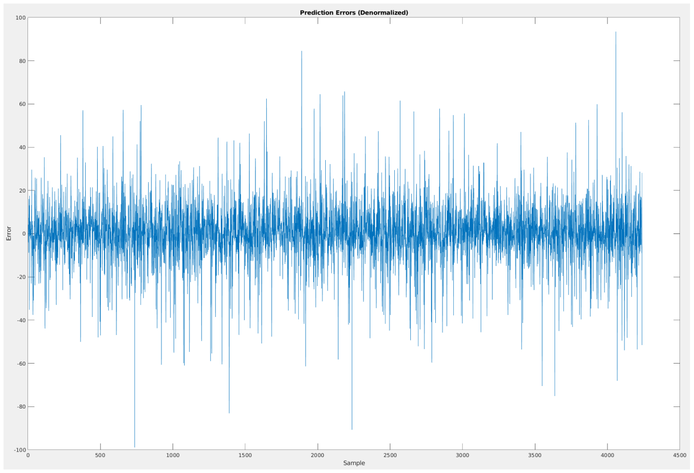


Fig. 26: Prediction errors (denormalized)

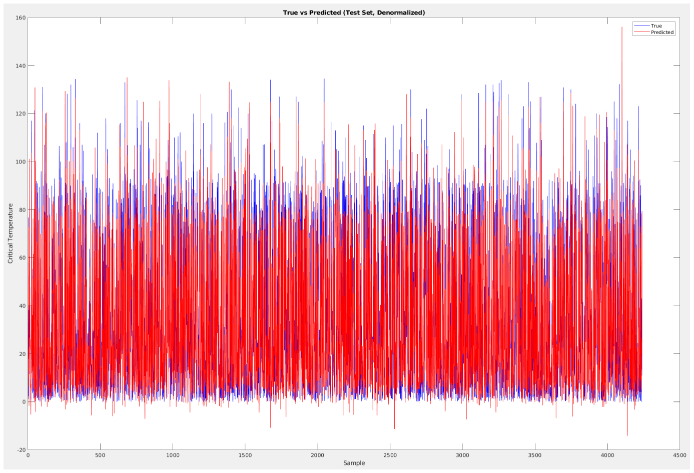


Fig. 27: True vs Predicted values (denormalized)

As shown in Figure 25, both the training and validation errors decrease steadily over the course of training, which indicates that the model is learning useful input–output relationships. The best model, marked in green, appears towards the later epochs, showing that extended training helped the model converge. One interesting point is the behavior of the validation curve relative to the training curve: at the beginning the validation error is slightly higher, then dips below the training error, rises above it again, and finally settles below towards the end. This "crossing" effect is not unusual in large datasets with complex patterns. It can sometimes be a sign of mild overfitting (when the training error keeps decreasing but the validation error does not improve at the same rate), but here the two curves remain close throughout, suggesting that any overfitting is minimal and that the model generalizes well overall. Given the high dimensionality and size of the superconductivity dataset, such fluctuations in the validation error are expected.

The prediction error plot in Figure 26 shows the residuals between the true and predicted critical temperatures on the test set (denormalized). The errors are centered around zero, with no clear systematic bias, which means the model captures the main nonlinear dependencies between input features and the target. While a few large deviations occur (with spikes above ± 80), most errors remain within a much tighter band.

Finally, the performance metrics in Table VII back up these observations. The model achieves an RMSE of 13.91 and an R^2 of 0.834, meaning it explains over 83% of the variance in the critical temperature. The normalized measures (NMSE $= 0.166$, NDEI $= 0.407$) are also well below 1.0, which means the model is making predictions that are much more accurate than just guessing the average value.

Taken together, these results show that the fuzzy regression model, combined with feature selection and subtractive clustering, is a strong approach for modeling critical temperatures in the large-scale superconductivity dataset.

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):

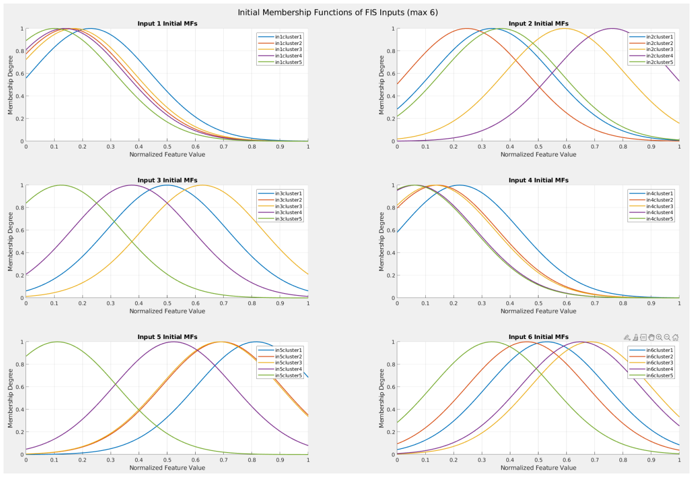


Fig. 28: Some of the Initial Membership Functions of the FIS inputs

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

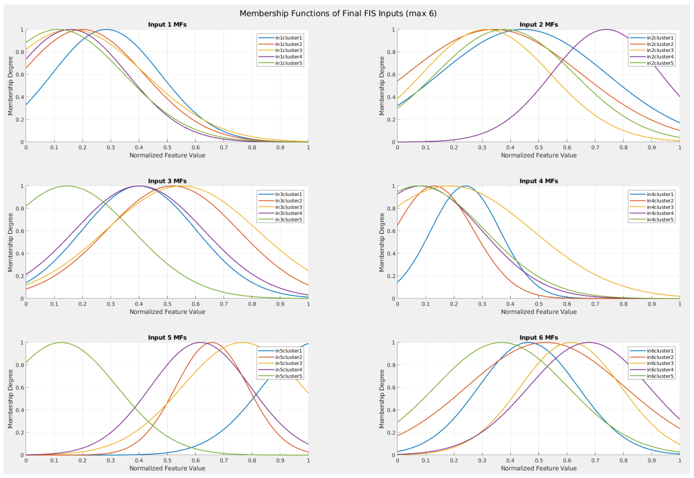


Fig. 29: Some of the Final Membership Functions of the FIS inputs

CONCLUDING OBSERVATIONS AND COMMENTS

For the `superconductivity` dataset, we observe that developing an `ANFIS` model using the 30 most significant features out of the total 81 yields satisfactory regression performance. This is evident from the metrics in Table VII, where the model achieves an RMSE of 13.91 and an R^2 value of 0.834. Interestingly, the R^2 value is even higher than the best-performing model in Part 1 (airfoil dataset: Model 3, $R^2 = 0.804$), despite the superconductivity dataset being much larger, higher-dimensional, and more complex. This highlights the effectiveness of combining feature selection and fuzzy regression for extracting predictive structure in challenging data.

The final model was constructed with only 6 fuzzy rules, which highlights the efficiency of subtractive clustering for rule generation. By comparison, a grid partitioning scheme with 2 or 3 membership functions per input would have required an infeasible number of rules (on the order of 2^{30} or 3^{30}, for 30 inputs). This shows that clustering makes fuzzy regression practical for high-dimensional datasets, even if it means possibly giving up a bit of accuracy, which did not happen in our case.

Overall, using `ReliefF` for feature selection together with subtractive clustering allowed for a compact but effective fuzzy regression model in this particularly large-scale problem. This shows how important it is to reduce dimensionality and adaptively generate rules when applying fuzzy systems to real-world datasets with many input features.

Finally, for completeness, we list the 30 most important features selected by `ReliefF` for training the final model, as shown in **Selected_Features.csv**:

63	66	1	69	31	37	58	81	12	14
27	21	67	71	7	13	28	16	78	57
15	76	26	23	60	61	22	77	25	36

TABLE VIII: Indices of the 30 most important features selected by ReliefF ($K = 100$).