Intelligent Systems (SInt) - Assignment 1

Students:

- Pedro Geitoeira, No. 87489
- Eloy Marquesan Dones, No. 112861

GitHub: SInt_G09

Hair Dryer Dataset Problem

This is a **SISO regression** problem.

In [22]:
```
from pandas import read_csv
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from pyfume.Clustering import Clusterer
from pyfume.EstimateAntecendentSet import AntecedentEstimator
from pyfume.EstimateConsequentParameters import ConsequentEstimator
from pyfume.SimpfulModelBuilder import SugenoFISBuilder
from pyfume.Tester import SugenoFISTester
from sklearn.metrics import mean_squared_error, mean_absolute_percentage_error,

# Load the hairdryer dataset.
data = read_csv('data/hairdryer.csv')

# Define the input and output columns, as well as the input variable names.
input_col = 'Voltage' # The first column as input (Voltage).
output_col = 'Temperature' # The second column as output (Temperature).
var_names = [input_col] # List of input variable names (only one input variable

# Convert the dataset to a numpy array.
data = data.to_numpy()

# Assign the X and y for the SISO system.
X = data[:, 0].reshape(-1, 1) # Single input feature.
y = data[:, 1] # Single output variable.

# Split the dataset into training and testing sets.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, rando

# Cluster the input-output space.
cl = Clusterer(x_train = X_train, y_train = y_train, nr_clus = 5) # Assuming 5 c
clust_centers, part_matrix, _ = cl.cluster(method = 'fcm') # Using the Fuzzy C-M

# Estimate the membership function parameters.
ae = AntecedentEstimator(X_train, part_matrix)
antecedent_params = ae.determineMF()

# Estimate the consequent parameters.
ce = ConsequentEstimator(X_train, y_train, part_matrix)
conseq_params = ce.suglms()
```

```
# Build the first-order Takagi-Sugeno model.
modbuilder = SugenoFISBuilder(antecedent_params, conseq_params, var_names, save_
model = modbuilder.get_model()

# Get the model predictions.
modtester = SugenoFISTester(model, X_test, var_names)
y_pred = modtester.predict()[0]


# Compute the regression metrics.

mse = mean_squared_error(y_test, y_pred)
print('Mean Squared Error: {:.3f}'.format(mse))

mape = mean_absolute_percentage_error(y_test, y_pred)
print('Mean Absolute Percentage Error: {:.1f}%'.format(mape*100))

exp_var = explained_variance_score(y_test, y_pred)
print('Explained Variance Score: {:.3f}'.format(exp_var))
```

```
 * Detected 5 rules / clusters
 * Detected Sugeno model type
Mean Squared Error: 0.642
Mean Absolute Percentage Error: 15.2%
Explained Variance Score: 0.041
```

In this problem, the best results were obtained using the **Fuzzy C-Means (FCM)** clustering algorithm with 5 clusters, resulting in an mean absolute error of 15.2%. In contrast, using the same number of clusters with the **Gustafson-Kessel (GK)** clustering algorithm yielded a higher mean absolute error of 15.4%, as detailed below.

In [20]:
```
from pandas import read_csv
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from pyfume.Clustering import Clusterer
from pyfume.EstimateAntecendentSet import AntecedentEstimator
from pyfume.EstimateConsequentParameters import ConsequentEstimator
from pyfume.SimpfulModelBuilder import SugenoFISBuilder
from pyfume.Tester import SugenoFISTester
from sklearn.metrics import mean_squared_error, mean_absolute_percentage_error,

# Load the hairdryer dataset.
data = read_csv('data/hairdryer.csv')

# Define the input and output columns, as well as the input variable names.
input_col = 'Voltage' # The first column as input (Voltage).
output_col = 'Temperature' # The second column as output (Temperature).
var_names = [input_col] # List of input variable names (only one input variable

# Convert the dataset to a numpy array.
data = data.to_numpy()

# Assign the X and y for the SISO system.
X = data[:, 0].reshape(-1, 1) # Single input feature.
y = data[:, 1] # Single output variable.

# Split the dataset into training and testing sets.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, rando
```

```
# Cluster the input-output space.
cl = Clusterer(x_train = X_train, y_train = y_train, nr_clus = 5) # Assuming 5 c
clust_centers, part_matrix, _ = cl.cluster(method = 'gk') # Using the Gustafson-

# Estimate the membership function parameters.
ae = AntecedentEstimator(X_train, part_matrix)
antecedent_params = ae.determineMF()

# Estimate the consequent parameters.
ce = ConsequentEstimator(X_train, y_train, part_matrix)
conseq_params = ce.suglms()

# Build the first-order Takagi-Sugeno model.
modbuilder = SugenoFISBuilder(antecedent_params, conseq_params, var_names, save_
model = modbuilder.get_model()

# Get the model predictions.
modtester = SugenoFISTester(model, X_test, var_names)
y_pred = modtester.predict()[0]


# Compute the regression metrics.

mse = mean_squared_error(y_test, y_pred)
print('Mean Squared Error: {:.3f}'.format(mse))

mape = mean_absolute_percentage_error(y_test, y_pred)
print('Mean Absolute Percentage Error: {:.1f}%'.format(mape*100))

exp_var = explained_variance_score(y_test, y_pred)
print('Explained Variance Score: {:.3f}'.format(exp_var))
```

```
 * Detected 5 rules / clusters
 * Detected Sugeno model type
Mean Squared Error: 0.649
Mean Absolute Percentage Error: 15.4%
Explained Variance Score: 0.031
```

Wisconsin Breast Cancer Original Dataset Problem

This is a **binary classification** problem.

In [59]:

```
import pandas as pd
from pandas import read_csv
from sklearn.model_selection import train_test_split
from pyfume.Clustering import Clusterer
from pyfume.EstimateAntecendentSet import AntecedentEstimator
from pyfume.EstimateConsequentParameters import ConsequentEstimator
from pyfume.SimpfulModelBuilder import SugenoFISBuilder
from pyfume.Tester import SugenoFISTester
from numpy import clip, column_stack, argmax
from sklearn.metrics import accuracy_score, recall_score, precision_score, f1_sc

# # Load the Wisconsin Breast Cancer Original dataset.
data = read_csv('data/wbco.csv')

# # Convert non-numeric values to NaN using a function that tries to convert val
data = data.apply(pd.to_numeric, errors = 'coerce')
```

```
# # Drop rows with any NaN values.
# data.dropna(inplace = True)

# Replace missing values with the mean of the respective columns.
data.fillna(data.mean(), inplace = True)

# Define the input and output columns, as well as the input variable names.
input_cols = ['Feature 1', 'Feature 2', 'Feature 3', 'Feature 4', 'Feature 5',
              'Feature 6', 'Feature 7', 'Feature 8', 'Feature 9'] # The first 9
output_col = 'Cancer Presence' # The last column as output (Cancer Presence).
var_names = input_cols # List of input variable names.
var_names = [var_names[i].title().replace(' ', '') for i in range(0, len(var_nam

# Convert the dataset to a numpy array.
data = data.to_numpy()

# Assign the X and y for the classification problem.
X = data[:, :-1] # All columns except the last one as input features.
y = data[:, -1] # The last column as the output variable (Cancer Presence).

# Splitting into training and test sets (ensuring class distribution is preserve
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, rando


# Cluster the input-output space.
cl = Clusterer(x_train = X_train, y_train = y_train, nr_clus = 7) # Assuming 7 c
clust_centers, part_matrix, _ = cl.cluster(method = 'fcm') # Using the Fuzzy C-M

# Estimate the membership functions parameters.
ae = AntecedentEstimator(X_train, part_matrix)
antecedent_params = ae.determineMF()

# Estimate the consequent parameters.
ce = ConsequentEstimator(X_train, y_train, part_matrix)
conseq_params = ce.suglms()

# Build the first-order Takagi-Sugeno model.
modbuilder = SugenoFISBuilder(antecedent_params, conseq_params, var_names, save_
model = modbuilder.get_model()

# Get the model predictions.
modtester = SugenoFISTester(model, X_test, var_names)
y_pred_probs = clip(modtester.predict()[0], 0, 1)
y_pred_probs = column_stack((1 - y_pred_probs, y_pred_probs))
y_pred = argmax(y_pred_probs, axis = 1)


# Compute the classification metrics.

acc_score = accuracy_score(y_test, y_pred)
print('Accuracy: {:.3f}'.format(acc_score))

rec_score = recall_score(y_test, y_pred)
print('Recall: {:.3f}'.format(rec_score))

prec_score = precision_score(y_test, y_pred)
print('Precision Score: {:.3f}'.format(prec_score))

F1_score = f1_score(y_test, y_pred)
print('F1-Score: {:.3f}'.format(F1_score))
```

```
kappa = cohen_kappa_score(y_test, y_pred)
print('Kappa Score: {:.3f}'.format(kappa))
```

```
 * Detected 7 rules / clusters
 * Detected Sugeno model type
Accuracy: 0.979
Recall: 0.938
Precision Score: 1.000
F1-Score: 0.968
Kappa Score: 0.952
```

In this problem, the best results were obtained using the **Fuzzy C-Means (FCM)** clustering algorithm with 7 clusters, resulting in an accuracy of 97.9%. In contrast, using the same number of clusters with the **Gustafson-Kessel (GK)** clustering algorithm yielded a lower accuracy of 95%, as detailed below.

In the `train_test_split` function, the parameter `stratify = y` was used to ensure that both the training and test datasets maintain the same class distribution as the original dataset.

In [56]:
```
import pandas as pd
from pandas import read_csv
from sklearn.model_selection import train_test_split
from pyfume.Clustering import Clusterer
from pyfume.EstimateAntecendentSet import AntecedentEstimator
from pyfume.EstimateConsequentParameters import ConsequentEstimator
from pyfume.SimpfulModelBuilder import SugenoFISBuilder
from pyfume.Tester import SugenoFISTester
from numpy import clip, column_stack, argmax
from sklearn.metrics import accuracy_score, recall_score, precision_score, f1_sc

# # Load the Wisconsin Breast Cancer Original dataset.
data = read_csv('data/wbco.csv')

# # Convert non-numeric values to NaN using a function that tries to convert val
data = data.apply(pd.to_numeric, errors = 'coerce')

# # Drop rows with any NaN values.
# data.dropna(inplace = True)

# Replace missing values with the mean of the respective columns.
data.fillna(data.mean(), inplace = True)

# Define the input and output columns, as well as the input variable names.
input_cols = ['Feature 1', 'Feature 2', 'Feature 3', 'Feature 4', 'Feature 5',
              'Feature 6', 'Feature 7', 'Feature 8', 'Feature 9'] # The first 9
output_col = 'Cancer Presence' # The last column as output (Cancer Presence).
var_names = input_cols # List of input variable names.
var_names = [var_names[i].title().replace(' ', '') for i in range(0, len(var_nam

# Convert the dataset to a numpy array.
data = data.to_numpy()

# Assign the X and y for the classification problem.
X = data[:, :-1] # All columns except the last one as input features.
y = data[:, -1] # The last column as the output variable (Cancer Presence).
```

```
# Splitting into training and test sets (ensuring class distribution is preserve
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, rando


# Cluster the input-output space.
cl = Clusterer(x_train = X_train, y_train = y_train, nr_clus = 7) # Assuming 7 c
clust_centers, part_matrix, _ = cl.cluster(method = 'gk') # Using the Gustafson-

# Estimate the membership functions parameters.
ae = AntecedentEstimator(X_train, part_matrix)
antecedent_params = ae.determineMF()

# Estimate the consequent parameters.
ce = ConsequentEstimator(X_train, y_train, part_matrix)
conseq_params = ce.suglms()

# Build the first-order Takagi-Sugeno model.
modbuilder = SugenoFISBuilder(antecedent_params, conseq_params, var_names, save_
model = modbuilder.get_model()

# Get the model predictions.
modtester = SugenoFISTester(model, X_test, var_names)
y_pred_probs = clip(modtester.predict()[0], 0, 1)
y_pred_probs = column_stack((1 - y_pred_probs, y_pred_probs))
y_pred = argmax(y_pred_probs, axis = 1)


# Compute the classification metrics.

acc_score = accuracy_score(y_test, y_pred)
print('Accuracy: {:.3f}'.format(acc_score))

rec_score = recall_score(y_test, y_pred)
print('Recall: {:.3f}'.format(rec_score))

prec_score = precision_score(y_test, y_pred)
print('Precision Score: {:.3f}'.format(prec_score))

F1_score = f1_score(y_test, y_pred)
print('F1-Score: {:.3f}'.format(F1_score))

kappa = cohen_kappa_score(y_test, y_pred)
print('Kappa Score: {:.3f}'.format(kappa))
```

```
 * Detected 7 rules / clusters
 * Detected Sugeno model type
Accuracy: 0.950
Recall: 0.854
Precision Score: 1.000
F1-Score: 0.921
Kappa Score: 0.885
```

In both problems, it was also observed that the FCM algorithm is relatively faster than the GK algorithm.