Πανεπιστήμιο Δυτικής Αττικής

Σχολή Μηχανικών

Τμήμα Μηχανικών Πληροφορικής και Υπολογιστών

Διπλωματική εργασία

Ανάπτυξη πακέτου Python για την υλοποίηση ασαφών συστημάτων

Ιωάννης Γιαννόπουλος
711161025

Επιβλέπων:

Πάρις Μαστοροκώστας
Καθηγητής

Αιγάλεω - Αθήνα, Δεκέμβριος, 2023

Ημερομηνία εξέτασης: 13/3/2024

Μέλη της εξεταστικής επιτροπής:

Paris Mastorokostas Digitally signed by Paris Mastorokostas Date: 2024.03.13 13:05:46 +02'00'	Christos Troussas Digitally signed by Christos Troussas Date: 2024.03.13 16:23:06 +02'00'	Panagiota Tselenti Digitally signed by Panagiota Tselenti Date: 2024.03.13 14:11:55 +02'00'
Πάρις Μαστορокώστας	Χρήστος Τρούσσας	Παναγιώτα Τσελέντη
Καθηγητής	Επίκουρος Καθηγητής	ΕΔΙΠ

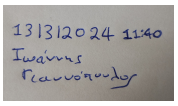
ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος Ιωάννης Γιαννόπουλος του Γεωργίου, με αριθμό μητρώου 711161025 φοιτητής του Πανεπιστημίου Δυτικής Αττικής της Σχολής Μηχανικών του Τμήματος Μηχανικών Πληροφορικής και Υπολογιστών, δηλώνω υπεύθυνα ότι:

«Είμαι συγγραφέας αυτής της διπλωματικής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.

Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου».

Ο Δηλών



Περίληψη

Σκοπός της εργασίας είναι αρχικά η θεωρητική μελέτη των ασαφών συστημάτων τύπου Mamdani και Takagi-Sugeno και η υλοποίηση τους σε Python ώστε να μελετηθούν οι δυνατότητές τους σε προβλήματα αναγνώρισης συστημάτων και υποστήριξης αποφάσεων. Αρχικά μελετήθηκαν οι υπάρχουσες υλοποιήσεις Συστημάτων Ασαφούς Συμπερασμού σε Python. Αναπτύχθηκε ο αλγόριθμος Subtractive Clustering και το Νευροασαφές Σύστημα Ασαφούς Συμπερασμού ANFIS που συνδυάστηκαν με το πακέτο Simpful για τη δημιουργία ενός ολοκληρωμένου πακέτου για την ανάπτυξη Συστημάτων Ασαφούς Συμπερασμού. Τέλος, παρουσιάζετε η αποτελεσματικότητα του πακέτου σε πραγματικά σύνολα δεδομένων.

Abstract

The aim of this paper is firstly the theoretical study of fuzzy systems of the type Mamdani and Takagi-Sugeno and their implementation in Python in order to study their potential in system identification problems and decision support. Initially, existing implementations of Systems Fuzzy Inference Systems were studied in Python. The Subtractive Clustering algorithm and the ANFIS Fuzzy Inference System were developed and combined with the Simpful package to create a complete package for developing Fuzzy Inference Systems. Finally, the effectiveness of the package is shown on real datasets.

Πίνακας περιεχομένων

Πίνακας σχημάτων

Πίνακας πινάκων

1

Εισαγωγή

Η επιδίωξη της προσομοίωσης της ανθρώπινης λήψης αποφάσεων σε υπολογιστικά μοντέλα αποτελεί εδώ και πολύ καιρό έναν δελεαστικό στόχο για τους ερευνητές σε διάφορους επιστημονικούς τομείς. Η παραδοσιακή δυαδική λογική, αν και αποτελεί τον ακρογωνιαίο λίθο της κλασικής πληροφορικής, συχνά υπολείπεται όταν έρχεται αντιμέτωπη με τις περίπλοκες λεπτές αποχρώσεις των σεναρίων του πραγματικού κόσμου, όπου οι δυαδικές αποφάσεις είναι ανεπαρκείς. Εδώ έγκειται η γοητεία της ασαφούς λογικής, ενός παραδείγματος που εισήγαγε ο Lotfi Zadeh τη δεκαετία του 1960 και σχεδιάστηκε για να μιμηθεί την πολύπλοκη διαδικασία της ανθρώπινης λογικής καθώς παλεύει με το ανακριβές και το αόριστο.

Στον πυρήνα της, η ασαφής λογική επεκτείνει τη συμβατική δυαδική αξιολόγηση ”αληθές ή ψευδές” ώστε να περιλαμβάνει ένα φάσμα τιμών αλήθειας μεταξύ 0 και 1, προσφέροντας έτσι μια πιο ρεαλιστική αναπαράσταση της εγγενούς ασάφειας του κόσμου. Πρόκειται για μια μορφή λογικής πολλαπλών τιμών που προέρχεται από τη θεωρία ασαφών συνόλων, όπου η αλήθεια κάθε δήλωσης γίνεται θέμα βαθμού. Αυτή η διαφοροποιημένη προσέγγιση της λογικής έχει βαθιές επιπτώσεις σε τομείς που μαστίζονται από την αβεβαιότητα και την ασάφεια, από τα συστήματα ελέγχου και την αναγνώριση προτύπων μέχρι την τεχνητή νοημοσύνη και όχι μόνο.

Η Python, μια γλώσσα προγραμματισμού που φημίζεται για την απλότητα και την αναγνωσιμότητά της, αποτελεί ιδανική πλατφόρμα για την εξερεύνηση και την υλοποίηση συστημάτων ασαφούς συμπερασμού (ΣΑΣ). Η ευρεία γκάμα βιβλιοθηκών και πλαισίων που διαθέτει, σε συνδυασμό με μια εκτεταμένη κοινότητα προγραμματιστών και ερευνητών, παρέχει πρόσφορο έδαφος για την εφαρμογή της ασαφούς λογικής στην επιστημονική έρευνα. Επιπλέον, η εγγενής ευελιξία και η διαλειτουργικότητα της Python με άλλες γλώσσες προγραμματισμού και εργαλεία επιτρέπουν την απρόσκοπτη ενσωμάτωση της ασαφούς λογικής σε ένα πλήθος υπολογιστικών συστημάτων.

Η παρούσα εργασία έχει ως στόχο να γεφυρώσει το χάσμα μεταξύ των θεωρητικών βά-

σεων της ασαφούς λογικής και των πρακτικών εφαρμογών της, αξιοποιώντας το οικοσύστημα της Python. Ξεκινάμε περιγράφοντας τις έννοιες της Ασαφής Λογικής και των Συστημάτων Ασαφούς Συμπερασμού. Στη συνέχεια, αναφερόμαστε στα προγραμματιστικά εργαλεία που χρησιμοποιήθηκαν και δείχνουμε τα αποτελέσματα από πραγματικά δεδομένα. Τέλος, παρουσιάζεται ο τρόπος με τον οποίο έγινε η παραγωγή των αποτελεσμάτων με τη χρήση της Python και των βιβλιοθηκών της.

2

Θεωρητικό μέρος

Σε αυτό το κεφάλαιο παρουσιάζεται το θεωρητικό υπόβαθρο της διπλωματικής εργασίας.

2.1 Ασαφής Λογική

Η ασαφής λογική είναι μια μορφή λογικής πολλών τιμών ή πιθανολογικής λογικής: ασχολείται με συλλογισμούς που είναι προσεγγιστικοί και όχι σταθεροί και ακριβείς. Σε αντίθεση με τα παραδοσιακά δυαδικά σύνολα (όπου οι μεταβλητές μπορούν να λάβουν αληθείς ή ψευδείς τιμές), οι μεταβλητές της ασαφούς λογικής μπορούν να έχουν μια τιμή αλήθειας που κυμαίνεται σε βαθμό μεταξύ 0 και 1. Η ασαφής λογική έχει επεκταθεί για να χειριστεί την έννοια της μερικής αλήθειας, όπου η τιμή αλήθειας μπορεί να κυμαίνεται μεταξύ εντελώς αληθούς και εντελώς ψευδούς.

2.1.1 Ασαφές Σύνολο

Το ασαφές σύνολο είναι μια μαθηματική έννοια που αντιπροσωπεύει μια κλάση αντικειμένων με ένα συνεχές φάσμα βαθμών συμμετοχής, το οποίο έρχεται σε αντίθεση με τα κλασικά σύνολα όπου ένα αντικείμενο είτε ανήκει είτε δεν ανήκει στο σύνολο. Η έννοια αυτή είναι κεντρική στον τομέα της ασαφούς λογικής και εισήχθη από τον Lotfi Zadeh το 1965[21] ως επέκταση της κλασικής έννοιας του συνόλου.

Μαθηματικός τύπος σε διακριτό χώρο:

Έστω το διακριτό πεδίο $U = \{x_1, x_2 \dots x_n\}$. Ένα ασαφές σύνολο A περιγράφεται από τα ζεύγη ασαφών singletons $(x_i, \mu_A(x_i)),\ i \in \mathbb{N}$:

$$A = \{(x_1, \mu_A(x_1)), (x_2, \mu_A(x_2)), \dots, (x_n, \mu_A(x_n))\}$$

Εναλλακτική περιγραφή με τη σημειογραφία άθροισης (summation notation):

$$A = \frac{\mu_A(x_1)}{x_1} + \frac{\mu_A(x_2)}{x_2} + \dots + \frac{\mu_A(x_n)}{x_n} = \sum_{i=1}^{n} \frac{\mu_A(x_i)}{x_i}$$

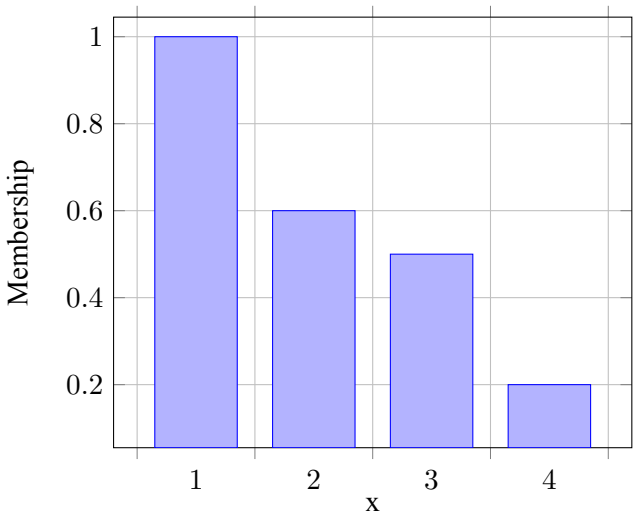
Τα σύμβολα «Σ»,«+» δηλώνουν την ένωση των ασαφών singletons και το «/» δηλώνει το ασαφές singleton $(x_i, \mu_A(x_i))$.

Παράδειγμα:

$$U = \{1, 2, 3, 4\}$$

$$A = \{(1, 1), (2, 0.6), (3, 0.5), (4, 0.2))\}$$

$$A = \{\frac{1}{1} + \frac{0.6}{2} + \frac{0.5}{3} + \frac{0.2}{4}\}$$



Εικόνα 2.1. Απεικόνιση ασαφούς συνόλου

Μαθηματικός τύπος σε συνεχή χώρο:

$$A = \int_U \frac{\mu_A(x)}{x}$$

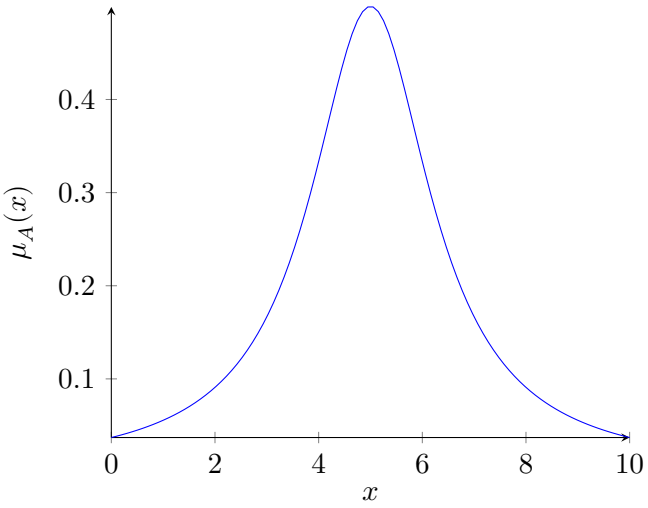
Το σύμβολο «$\int$» δηλώνει την ένωση των ασαφών singleton.

Παράδειγμα:

$$U = [0, 10]$$

$$A = \{x, \mu_A(x) | x \in U\}$$

$$\mu_A(x) = \frac{1}{2 + (x + 5)^2}$$



Εικόνα 2.2. Απεικόνιση ασαφούς συνόλου

2.1.2 Συναρτήσεις συμμετοχής

Μια συνάρτηση συμμετοχής στο πλαίσιο της ασαφούς λογικής είναι μια καμπύλη που ορίζει πώς κάθε σημείο του χώρου εισόδου αντιστοιχίζεται σε μια τιμή συμμετοχής ή βαθμό συμμετοχής μεταξύ 0 και 1. Η συνάρτηση αυτή χρησιμοποιείται για να ποσοτικοποιήσει την ασάφεια και την ανακρίβεια ενός ασαφούς συνόλου και αποτελεί βασικό συστατικό του συστήματος ασαφούς λογικής.

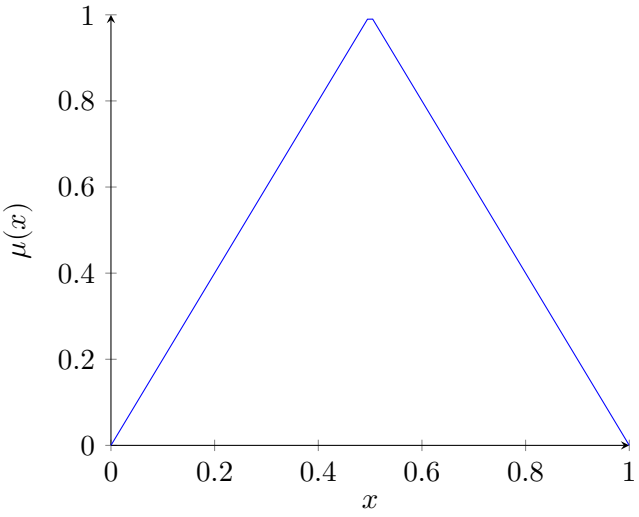
Οι πιο διαδεδομένες συναρτήσεις συμμετοχής είναι: η τριγωνική, η τραπεζοειδής, η γκαουσιανή και η γενικευμένη καμπανοειδής.

Τριγωνική συνάρτηση συμμετοχής

$$\text{tri\_MF}(x; a, b, c) = \begin{cases} 0 & x < a \\ \frac{x-a}{b-a} & a \geq x < b \\ \frac{c-x}{c-b} & b \geq x < c \\ 0 & c \geq x \end{cases}$$

Συμπαγής μορφή σε ψευδοκώδικα:

$$\text{tri\_MF}(x; a, b, c) = \max(\min(\frac{x-a}{b-a}, \frac{c-x}{c-b}), 0)$$



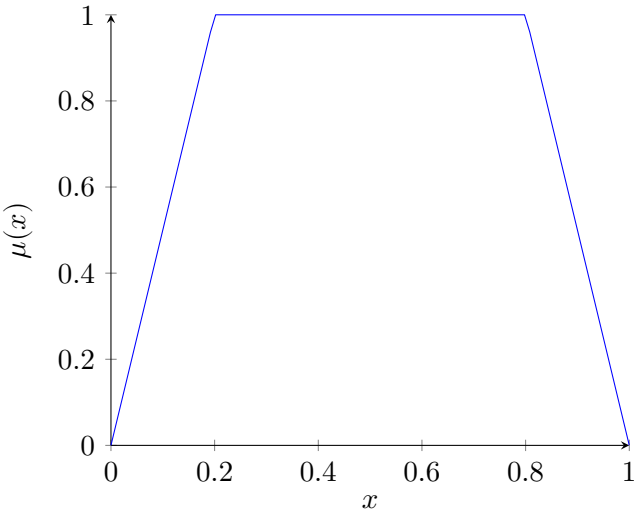
Εικόνα 2.3. Παράδειγμα tri_MF(x;0,0.5,1)

Τραπεζοειδής συνάρτηση συμμετοχής

$$\text{trap\_MF}(x;a,b,c,d) = \begin{cases} 0 & x < a \\ \frac{x-a}{b-a} & a \geq x < b \\ 1 & b \geq x < c \\ \frac{d-x}{d-c} & c \geq x < d \\ 0 & d \geq x \end{cases}$$

Συμπαγής μορφή σε ψευδοκώδικα:

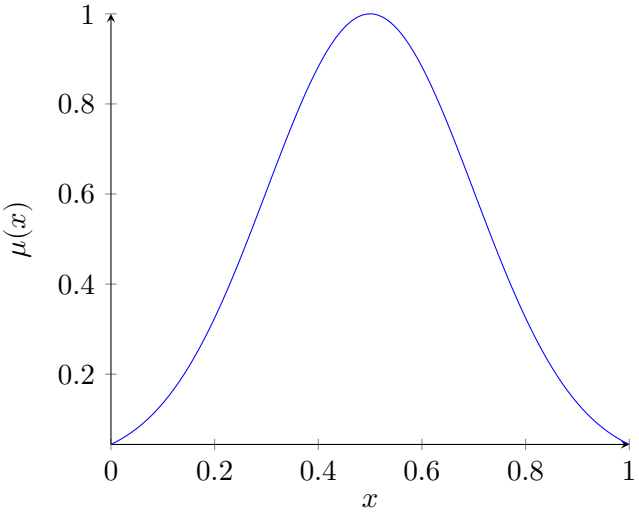
$$\text{trap\_MF}(x;a,b,c) = \max(\min(\frac{x-a}{b-a}, 1, \frac{d-x}{d-c}), 0)$$



Εικόνα 2.4. Παράδειγμα trap_MF(x;0,0.2,0.8,1)

Γκαουσιανή συνάρτηση συμμετοχής

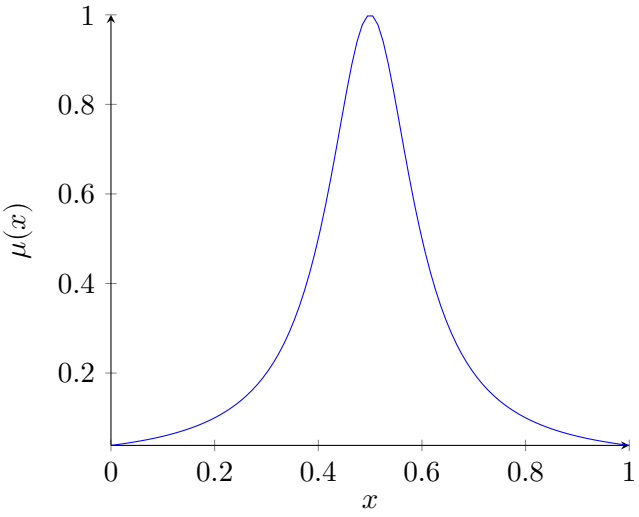
$$\text{gauss\_MF}(x;\mu,\sigma) = e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$$



Εικόνα 2.5. Παράδειγμα gauss_MF(x;0.5,0.2)

Γενικευμένη καμπανοειδής συνάρτηση συμμετοχής

$$\text{bell\_MF}(x;a,b,c) = \frac{1}{1+(\frac{x-c}{a})^{2b}}$$



Εικόνα 2.6. Παράδειγμα gauss_MF(x;0.1,1,0,5)

2.1.3 Λεκτικές Μεταβλητές

Στην ασαφή λογική, μια λεκτική μεταβλητή είναι μια μεταβλητή της οποίας οι τιμές είναι λέξεις ή προτάσεις μιας φυσικής ή τεχνητής γλώσσας και όχι αριθμητικές τιμές. Αυτές οι λέξεις ή προτάσεις περιγράφουν μια συγκεκριμένη ιδιότητα του συστήματος που μοντελοποιείται με ποιοτικούς όρους και είναι εγγενώς ανακριβείς ή ασαφείς. Η χρήση λεκτικών μεταβλητών αποτελεί ακρογωνιαίο λίθο της ασαφούς λογικής, καθώς επιτρέπει την αναπαράσταση υποκειμενικών ή ανακριβών εννοιών όπως «ζεστό», «κρύο», «νέος» ή «γέρος».

Κάθε λεκτική μεταβλητή συνδέεται με ένα σύνολο όρων, το οποίο είναι το σύνολο των ονομάτων των λεκτικών τιμών που μπορεί να λάβει η μεταβλητή. Για παράδειγμα, αν έχουμε μια λεκτικών μεταβλητή με το όνομα «Θερμοκρασία», το σύνολο όρων της μπορεί να περιλαμβάνει τις τιμές «Χαμηλή», «Μέτρια» και «Υψηλή».

Επιπλέον, κάθε λεκτική τιμή χαρακτηρίζεται από ένα ασαφές σύνολο, το οποίο ορίζεται από μια συνάρτηση συμμετοχής που αποδίδει σε κάθε πιθανή αριθμητική τιμή της μεταβλητής έναν βαθμό συμμετοχής που κυμαίνεται από 0 έως 1. Αυτή η συνάρτηση συμμετοχής ορίζει τον βαθμό στον οποίο μια αριθμητική τιμή ανήκει σε ένα ασαφές σύνολο. Για παράδειγμα, η λεκτική τιμή «Υψηλή» στο σύνολο όρων «Θερμοκρασία» μπορεί να αναπαρασταθεί από ένα ασαφές σύνολο που αποδίδει υψηλότερες τιμές συμμετοχής σε υψηλότερες θερμοκρασίες.

Η χρήση λεκτικών μεταβλητών επιτρέπει στα συστήματα ασαφούς λογικής να μιμούνται τις ανθρώπινες διαδικασίες λήψης αποφάσεων και συλλογισμού, χειριζόμενα ασαφείς έννοιες με τρόπο που είναι πιο διαισθητικός και φυσικός για τους ανθρώπους. Είναι ένας τρόπος να γεφυρωθεί το χάσμα μεταξύ της δυαδικής ακρίβειας των παραδοσιακών υπολογιστών

2.1.4 Ασαφείς Κανόνες

Ένας ασαφής κανόνας στο πλαίσιο των συστημάτων ασαφούς λογικής είναι μια υπό συνθήκη δήλωση της μορφής «IF-THEN» που ορίζει πώς να συμπεράνουμε μια έξοδο από δεδομένες εισόδους. Σε αντίθεση με τους δυαδικούς κανόνες στα συστήματα κλασικής λογικής, όπου τα κατηγορήματα πρέπει να είναι εξ ολοκλήρου αληθή ή ψευδή, οι ασαφείς κανόνες ασχολούνται με βαθμούς αλήθειας. Αυτοί οι κανόνες έχουν σχεδιαστεί για να αποτυπώνουν την ανακρίβεια της ανθρώπινης γλώσσας και των διαδικασιών λήψης αποφάσεων στον πραγματικό κόσμο.

Ακολουθεί η τυπική δομή ενός ασαφούς κανόνα:

IF [σύνολο προϋποθέσεων που ικανοποιούνται] THEN [σύνολο συνεπειών].

Για παράδειγμα, ένας απλός ασαφής κανόνας μπορεί να είναι:

IF the temperature is high THEN the cooling system speed is high.

Εδώ, «high temperature» και «high cooling system speed» είναι ασαφείς μεταβλητές με βαθμούς συμμετοχής που κυμαίνονται μεταξύ 0 και 1. Αυτός ο κανόνας αντιστοιχίζει το ασαφές σύνολο εισόδου «temperature» στο ασαφές σύνολο εξόδου «cooling system speed».

Σε αυτόν τον κανόνα:

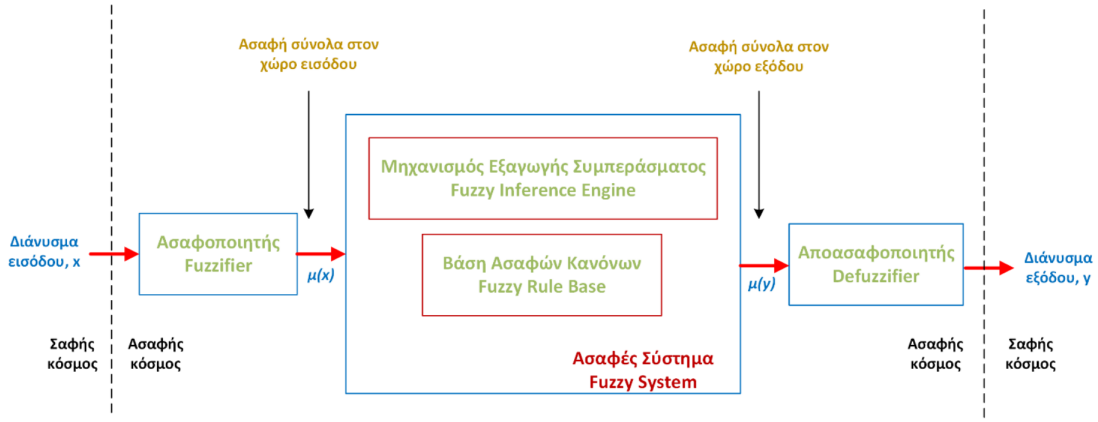
- «temperature» είναι μια λεκτική μεταβλητή εισόδου που έχει προηγουμένως οριστεί με ένα αντίστοιχο ασαφές σύνολο.
- «high» είναι μια λεκτική τιμή που έχει ένα ασαφές σύνολο που ορίζει ποιο εύρος τιμών θερμοκρασίας θεωρείται «high» (συνήθως με μια συνάρτηση συμμετοχής).
- «cooling system speed» είναι η λεκτική μεταβλητή εξόδου.
- «high», όπως σχετίζεται με τη μεταβλητή «cooling system speed», θα έχει επίσης ένα καθορισμένο ασαφές σύνολο με μια συνάρτηση συμμετοχής που απεικονίζει την έξοδο σε ένα εύρος τιμών ταχύτητας.

Πολλαπλοί ασαφείς κανόνες μπορούν να συνδυαστούν για να σχηματίσουν μια βάση κανόνων, η οποία αποτελεί βασικό στοιχείο ενός ασαφούς συστήματος συμπερασμού. Όταν παρέχονται δεδομένα εισόδου στο σύστημα, όλοι οι κανόνες αξιολογούνται παράλληλα και τα αποτελέσματα αθροίζονται για να παραχθεί μια ασαφής έξοδος. Αυτή η έξοδος συνήθως αποασαφοποιείται για να προκύψει ένα σαφές, εφαρμόσιμο αποτέλεσμα.

Οι ασαφείς κανόνες κατασκευάζονται με βάση τη γνώση των εμπειρογνωμόνων ή τα εμπειρικά δεδομένα και είναι θεμελιώδους σημασίας για τη λειτουργία των ελεγκτών ασαφούς λογικής και των συστημάτων λήψης αποφάσεων. Επιτρέπουν τον χειρισμό προβλημάτων που είναι πολύ πολύπλοκα για τις παραδοσιακές ποσοτικές τεχνικές, αλλά εξακολουθούν να είναι διαχειρίσιμα με ποιοτικές περιγραφές.

2.2 Συστήματα Ασαφούς Συμπερασμού - ΣΑΣ

Ένα Σύστημα Ασαφούς Συμπερασμού (Fuzzy Inference System, FIS) είναι ένα πλαίσιο συλλογισμού και υπολογισμού που βασίζεται στην ασαφή θεωρία συνόλων, στους ασαφείς κανόνες IF-THEN και στην ασαφή συλλογιστική. Είναι μια μορφή τεχνητής νοημοσύνης που επιτρέπει τη μοντελοποίηση πολύπλοκων συστημάτων που είναι δύσκολο να οριστούν με ακριβείς εξισώσεις και την αντιμετώπιση της αβεβαιότητας με δομημένο τρόπο. Ο στόχος ενός ΣΑΣ είναι η χρήση λεκτικών μεταβλητών και ενός συνόλου κανόνων για τη λήψη αποφάσεων ή αξιολογήσεων που μιμούνται την ανθρώπινη συλλογιστική.



Εικόνα 2.7. Δομή συστήματος ασαφούς συμπερασμού

Υπάρχουν δύο κύριοι τύποι ΣΑΣ:

- Σύστημα Ασαφούς Συμπερασμού Mamdani: Προτάθηκε από τον Ebrahim Mamdani το 1975 [11]. Χρησιμοποιείται ευρέως για συστήματα ελέγχου. Αξιοποιείται η ασαφή λογική για να αντιστοιχισθεί ένας χώρος εισόδου σε έναν χώρο εξόδου χρησιμοποιώντας μια σειρά από ασαφείς κανόνες IF-THEN.

- Σύστημα Ασαφούς Συμπερασμού Takagi-Sugeno-Kang (TSK): Αναπτύχθηκε από τους Takagi, Sugeno [18] και βελτιώθηκε αργότερα από τον Kang. Το TSK ΣΑΣ χρησιμοποιεί μια διαφορετική προσέγγιση για τις συνέπειες των κανόνων, οι οποίες είναι συναρτήσεις των μεταβλητών εισόδου, συνήθως γραμμικές εξισώσεις, αντί για ασαφή σύνολα. Αυτός ο τύπος συστήματος χρησιμοποιείται συχνά για την προσέγγιση συναρτήσεων και τη μοντε-

λοποίηση.

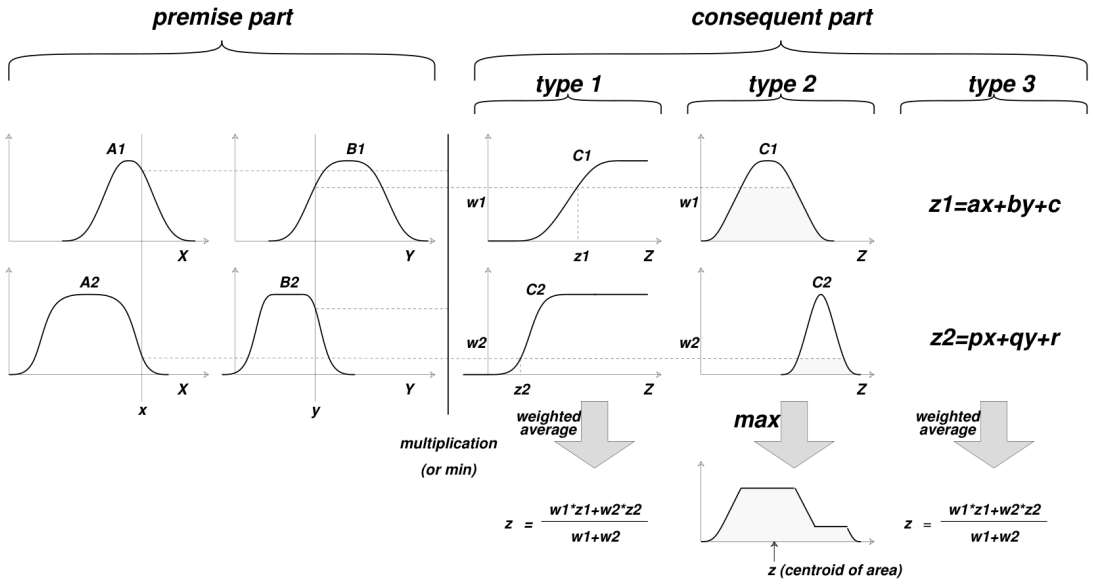
Η τυπική διαδικασία ασαφούς συμπερασμού περιλαμβάνει τα ακόλουθα βήματα:

1. Ασαφοποίηση: Η διαδικασία μετατροπής «σαφών» εισόδων (ακριβείς αριθμητικές τιμές) σε ασαφείς τιμές (βαθμοί συμμετοχής) σύμφωνα με προκαθορισμένα ασαφή σύνολα. Αυτό είναι απαραίτητο επειδή το σύστημα συμπερασμού λειτουργεί με ασαφείς έννοιες και όχι με ακριβείς αριθμούς.

2. Αξιολόγηση κανόνων: Εφαρμογή τελεστών ασαφούς λογικής (AND, OR, NOT) για την αξιολόγηση της τιμής αλήθειας της προϋπόθεσης κάθε κανόνα. Στην περίπτωση ενός συστήματος Mamdani, αυτό περιλαμβάνει την τομή των ασαφών συνόλων που σχετίζονται με κάθε μέρος της προϋπόθεσης για τον προσδιορισμό του βαθμού αλήθειας του κανόνα.

3. Συγκέντρωση: Συνδυασμός των ασαφών συνόλων που έχουν κοπεί στο ύψος που καθορίζεται από τη διαδικασία αξιολόγησης του κανόνα για να σχηματιστεί ένα ενιαίο ασαφές σύνολο για κάθε μεταβλητή εξόδου.

4. Αποασαφοποίηση: Η διαδικασία μετατροπής του ασαφούς συνόλου εξόδου σε μια ευκρινή τιμή. Υπάρχουν διάφορες μέθοδοι για την αποασαφοποίηση, αλλά η πιο συνηθισμένη είναι η μέθοδος του κεντροειδούς (centroid), η οποία υπολογίζει το κέντρο βάρους του ασαφούς συνόλου για να βρει μια ασαφή τιμή που αντιπροσωπεύει καλύτερα το συναγόμενο ασαφές συμπέρασμα.



Εικόνα 2.8. Απεικόνιση ασαφούς συμπερασμού, όπου type-2 ανιστοιχεί σε Mamdani και type-3 σε TSK

Τα Συστήματα Ασαφούς Συμπερασμού χρησιμοποιούνται εκτενώς σε διάφορες εφαρμογές, όπως συστήματα ελέγχου, αναγνώριση προτύπων, συστήματα υποστήριξης αποφάσεων και συστήματα εμπειρογνωμόνων, επειδή μπορούν να χειριστούν την ασάφεια και την αβεβαιότητα που ενυπάρχουν σε πολύπλοκα και ασαφώς καθορισμένα συστήματα.

2.3 Συσταδοποίηση

Όταν υπάρχουν πολλά δεδομένα για το μοντέλο που θέλουμε να φτιάξουμε ή/και δεν υπάρχει εμπειρική γνώση, τότε μπορούν να χρησιμοποιηθούν τεχνικές συσταδοποίησης για την εύρεση των κανόνων του Συστήματος Ασαφούς Συμπερασμού. Οι πιο διαδεδομένες τεχνικές συσταδοποίησης για τα Συστήματα Ασαφούς Συμπερασμού είναι η **Fuzzy C-Means**[1] και η **Subtractive Clustering**[3]. Σε αυτή την εργασία χρησιμοποιήθηκε η τεχνική Subtractive Clustering.

2.3.1 Subtractive Clustering

Η τεχνική αυτή λειτουργεί ως εξής: έστω ένα σύνολο σημείων $x_1, x_2, \dots, x_n$ σε ένα χώρο M διαστάσεων.

1. Κανονικοποιούμε όλα τα δεδομένα στον ίδιο χώρο διαστάσεων (υπερκύβο).

2. Υπολογίζουμε τη δυναμική του κάθε σημείου, $i \in N$, ως κέντρο συστάδας:

$$P_i = \sum_{j=1}^{n} e^{-a||x_i - x_j||^2} \text{ όπου } a = \frac{4}{r_a^2}$$

$$P_i = \sum_{j=1}^{n} e^{-4\frac{||x_i - x_j||^2}{r_a^2}} = \sum_{j=1}^{n} e^{-\frac{1}{2}\left(\frac{||x_i - x_j||}{\frac{r_a}{2\sqrt{2}}}\right)^2}$$

3. Επιλέγουμε το σημείο με τη μεγαλύτερη δυναμική.

4. Ξανά υπολογίζουμε τη δυναμική του κάθε σημείου ως:

$$P_i = P_i - P_1^* e^{-\beta||x_i - x_1^*||^2} \text{ όπου } \beta = \frac{4}{r_b^2}$$

$$P_i = P_i - P_1^* e^{-4\frac{||x_i - x_1^*||^2}{r_b^2}} = P_i - P_1^* e^{-\frac{1}{2}\left(\frac{||x_i - x_1^*||}{\frac{r_b}{2\sqrt{2}}}\right)^2}$$

5. Επιλέγουμε το σημείο x_i με τη μεγαλύτερη δυναμική P_i ως υποψήφιο κέντρο συστάδας x_k^* με P_k^*.

6. Αν το $P_k^* > \bar{\epsilon} P_1^*$ τότε αποδεχόμαστε το x_k^* ως κέντρο συστάδας

7. Αν το $P_k^* < \underline{\epsilon} P_1^*$ τότε τερματίζουμε τον αλγόριθμο

8. Διαφορετικά, διαλέγουμε τη μικρότερη απόσταση, έστω d_{min}, του x_k^* προς όλα τα κέντρα που έχουνε βρεθεί.

   8.1. Αν ισχύει $\frac{d_{min}}{r_a} + \frac{P_k^*}{P_1^*} \geq 1$ τότε επιλέγουμε το x_k^* ως κέντρο συστάδας

   8.2. Διαφορετικά απορρίπτουμε το x_k^* και θέτουμε $P_k^* = 0$. Επιλέγουμε το επόμενο στοιχείο x_i με τη μεγαλύτερη δυναμική ως το νέο υποψήφια κέντρο x_k^* και ξανά δοκιμάζουμε τις συνθήκες (πηγαίνουμε στο βήμα 6)

9. Επαναλαμβάνουμε το βήμα 4.

Τα κέντρα των συστάδων που προκύπτουν από τον αλγόριθμο, έστω z_i^* με M στοιχεία, χωρίζονται τα στοιχεία τους σε αυτά των μεταβλητών εισόδου και αυτά των μεταβλητών εξόδου.

Αν οι μεταβλητές εξόδου είναι J, τότε στοιχεία των μεταβλητών εισόδου είναι:

$$x_i^* = [z_{i,1}^*, z_{i,2}^*, \dots, z_{i,M-J}^*]^T$$

και τα στοιχεία των μεταβλητών εξόδου είναι:

$$y_i^* = [z_{i,M-J+1}^*, z_{i,M-J+2}^*, \dots, z_{i,M}^*]^T$$

Με τα παραπάνω προκύπτει το τμήμα υπόθεσης:

$$\mu_{A^i} = e^{-\frac{1}{2}\left(\frac{\|x-x_i^*\|}{\frac{r_a}{2\sqrt{2}}}\right)^2} = e^{-\frac{1}{2}\left(\frac{x-x_{i,1}^*}{\frac{r_a}{2\sqrt{2}}}\right)^2} \cdot e^{-\frac{1}{2}\left(\frac{x-x_{i,2}^*}{\frac{r_a}{2\sqrt{2}}}\right)^2} \dots e^{-\frac{1}{2}\left(\frac{x-x_{i,M-J}^*}{\frac{r_a}{2\sqrt{2}}}\right)^2}$$

Στο τμήμα απόδοσης, για μοντέλα Tagaki-Sugeno-Kang είναι:

$$R^{(i)} = \textit{IF } x_1 \textit{ is } A_1^i \textit{ AND ...AND } x_M \textit{ is } A_m^i \textit{ THEN } y \textit{ is } w_1$$

Στο τμήμα απόδοσης, για μοντέλα Tagaki-Sugeno-Kang με πολυωνυμικές συναρτήσεις:

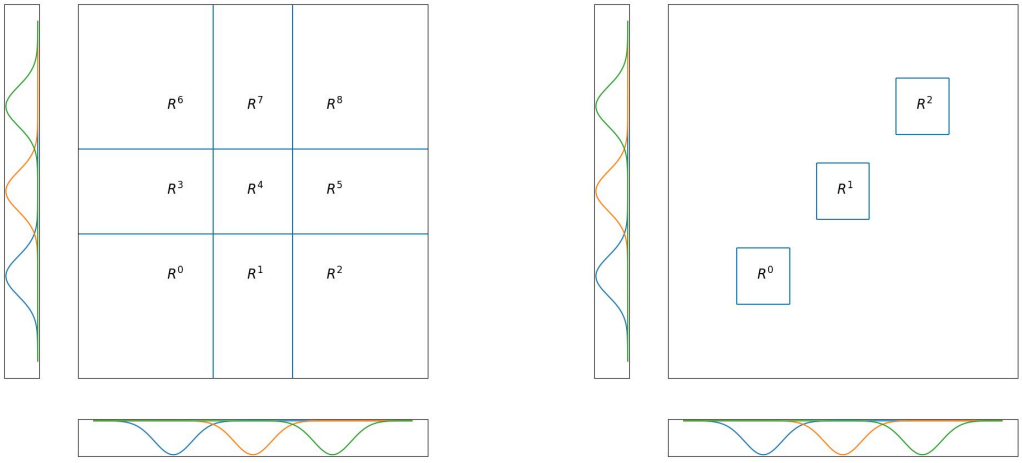
$$R^{(i)} = \textit{IF } x_1 \textit{ is } A_1^i \textit{ AND ...AND } x_m \textit{ is } A_m^i \textit{ THEN}$$
$$y = f_1(x) = a_{i,0} + a_{i,1}x_1 + \dots + a_{i,n}x_n$$

Σημείωση: Παρατηρούμε ότι στις παραπάνω εξισώσεις χρησιμοποιείται η Γκαουσιανή συνάρτηση συμμετοχής, με $\sigma = \frac{r_a}{2\sqrt{2}}$. Ίσως, ο αλγόριθμος να μπορούσε να αναπροσαρμοστεί για να υποστηρίξει άλλες συναρτήσεις συμμετοχής.

2.4 Grid Partitioning

Η τεχνική «Grid Partitioning» (διαχωρισμός πλέγματος) χρησιμοποιείται για τη διακριτοποίηση του χώρου εισόδου ενός ασαφούς συστήματος συμπερασμού. Περιλαμβάνει τη διαίρεση του πεδίου κάθε μεταβλητής εισόδου σε έναν αριθμό ασαφών συνόλων με τις σχετικές συναρτήσεις συμμετοχής, τα οποία μαζί σχηματίζουν ένα πλέγμα πάνω στο χώρο εισόδου. Ο κάθε πιθανός συνδυασμός μεταβλητών εισόδου, αποτελεί το μέρος υπόθεσης ενός κανόνα, στον οποίο θα πρέπει να επιλεχθεί το μέρος απόδοσης.

Η τεχνική «Grid Partitioning» δεν ορίζει ποιο θα είναι το μέρος απόδοσης κάθε κανόνα. Οπότε χρειάζεται να χρησιμοποιηθεί κάποιος αλγόριθμος για την επιλογή του μέρους απόδοσης σε κάθε κανόνα. Στην περίπτωση των συστημάτων TSK μπορεί να αξιοποιηθεί το νευροασαφές σύστημα ANFIS για την εύρεση των κατάλληλων τιμών στο μέρος της απόδοσης.



(α) Grid Partition (β) Subtractive Clustering

Εικόνα 2.9. Σύγκριση χώρου εισόδων μεταξύ Grid Partition και Subtractive Clustering για δύο μεταβλητές εισόδου.

2.5 Νευροασαφές Δίκτυο ANFIS

Το ANFIS, «Adaptive Neuro-Fuzzy Inference System» (Προσαρμοστικό Νευρο-Ασαφές Σύστημα Συμπερασμού), είναι ένα είδος νευρωνικού δικτύου που βασίζεται στο ασαφές σύστημα συ-

μπερασμού Takagi-Sugeno. Η αρχιτεκτονική ANFIS ενσωματώνει τα καλύτερα χαρακτηριστικά των ασαφών συστημάτων και των νευρωνικών δικτύων, γεγονός που της επιτρέπει να αξιοποιεί τα πλεονεκτήματα και των δύο σε ένα ενιαίο πλαίσιο. [8]

Ο συνδυασμός των Ασαφών Συστημάτων Συμπερασμού με τα Νευρωνικά Δίκτυα, στο ANFIS, παρέχει έναν μηχανισμό με τον οποίο τα ασαφή συστήματα μπορούν να μαθαίνουν από τα δεδομένα, καθιστώντας τους ασαφείς κανόνες προσαρμοστικούς. Ο μηχανισμός εκπαίδευσης συνήθως περιλαμβάνει την οπισθοδιάδοση (backpropagation), παρόμοια με τον τρόπο εκπαίδευσης των παραδοσιακών νευρωνικών δικτύων.

Ένα δίκτυο ANFIS αποτελείται συνήθως από πέντε επίπεδα:

1. Ασαφοποίηση: Οι κόμβοι σε αυτό το επίπεδο μετατρέπουν τα δεδομένα εισόδου σε κατάλληλες λεκτικές τιμές, οι οποίες είναι ασαφείς τιμές με βαθμό συμμετοχής σε διάφορα ασαφή σύνολα.

2. Κανόνες: Κάθε κόμβος σε αυτό το στρώμα αντιπροσωπεύει έναν ασαφή κανόνα. Εφαρμόζει έναν ασαφή τελεστή στις εισόδους που προέρχονται από το πρώτο στρώμα για να υπολογίσει την ισχύ πυροδότησης του κανόνα.

3. Κανονικοποίηση: Σε αυτό το στρώμα, η ισχύς πυροδότησης κάθε κανόνα κανονικοποιείται ως προς το άθροισμα της ισχύος πυροδότησης όλων των κανόνων.

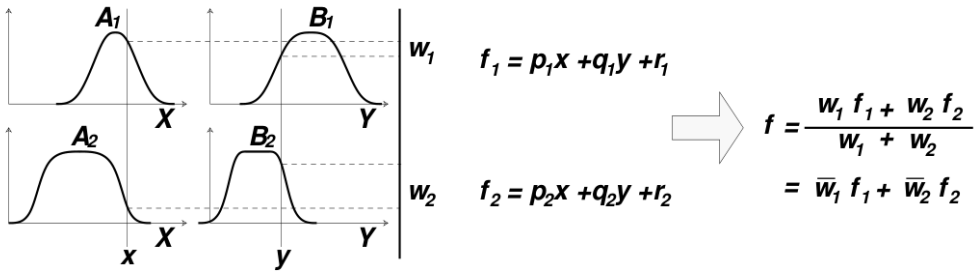
4. Αποασαφοποίηση: Αυτό το στρώμα υπολογίζει τη συμμετοχή κάθε κανόνα στην τελική έξοδο με βάση τις κανονικοποιημένες δυνάμεις πυροδότησης και τη συνάρτηση εξόδου του κανόνα.

5. Συνολική έξοδος: Το τελευταίο στρώμα υπολογίζει τη συνολική έξοδο ως άθροισμα όλων των εισερχόμενων σημάτων από το προηγούμενο στρώμα.

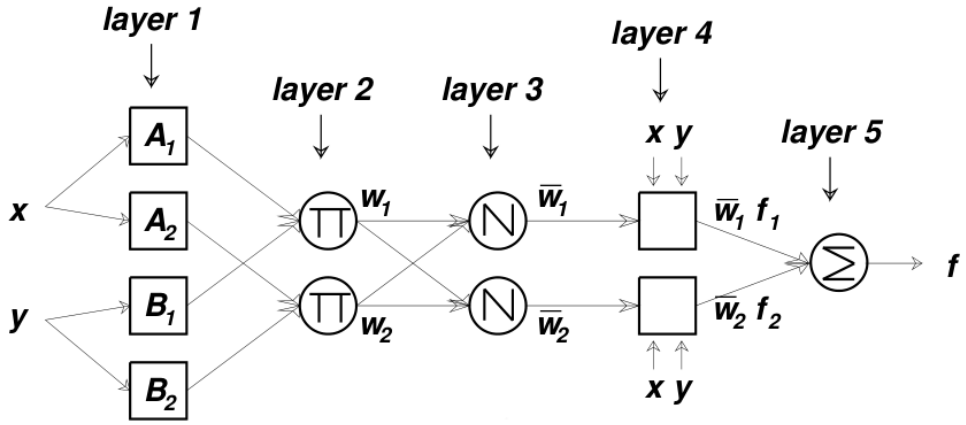
Για παράδειγμα, έστω ένα σύστημα ασαφούς συμπερασμού δύο εισόδων x, y και μιας εξόδου με τους παρακάτω ασαφείς κανόνες. Απεικονίζεται στη συνέχεια το αντίστοιχο σύστημα TSK.

$$R^{\{1\}}\text{: } \textit{If } x \textit{ IS } A_1 \textit{ AND } y \textit{ IS } B_1 \textit{ THEN } f_1 = p_1 x + q_1 y + r_1$$

$$R^{\{2\}}\text{: } \textit{If } x \textit{ IS } A_2 \textit{ AND } y \textit{ IS } B_2 \textit{ THEN } f_2 = p_2 x + q_2 y + r_2$$



Εικόνα 2.10. Ασαφής Συμπερασμός «type-3»



Εικόνα 2.11. ANFIS type-3

3

Τεχνικό Μέρος

Σε αυτό το κεφάλαιο θα παρουσιαστούν τα εργαλεία που χρησιμοποιήθηκαν με μια σύντομη περιγραφή.

3.1 Python

Δημιουργήθηκε από τον Ολλανδό Γκίντο βαν Ρόσσουμ (Guido van Rossum) στο ερευνητικό κέντρο Centrum Wiskunde & Informatica (CWI) το 1989 και κυκλοφόρησε για πρώτη φορά το 1991, ενώ αυτή τη στιγμή βρίσκεται στην έκδοση 3.12. Είναι διερμηνευόμενη γλώσσα, υψηλού επιπέδου και γενικού σκοπού και υποστηρίζει πολλά προγραμματιστικά μοτίβα όπως διαδικαστικός προγραμματισμός, αντικειμενοστραφής προγραμματισμός και συναρτησιακός προγραμματισμός.

3.2 Pytorch

Είναι ένα framework μηχανικής μάθησης βασισμένο στη βιβλιοθήκη Torch, που να αναπτύχθηκε αρχικά από τη Meta AI και τώρα αποτελεί μέρος της "ομπρέλας" του Linux Foundation. Είναι ευρέως διαδεμένη στην επιστημονική κοινότητα της python και καθιστά εύκολη την ανάπτυξη μοντέλων μηχανικής μάθησης.

3.3 Simpful

Το Simpful είναι ένα πακέτο Python που έχει σχεδιαστεί για να διευκολύνει τη χρήση της ασαφούς λογικής. Το πακέτο παρέχει μια απλή διεπαφή για τη δημιουργία και εκτέλεση ασαφών συστημάτων συμπερασμού με έμφαση στην αναγνωσιμότητα και την ευκολία χρήσης. Σκοπός του είναι να είναι φιλικό προς τον χρήστη, ειδικά για όσους μπορεί να μην είναι ειδικοί στην ασαφή λογική.

3.4 Numpy

Το NumPy, που σημαίνει Numerical Python, είναι ένα θεμελιώδες πακέτο για αριθμητικούς υπολογισμούς στην Python. Παρέχει υποστήριξη για πίνακες (συμπεριλαμβανομένων των πολυδιάστατων πινάκων), καθώς και μια σειρά μαθηματικών συναρτήσεων για να λειτουργήσει σε αυτούς τους πίνακες.

3.5 Matplotlib

Η Matplotlib είναι μια ολοκληρωμένη βιβλιοθήκη για τη δημιουργία στατικών, κινούμενων και διαδραστικών απεικονίσεων στην Python. Είναι μία από τις πιο δημοφιλείς και ευρέως χρησιμοποιούμενες βιβλιοθήκες οπτικοποίησης δεδομένων στην κοινότητα προγραμματισμού της Python. Η Matplotlib διευκολύνει τη δημιουργία γραφικών παραστάσεων, ιστογραμμάτων, φασμάτων ισχύος, ραβδογραμμάτων, διαγραμμάτων σφάλματος, διαγραμμάτων διασποράς κ.λπ. με λίγες μόνο γραμμές κώδικα.

3.6 Pandas

Το Pandas είναι μια δημοφιλής βιβλιοθήκη ανάλυσης και χειρισμού δεδομένων ανοικτού κώδικα για τη γλώσσα προγραμματισμού Python. Παρέχει ευέλικτες και ισχυρές δομές δεδομένων για τον αποτελεσματικό χειρισμό δομημένων (ταμπλό, πολυδιάστατων, δυνητικά ετερογενών) και χρονοσειρών δεδομένων. Το όνομα «pandas» προέρχεται από το «panel data», έναν όρο της οικονομετρίας για δομημένα σύνολα δεδομένων.

3.7 Seaborn

Το Seaborn είναι μια βιβλιοθήκη οπτικοποίησης δεδομένων Python που βασίζεται στο matplotlib. Παρέχει μια διεπαφή υψηλού επιπέδου για τη σχεδίαση ελκυστικών και κατατοπιστικών στατιστικών γραφικών.

3.8 Λοιπά εργαλεία που εξετάσθηκαν

Τα παρακάτω εργαλεία εξετάσθηκαν να συμπεριληφθούν στο τελικό πακέτο, αλλά τελικά απορρίφθηκαν. Όμως, η γνώση που αποκτήθηκε κατά τη δοκιμή τους ήταν χρήσιμη για την ανάπτυξη του τελικού πακέτου της Python.

3.8.1 Fuzzy-Expert

«Το Fuzzy-Expert είναι ένα πακέτο Python για τη δημιουργία ασαφών συστημάτων συμπερασμού Mamdani. Το Fuzzy-Expert δέχεται ένα μείγμα ασαφών και τραγανών τιμών και συναφών

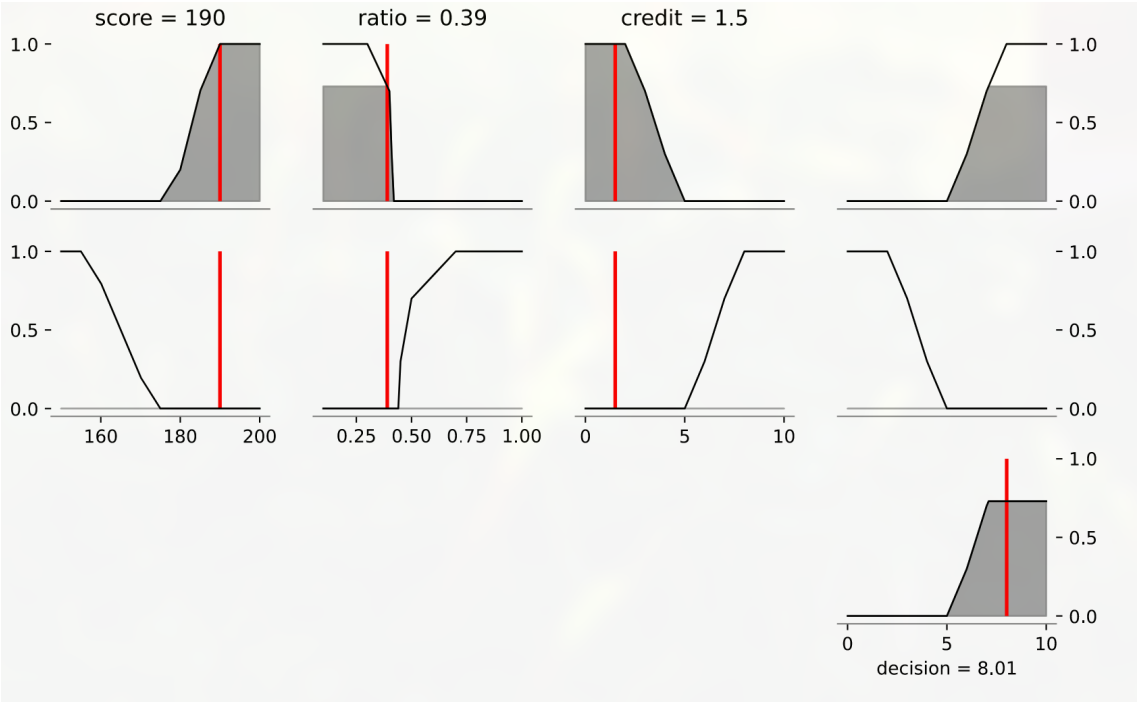
τιμών βεβαιότητας. Το σύστημα συμπερασμού επιστρέφει ασαφείς τιμές και συντελεστές βεβαιότητας για τα συμπεράσματα των κανόνων.»

Στη συνέχεια ακολουθεί παράδειγμα χρήσης του πακέτου Fuzzy Expert.

```
import matplotlib.pyplot as plt
import numpy as np

from fuzzy_expert.variable import FuzzyVariable
from fuzzy_expert.rule import FuzzyRule

variables = {
    ”score”: FuzzyVariable(
        universe_range=(150, 200),
        terms={
            ”High”: [(175, 0), (180, 0.2), (185, 0.7), (190, 1)],
            ”Low”: [(155, 1), (160, 0.8), (165, 0.5), (170, 0.2), (175, 0)],
        },
    ),
    ”ratio”: FuzzyVariable(
        universe_range=(0.1, 1),
        terms={
            ”Goodr”: [(0.3, 1), (0.4, 0.7), (0.41, 0.3), (0.42, 0)],
            ”Badr”: [(0.44, 0), (0.45, 0.3), (0.5, 0.7), (0.7, 1)],
        },
    ),
    ”credit”: FuzzyVariable(
        universe_range=(0, 10),
        terms={
            ”Goodc”: [(2, 1), (3, 0.7), (4, 0.3), (5, 0)],
            ”Badc”: [(5, 0), (6, 0.3), (7, 0.7), (8, 1)],
        },
    ),
    ”decision”: FuzzyVariable(
        universe_range=(0, 10),
        terms={
            ”Approve”: [(5, 0), (6, 0.3), (7, 0.7), (8, 1)],
            ”Reject”: [(2, 1), (3, 0.7), (4, 0.3), (5, 0)],
        },
```

```
    ),
}

rules = [
    FuzzyRule(
        premise=[
            (”score”, ”High”),
            (”AND”, ”ratio”, ”Goodr”),
            (”AND”, ”credit”, ”Goodc”),
        ],
        consequence=[(”decision”, ”Approve”)],
    ),
    FuzzyRule(
        premise=[
            (”score”, ”Low”),
            (”AND”, ”ratio”, ”Badr”),
            (”OR”, ”credit”, ”Badc”),
        ],
        consequence=[(”decision”, ”Reject”)],
    )
]

model = DecompositionalInference(
    and_operator=”min”,
    or_operator=”max”,
    implication_operator=”Rc”,
    composition_operator=”max-min”,
    production_link=”max”,
    defuzzification_operator=”cog”,
)

model(
    variables=variables,
    rules=rules,
    score=190,
    ratio=0.39,
    credit=1.5,
)
```

```
73
74 plt.figure(figsize=(10, 6))
75 model.plot(
76     variables=variables,
77     rules=rules,
78     score=190,
79     ratio=0.39,
80     credit=1.5,
81 )
```



Εικόνα 3.1. Απεικόνιση Συστήματος Ασαφούς Συμπερασμού

Συνοπτικά:

- Στις γραμμές 7-36 γίνεται η δήλωση των μεταβλητών.
- Στις γραμμές 38-35 γίνεται η δήλωση των κανόνων.
- Στις γραμμές 57-72 γίνεται η δήλωση του Συστήματος Ασαφούς Συμπερασμού.
- Στις γραμμές 74-81 δημιουργείται η απεικόνιση του συστήματος.

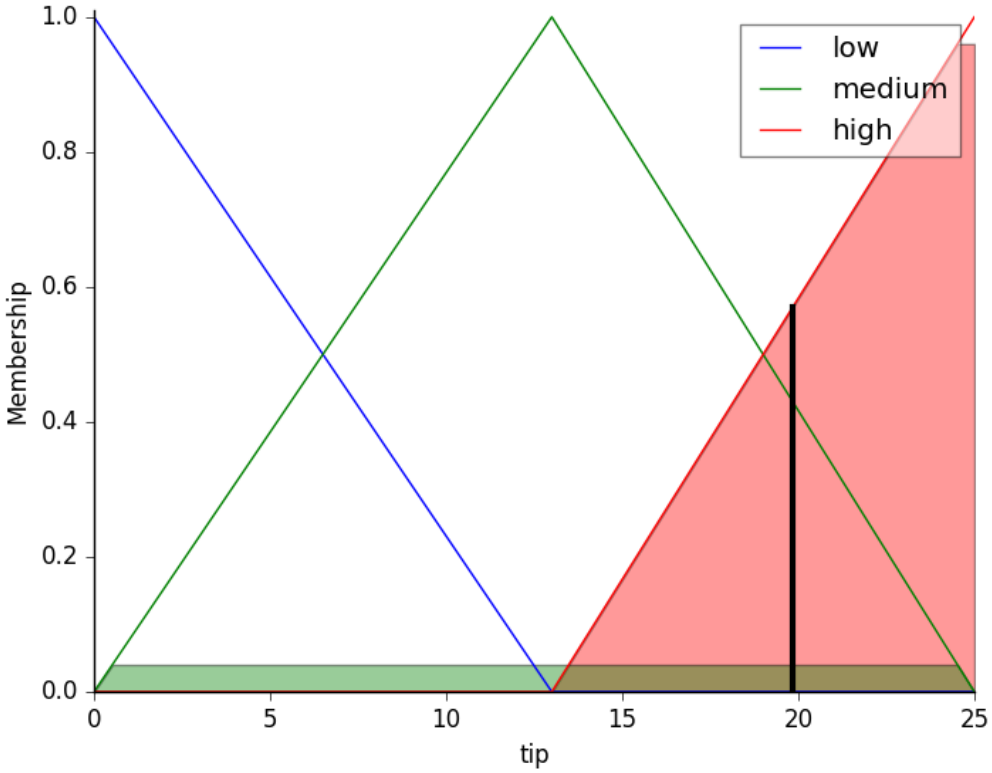
Το πακέτο Fuzzy Expert δε χρησιμοποιήθηκε γιατί έχει να ενημερωθεί από το 2021 και θα δημιουργούσε πρόβλημα στις σύγχρονες εκδόσεις της Python και των απαιτούμενων πακέτων.

3.8.2 Scikit-Fuzzy

Το πακέτο Scikit-Fuzzy στην Python, είναι μια συλλογή αλγορίθμων για την ασαφή λογική και τα ασαφή συστήματα στο πλαίσιο της στοίβας SciPy. Συμπληρώνει της γνωστές βιβλιοθήκες SciPy, NumPy και Matplotlib με λειτουργικότητα ασαφούς λογικής, προσφέροντας μια σειρά εργαλείων για τον χειρισμό ασαφών συνόλων και την προσομοίωση ασαφών συστημάτων.

Στη συνέχεια ακολουθεί παράδειγμα χρήσης του πακέτου Fuzzy Expert.

```
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl

quality = ctrl.Antecedent(np.arange(0, 11, 1), 'quality')
service = ctrl.Antecedent(np.arange(0, 11, 1), 'service')
tip = ctrl.Consequent(np.arange(0, 26, 1), 'tip')

quality.automf(3)
service.automf(3)

tip['low'] = fuzz.trimf(tip.universe, [0, 0, 13])
tip['medium'] = fuzz.trimf(tip.universe, [0, 13, 25])
tip['high'] = fuzz.trimf(tip.universe, [13, 25, 25])

rules = [
  ctrl.Rule(quality['poor'] | service['poor'], tip['low']),
  ctrl.Rule(service['average'], tip['medium']),
  ctrl.Rule(service['good'] | quality['good'], tip['high']),
]

tipping_ctrl = ctrl.ControlSystem([rule1, rule2, rule3])
tipping = ctrl.ControlSystemSimulation(tipping_ctrl)

tipping.input['quality'] = 6.5
tipping.input['service'] = 9.8
tipping.compute()
print(tipping.output['tip'])
tip.view(sim=tipping)
```



Εικόνα 3.2. Απεικόνιση Συστήματος Ασαφούς Συμπερασμού

Συνοπτικά:

- Στις γραμμές 5-7 γίνεται η δήλωση των μεταβλητών.
- Στις γραμμές 9-14 αποδίδονται σε κάθε μεταβλητή οι λεκτικές τιμές της.
- Στις γραμμές 16-20 δημιουργούνται οι κανόνες του συστήματος.
- Στις γραμμές 22-23 δημιουργείται το Σύστημα Ασαφούς Συμπερασμού.
- Στις γραμμές 38-35 γίνεται η δήλωση των κανόνων.
- Στις γραμμές 25-29 υπολογίζεται η έξοδος του Συστήματος Ασαφούς Συμπερασμού για δεδομένες τιμές εισόδου.

Το πακέτο scikit-fuzzy δε χρησιμοποιήθηκε γιατί εμφάνισε ανεπίλυτο πρόβλημα κατά τη διάρκεια της εκπόνησης της εργασίας.

3.8.3 anfis-pytorch και sanfis

Τα πακέτα anfis-pytorch και sanfis, είναι διαφορετικές υλοποιήσεις του ANFIS σε PyTorch. Δε χρησιμοποιήθηκαν γιατί εμφάνισαν πρόβλημα με την τελευταία έκδοση της Python και των υπόλοιπων πακέτων, όμως αποτέλεσαν έμπνευση για την υλοποίηση του ANFIS με το πακέτο Pytorch.

4

Αποτελέσματα

Αναπτύχθηκαν οι αλγόριθμοι Subtractive Clustering και ANFIS, με έμφαση στην ταχύτητα και την απλότητα-συντηρισιμότητα του κώδικα, ώστε να είναι όσο το δυνατόν περισσότερο συμβατές, ή αναβαθμίσιμες, με τις νεότερες εκδόσεις της python.

Στη συνέχεια ακολουθούν παραδείγματα χρήσης των παραπάνω υλοποιήσεων σε συνδυασμό με τη βιβλιοθήκη simpful. Συγκεκριμένα, χρησιμοποιήθηκαν 2 σετ δεδομένων, το **Tetuan Power Consumption**[16] και το **Office Occupancy**[2]. Για το πρώτο έγινε προσπάθεια πρόβλεψης της κατανάλωσης ενέργειας, ενώ στο δεύτερο έγινε προσπάθεια πρόβλεψης της «διαθεσιμότητας» ενός γραφείου. Ακολουθεί αναλυτικός πίνακας.

Πίνακας 4.1. Μέθοδοι που δοκιμάστηκαν ανά σύνολο δεδομένων

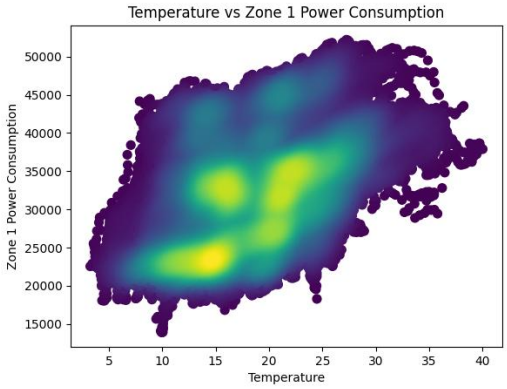
Dataset	Modeling	Type	Input Variables	Output Variables
Tetuan Power Cons.	Grid	Mamdani	Temperature,Humidity	Zone 1 Power Cons.
Tetuan Power Cons.	Grid	ANFIS	Temperature,Humidity	Zone 1 Power Cons.
Tetuan Power Cons.	SubClust	TSK	Temperature,Humidity	Zone 1 Power Cons.
Tetuan Power Cons.	SubClust	ANFIS	Temperature,Humidity	Zone 1 Power Cons.
Office Occupancy	Grid	Mamdani	Temperature,Light	Occupancy
Office Occupancy	Grid	ANFIS	Temperature,Light	Occupancy
Office Occupancy	SubClust	TSK	Temperature,Light	Occupancy
Office Occupancy	SubClust	ANFIS	Temperature,Light	Occupancy

4.1 Tetuan Power Consumption

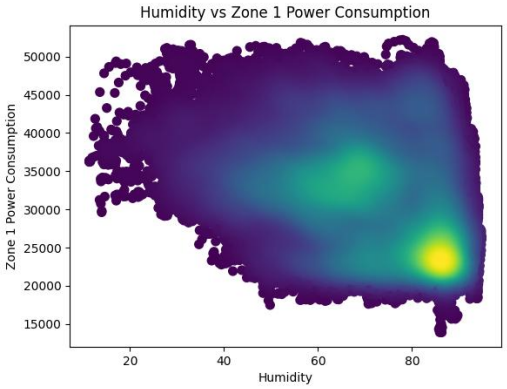
Πίνακας 4.2. Χαρακτηριστικά συνόλου δεδομένων

Name	Role	Type	Description
DateTime	Feature	Date	Each ten minutes
Temperature	Feature	Continuous	Weather Temperature of Tetouan city
Humidity	Feature	Continuous	Weather Humidity of Tetouan city
Wind Speed	Feature	Continuous	Wind speed of Tetouan city
General Diffuse Flows	Feature	Continuous	General Diffuse Flows
Diffuse Flows	Feature	Continuous	Diffuse Flows
Zone 1 Power Consumption	Target	Continuous	Power cons. of zone 1 of Tetouan City
Zone 2 Power Consumption	Target	Continuous	Power cons. of zone 2 of Tetouan City
Zone 3 Power Consumption	Target	Continuous	Power cons, of zone 3 of Tetouan City

Για το συγκεκριμένο σύνολο δεδομένων, επιλέχθηκε να προβλεφθεί η μεταβλητή «Zone 1 Power Consumption» με μεταβλητές εισόδου «Temperature» και «Humidity». Φαίνεται από την απεικόνιση των δεδομένων ότι αυτές οι δυο μεταβλητές έχουν κάποια συσχέτιση με τη μεταβλητή εξόδου, η οποία θα «αναγνωριστεί» από το Σύστημα Ασαφούς Συμπερασμού.



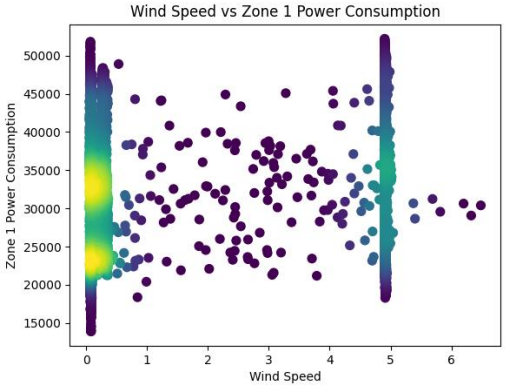
(α) Temperature vs Zone 1 Power Consumption



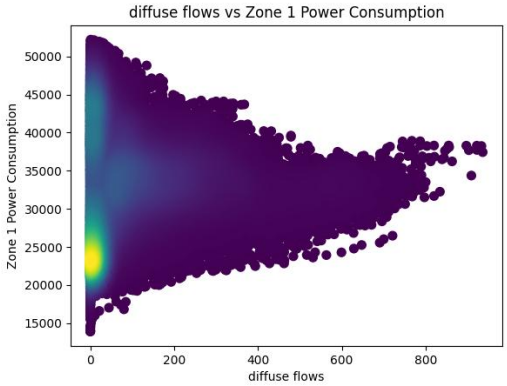
(β) Humidity vs Zone 1 Power Consumption

4.1.1 Subtractive Clustering (TSK)

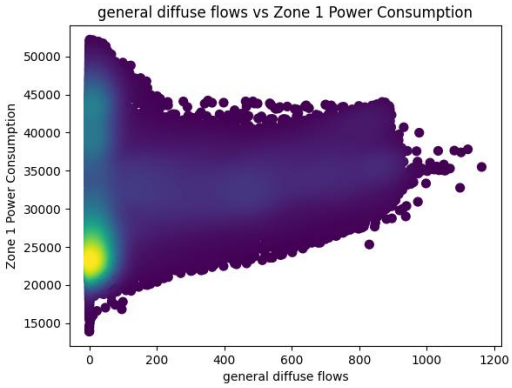
Στη συνέχεια θα παρουσιαστούν οι συστάδες που προέκυψαν από τον αλγόριθμο Subtractive Clustering, οι συναρτήσεις συμμετοχής των μεταβλητών εισόδου «Temperature» και «Humidity», οι συντελεστές στο τμήμα απόδοσης και η αποτελεσματικότητα του συστήματος που δημιουργήθηκε.



(γ) Window Speed vs Zone 1 Power Consumption

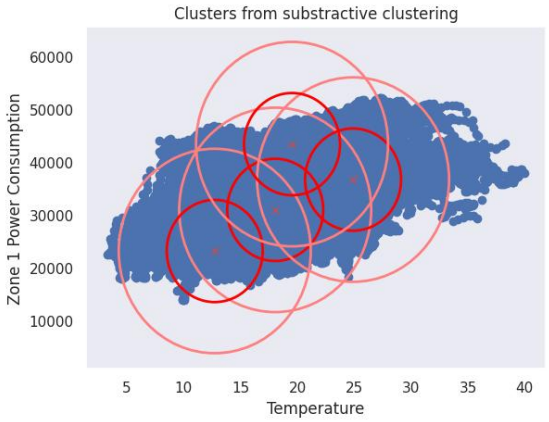


(δ) Diffuse Flows vs Zone 1 Power Consumption

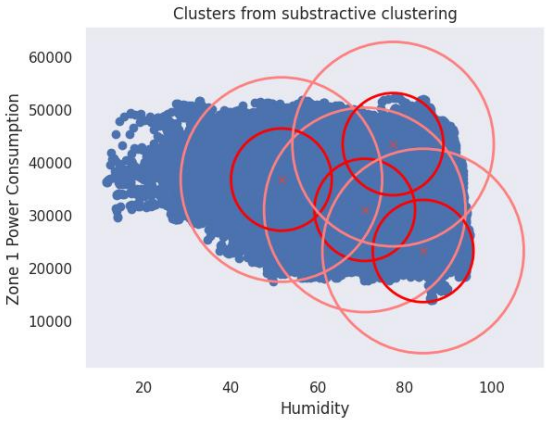


(ε) General Diffuse Flows vs Zone 1 Power Consumption

Εικόνα 4.1. Η ανάλυση του συνόλου δεδομένων, ως προς την κατανάλωση ενέργειας της περιοχής 1. Το θερμό χρώμα αντιστοιχεί σε υψηλή πυκνότητα στοιχείων, ενώ το ψυχρό χρώμα σε χαμηλή πυκνότητα.

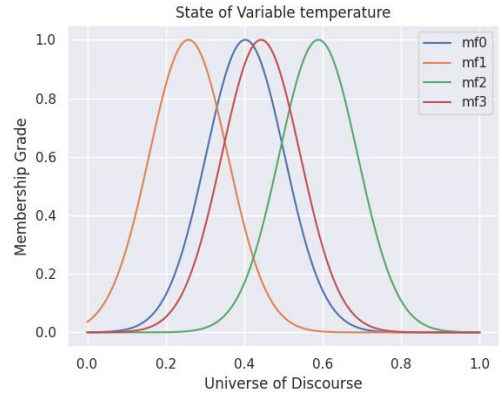


(α) Οι συστάδες απεικονισμένες στον χώρο «Temperature vs Zone 1 Power Consumption»

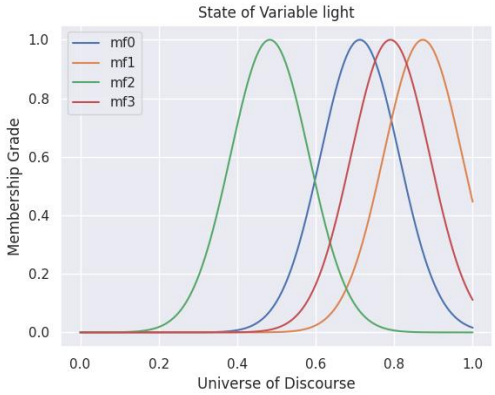


(β) Οι συστάδες απεικονισμένες στον χώρο «Humidity vs Zone 1 Power Consumption»

Εικόνα 4.2. Οι συστάδες που προέκυψαν από τον αλγόριθμο Subtractive Clustering. Το κόκκινο «x» αντιστοιχεί το κέντρο της συστάδας («μ» της κανονικής). Ο κόκκινος κύκλος αντιστοιχεί σε εύρος 1σ και ο ροζ κύκλος σε εύρος 2σ της κανονικής κατανομής.



(α) Συναρτήσεις συμμετοχής εισόδου «Temperature»



(β) Συναρτήσεις συμμετοχής εισόδου «Humidity»

Εικόνα 4.3. Μεταβλητές Εισόδου

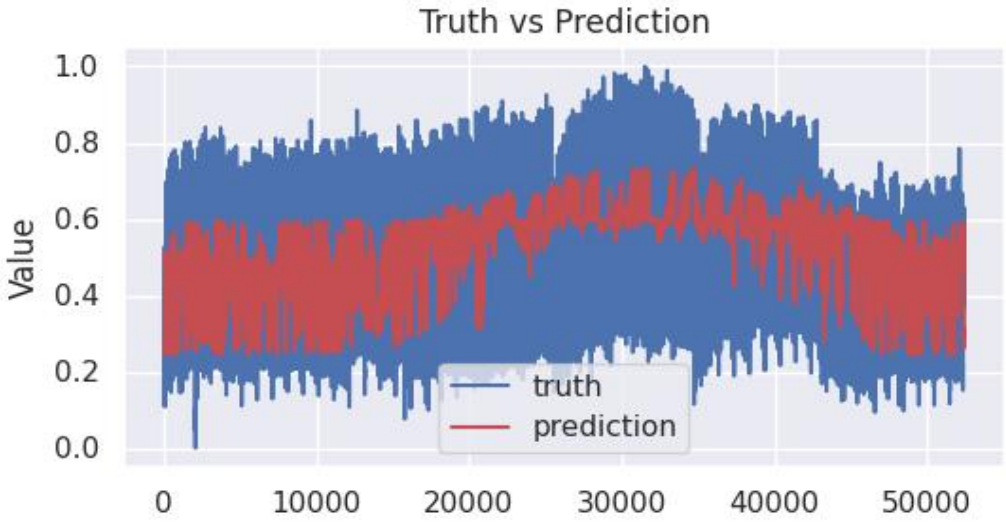
Απόδοση εξόδου TSK:
(Κάθε y_i αντιστοιχεί στην απόδοση του κανόνα R^i)

$$y_0 = 0.4461057$$

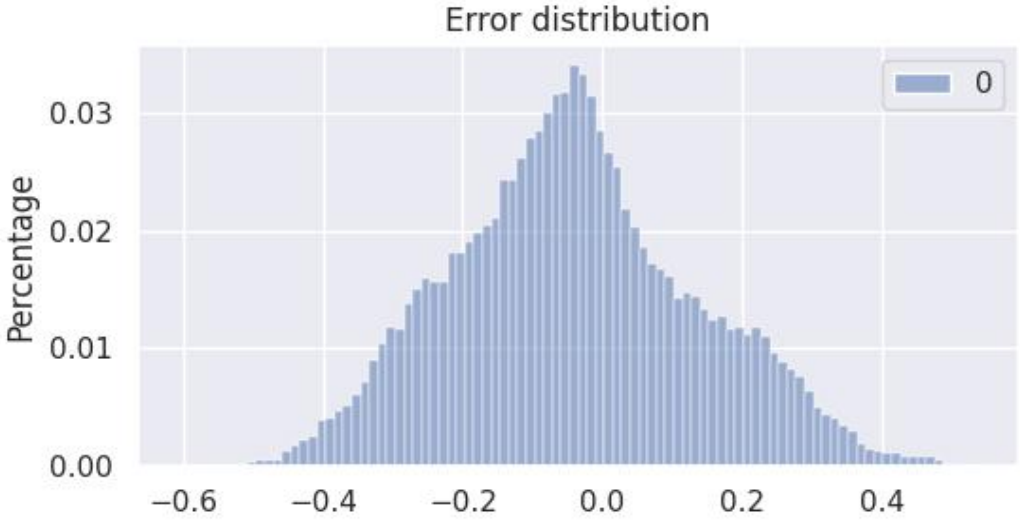
$$y_1 = 0.2433590$$

$$y_2 = 0.5955091$$

$$y_3 = 0.7704404$$



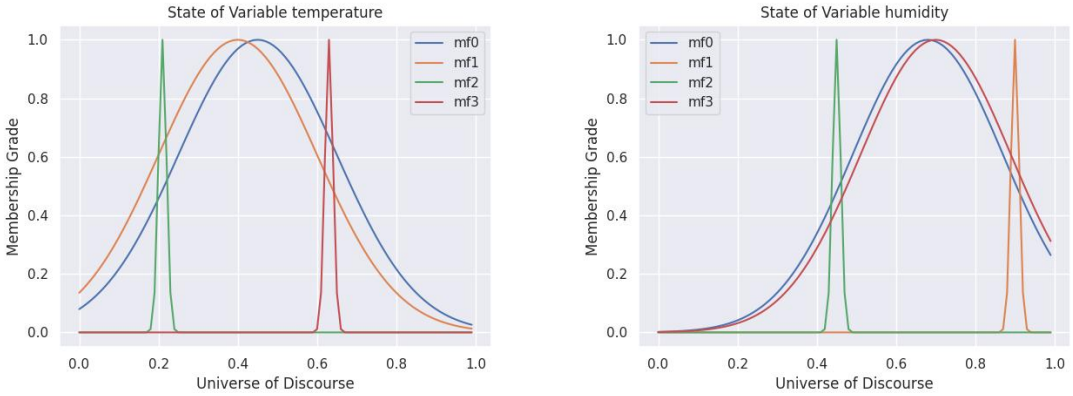
Εικόνα 4.4. Τιμές Πρόβλεψης vs Αληθινές Τιμές



Εικόνα 4.5. Κατανομή απόστασης τιμών πρόβλεψης από αληθινές τιμές

4.1.2 Subtractive Clustering με ANFIS (TSK)

Στη συνέχεια θα παρουσιαστεί το Σύστημα Ασαφούς Συμπερασμού ANFIS που προκύπτει από τις συστάδες που ευρέθησαν. Οι μεταβλητές εισόδου είναι αρχικά όπως παρουσιάστηκαν προηγουμένως και οι συντελεστές στο τμήμα απόδοσης επιλέγονται σύμφωνα με τον τυχαία σύμφωνα με τον αλγόριθμο του Subtractive Clustering. Θα παρουσιαστούν οι τελικές συναρτήσεις συμμετοχής των μεταβλητών εισόδου, οι τελικοί συντελεστές του τμήματος απόδοσης και η αποτελεσματικότητα του συστήματος.



(α) Είσοδος #1: «Temperature» (β) Είσοδος #2: «Humidity»

Εικόνα 4.6. Μεταβλητές Εισόδου

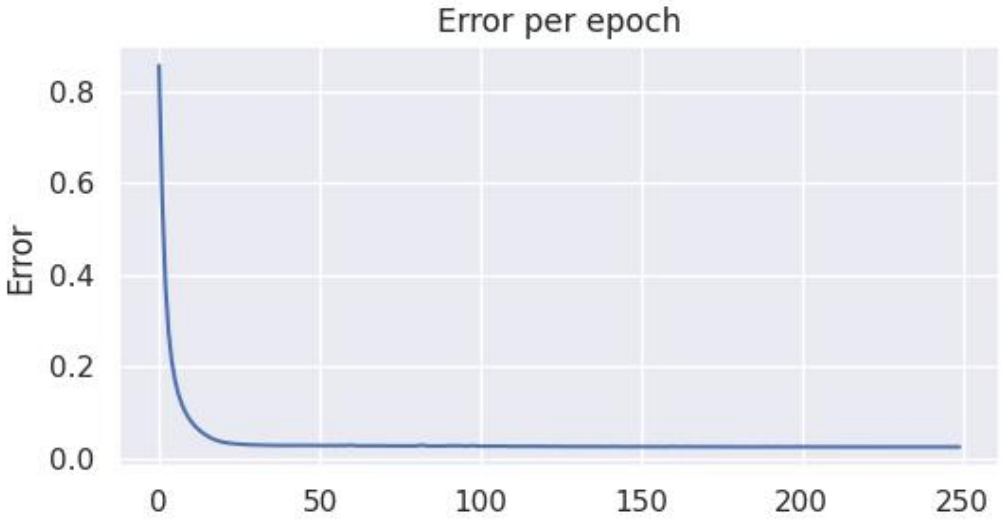
Απόδοση εξόδου TSK: (Κάθε y_i αντιστοιχεί στην απόδοση του κανόνα R^i)

$$y_0 = 0.4805 - 0.2781x_0 - 0.6589x_1$$

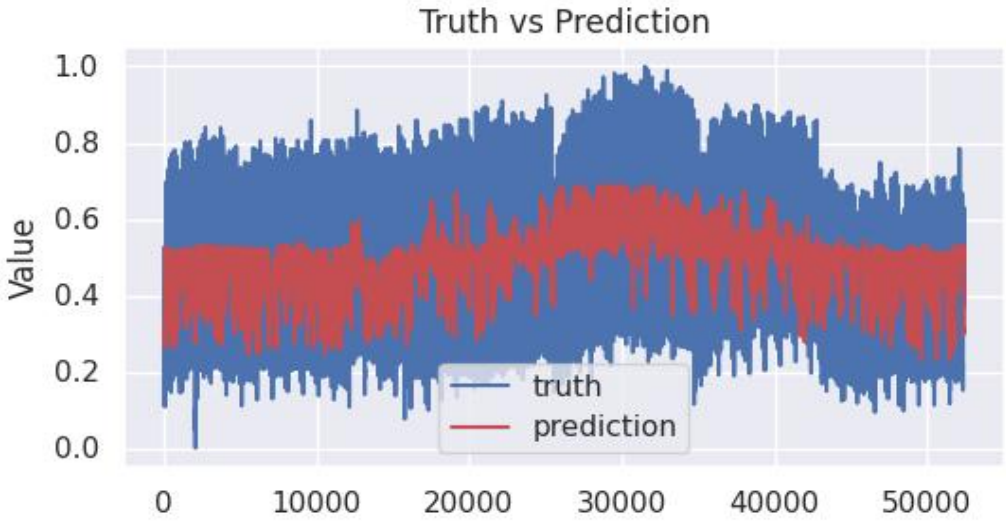
$$y_1 = 0.1812 + 1.4656x_0 + 1.6120x_1$$

$$y_2 = 0.5211 + 1.4310x_0 + 0.2922x_1$$

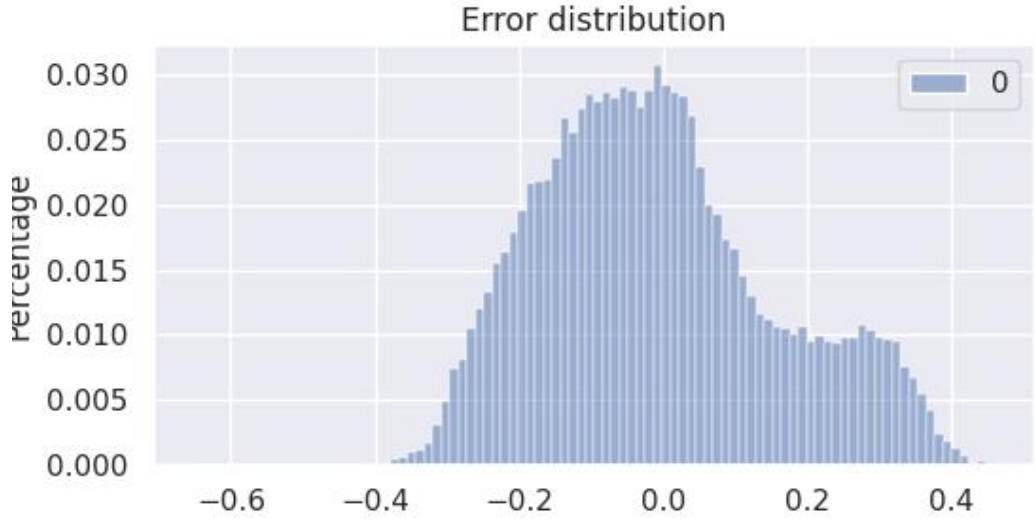
$$y_3 = 0.4291 + 1.4629x_0 + 0.1221x_1$$



Εικόνα 4.7. Σφάλμα «Mean Squared Error» ανά epoch



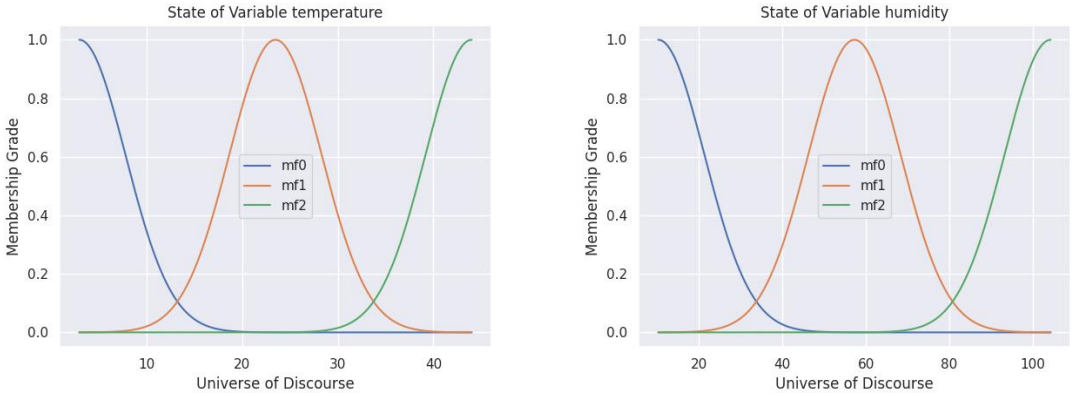
Εικόνα 4.8. Τιμές Πρόβλεψης vs Αληθινές Τιμές



Εικόνα 4.9. Κατανομή απόστασης τιμών πρόβλεψης από αληθινές τιμές

4.1.3 Grid Partitioning (Mamdani)

Στη συνέχεια θα παρουσιαστεί το Σύστημα Ασαφούς Συμπερασμού Mamdani που προέκυψε με τη χρήση της τεχνικής διαμοιρασμού του χώρου εισόδου Grid Partitioning.



(α) Είσοδος #1: «Temperature» (β) Είσοδος #2: «Humidity»

Εικόνα 4.10. Μεταβλητές Εισόδου

Απόδοση εξόδου TSK:
(Κάθε y_i αντιστοιχεί στην απόδοση του κανόνα R^i)

$$y_0 = 0.3281 - 1.1156x_0 + 0.5259x_1$$

$$y_1 = 1.2250 + 0.2998x_0 + 0.5492x_1$$

$$y_2 = 0.2651 + 0.5207x_0 + 0.7621x_1$$

$$y_3 = 0.1425 + 1.0524x_0 + 0.4322x_1$$

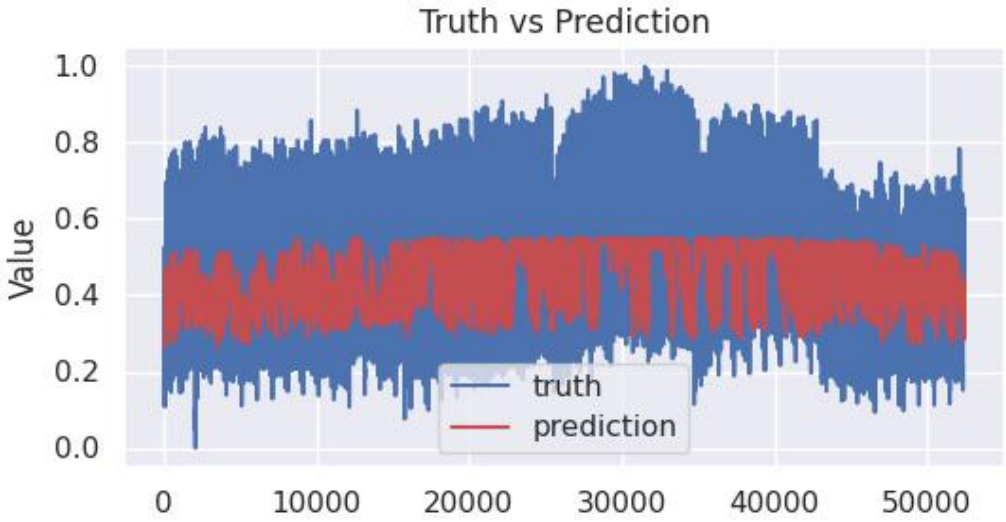
$$y_4 = 0.4643 - 0.0923x_0 - 0.8090x_1$$

$$y_5 = 1.7110 - 0.7218x_0 - 1.0650x_1$$

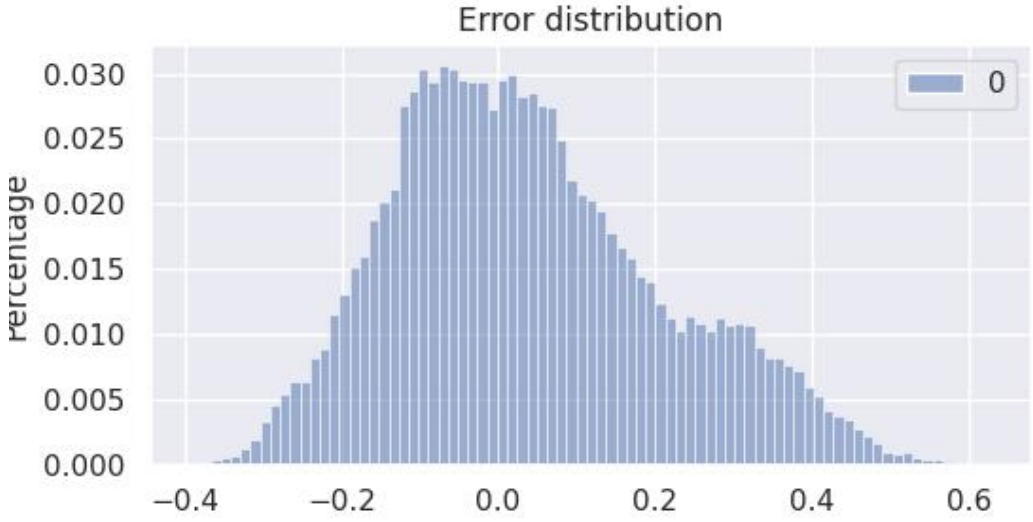
$$y_6 = -0.2810 + 0.2359x_0 + 0.4901x_1$$

$$y_7 = 1.1918 + 1.6999x_0 - 0.4591x_1$$

$$y_8 = -0.8835 - 0.0364x_0 + 1.7837x_1$$



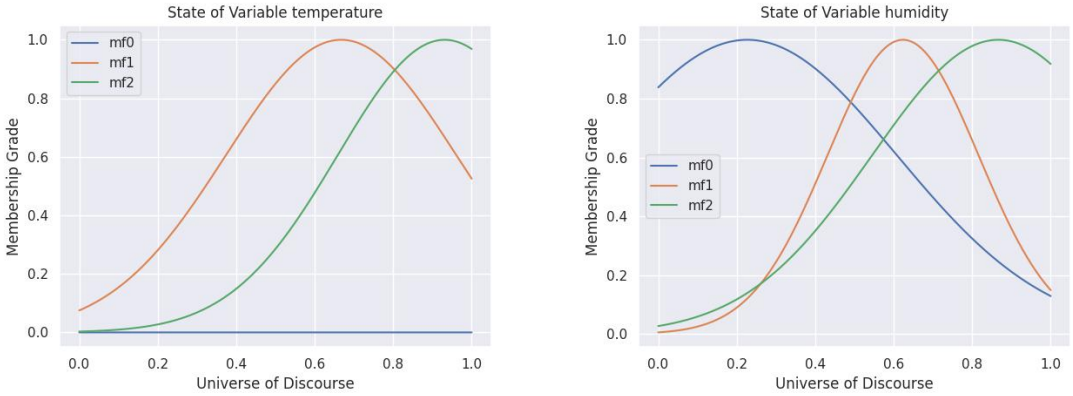
Εικόνα 4.11. Τιμές Πρόβλεψης vs Αληθινές Τιμές



Εικόνα 4.12. Κατανομή απόστασης τιμών πρόβλεψης από αληθινές τιμές

4.1.4 Grid Partitioning με ANFIS (TSK)

Στη συνέχεια θα παρουσιαστεί το Σύστημα Ασαφούς Συμπερασμού Tagaki-Sugeno-Kang που προέκυψε με τη χρήση της τεχνικής διαμοιρασμού του χώρου εισόδου Grid Partitioning. Οι συναρτήσεις συμμετοχής για την είσοδο είναι αρχικά ίδιες με την προηγούμενη περίπτωση. Με την εκπαίδευση θα αλλάξουν και φαίνεται στις παρακάτω εικόνες το αποτέλεσμα.



(α) Είσοδος #1: «Temperature» (β) Είσοδος #2: «Humidity»

Εικόνα 4.13. Μεταβλητές Εισόδου

Απόδοση εξόδου TSK:
(Κάθε y_i αντιστοιχεί στην απόδοση του κανόνα R^i)

$$y_0 = -1.1131 + 0.5160x_0 + 0.3182x_1$$

$$y_1 = 0.3303 + 0.5554x_0 + 1.2040x_1$$

$$y_2 = 0.5129 + 0.7611x_0 + 0.2573x_1$$

$$y_3 = 1.0801 + 0.4312x_0 + 0.1425x_1$$

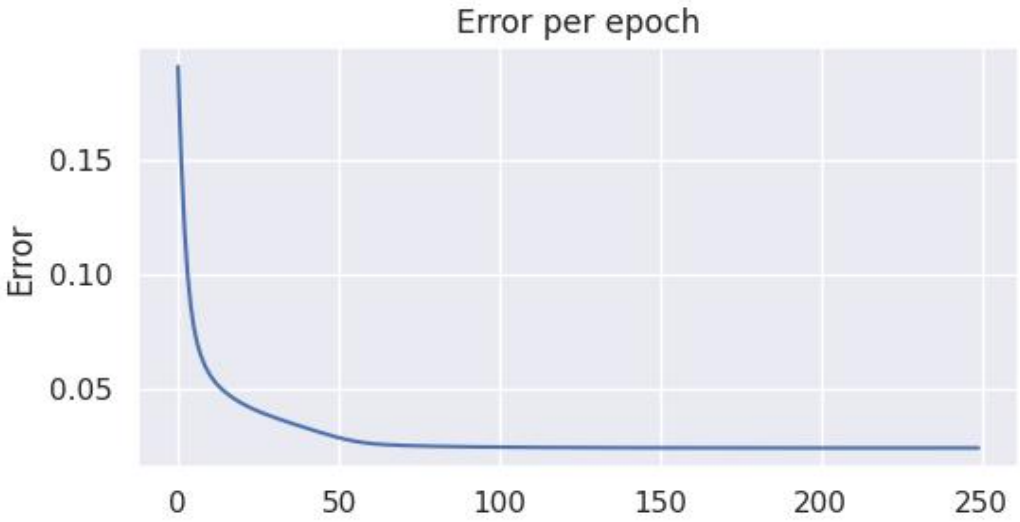
$$y_4 = -0.0798 - 0.8114x_0 + 0.4644x_1$$

$$y_5 = -0.7331 - 1.0648x_0 + 1.0922x_1$$

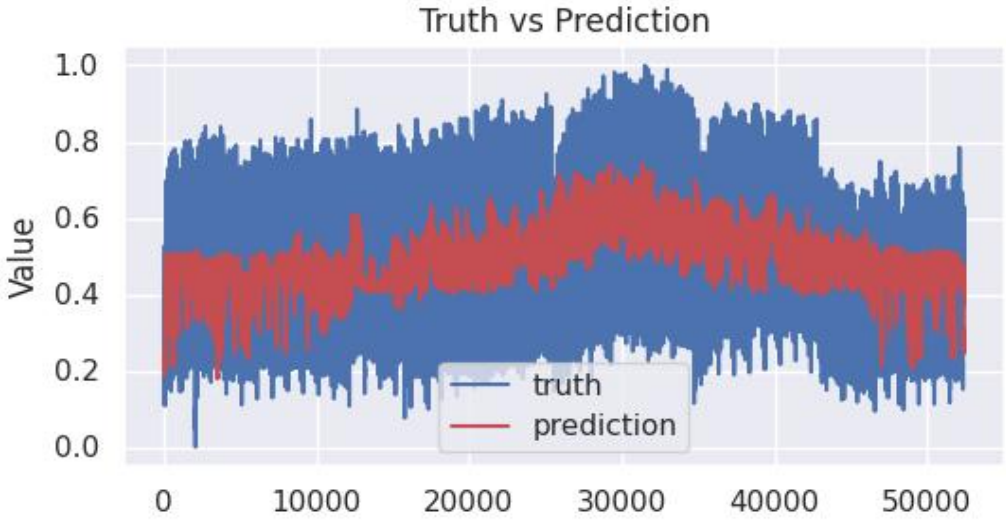
$$y_6 = 0.2399 + 0.4922x_0 - 0.2675x_1$$

$$y_7 = 1.7425 - 0.4621x_0 + 1.2292x_1$$

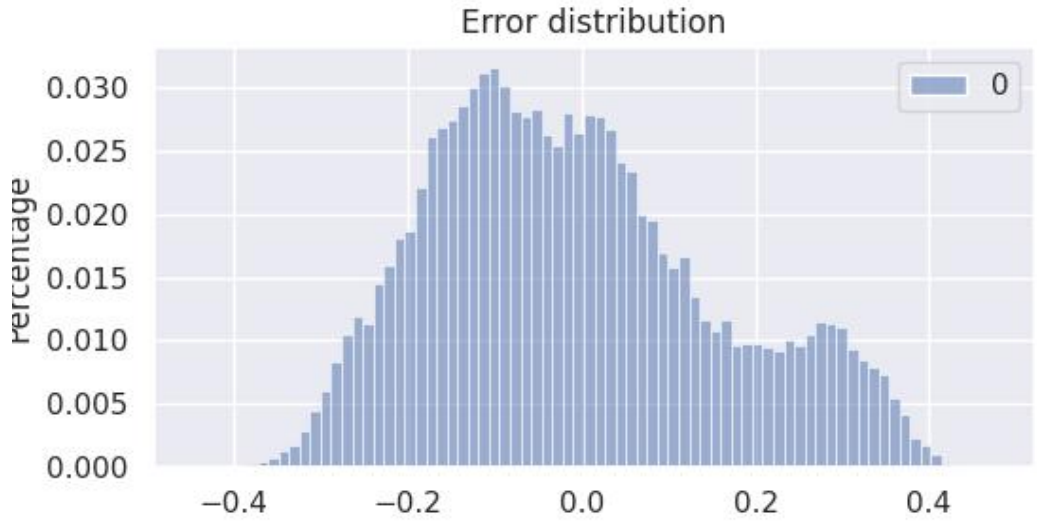
$$y_8 = -0.0419 + 1.7719x_0 - 0.8924x_1$$



Εικόνα 4.14. Σφάλμα «Mean Squared Error» ανά epoch



Εικόνα 4.15. Τιμές Πρόβλεψης vs Αληθινές Τιμές



Εικόνα 4.16. Κατανομή απόστασης τιμών πρόβλεψης από αληθινές τιμές

4.1.5 Σύνοψη

Πίνακας 4.3. Πίνακας σφάλματος MSE

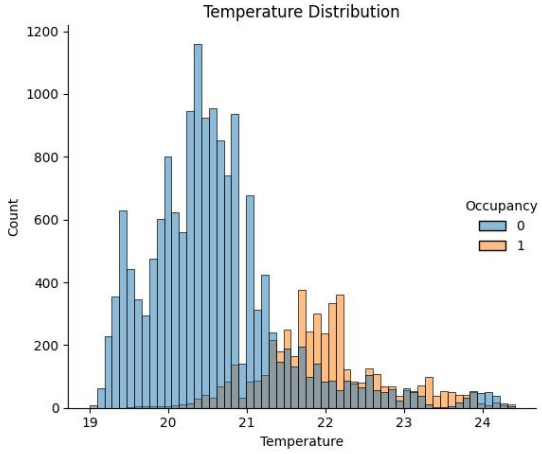
Method	MSE Loss
Subtractive Clustering (TSK)	0.03251
Subtractive Clustering with ANFIS	0.02671
Grid Partitioning (Mamdani)	0.03171
Grid Partitioning with ANFIS	0.02748

4.2 Office Occupancy

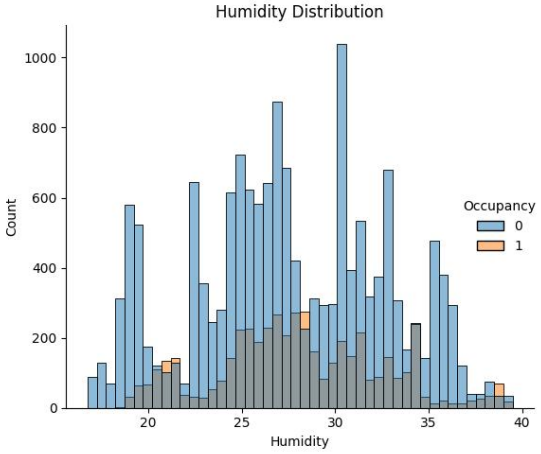
Πίνακας 4.4. Χαρακτηριστικά συνόλου δεδομένων

Name	Role	Type	Description
Date	Feature	Date	Each minute
Temperature	Feature	Continuous	Temperature
Humidity	Feature	Continuous	Humidity
Light	Feature	Continuous	Light
CO2	Feature	Continuous	CO2
HumidityRatio	Feature	Continuous	in kg-water-vapor/kg-air
Occupancy	Target	Categorical	0 for not occupied, 1 for occupied status

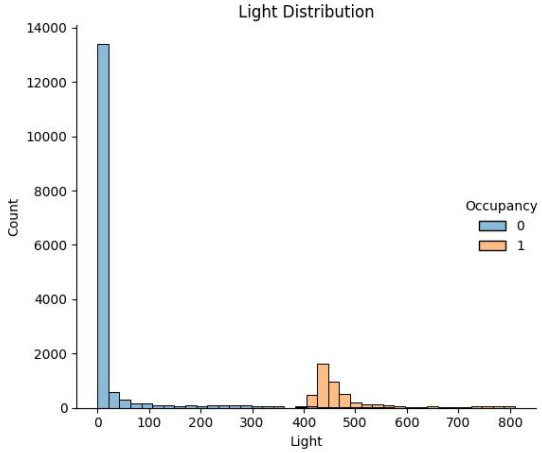
Για το συγκεκριμένο σύνολο δεδομένων, έγινε προσπάθεια πρόβλεψης της μεταβλητής «Occupancy» με μεταβλητές εισόδου «Temperature» και «Light». Το μοντέλο έχει ως αποτέλεσμα πραγματικό αριθμό, οποίος μετά στρογγυλοποιείται στο 0 ή στο 1.



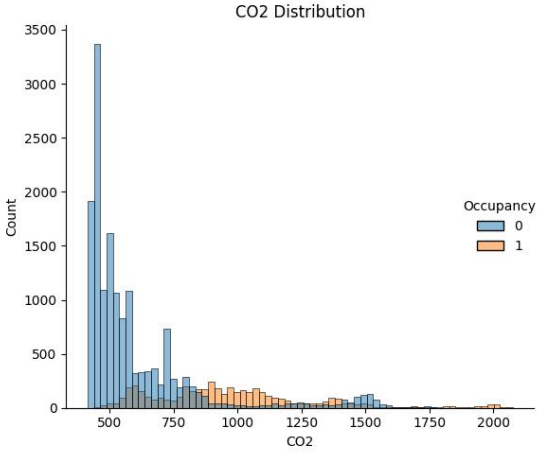
(α) Κατανομή «Temperature»



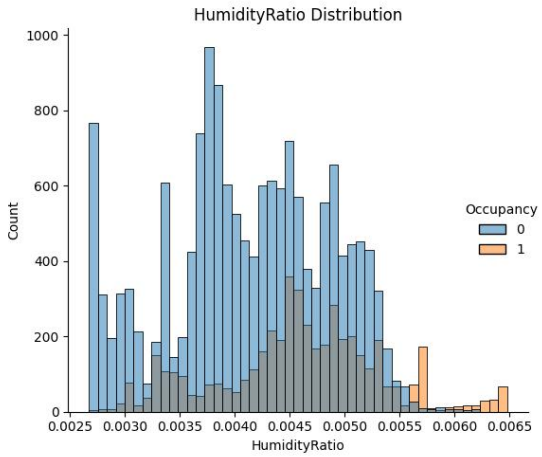
(β) Κατανομή «Humidity»



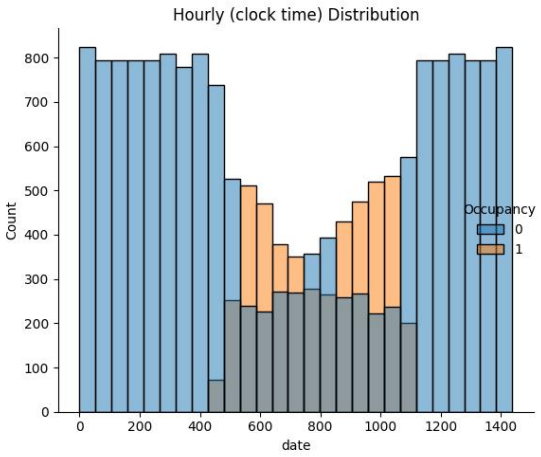
(γ) Κατανομή «Light»



(δ) Κατανομή «CO2»



(ε) Κατανομή «HumidityRatio»

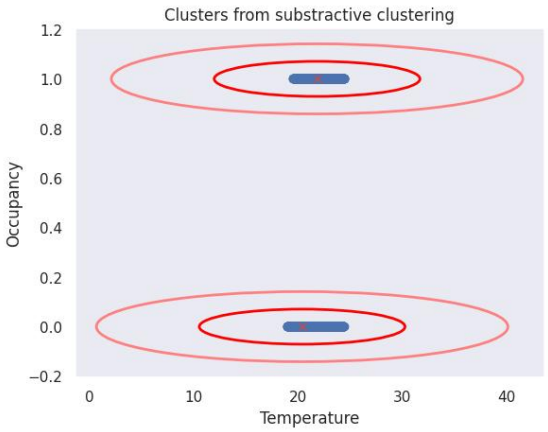


(στ) Κατανομή Ωρών

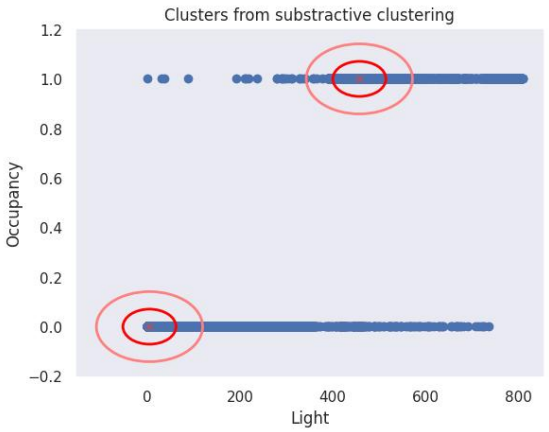
Εικόνα 4.17. Η ανάλυση του συνόλου δεδομένων «Office Occupancy», ως προς το αν ο χώρος είναι κατειλημμένος.

4.2.1 Subtractive Clustering (TSK)

Στη συνέχεια θα παρουσιαστούν οι συστάδες που προέκυψαν από τον αλγόριθμο Subtractive Clustering, οι συναρτήσεις συμμετοχής των μεταβλητών εισόδου «Temperature» και «Light», οι συντελεστές στο τμήμα απόδοσης και η αποτελεσματικότητα του συστήματος που δημιουργήθηκε.

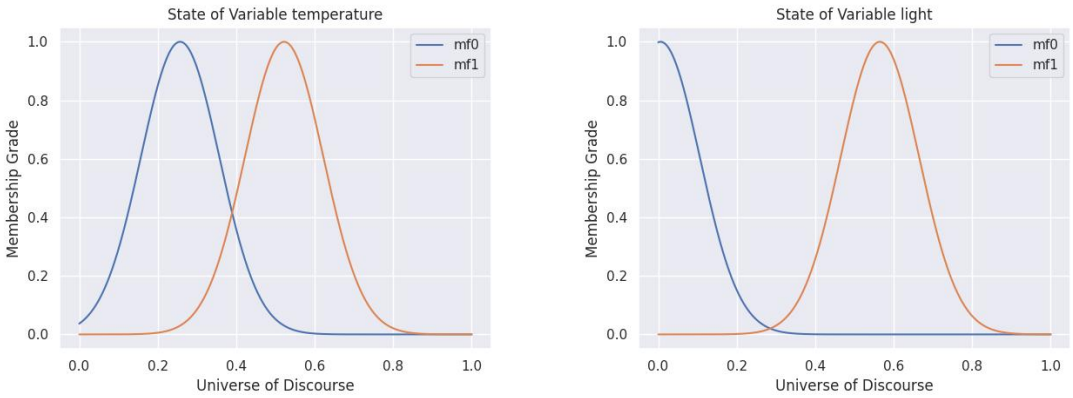


(α)



(β)

Εικόνα 4.18. Τα clusters που δημιουργήθηκαν. Απεικονίζονται ως προς τις μεταβλητές εισόδου.



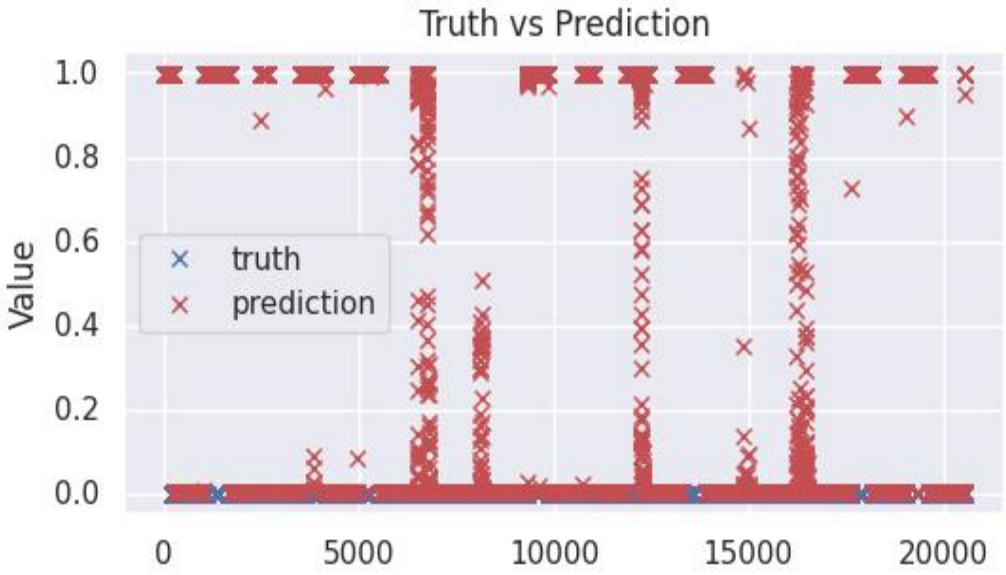
(α) Είσοδος #1: «Temperature» (β) Είσοδος #2: «Light»

Εικόνα 4.19. Μεταβλητές Εισόδου

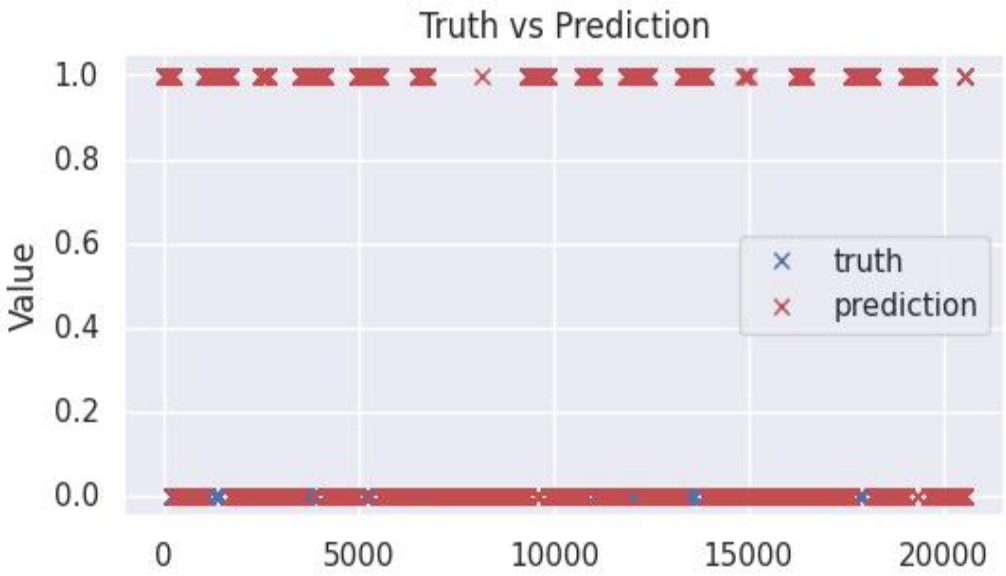
Απόδοση εξόδου TSK:
(Κάθε y_i αντιστοιχεί στην απόδοση του κανόνα R^i)

$$y_0 = 0$$

$$y_1 = 1$$



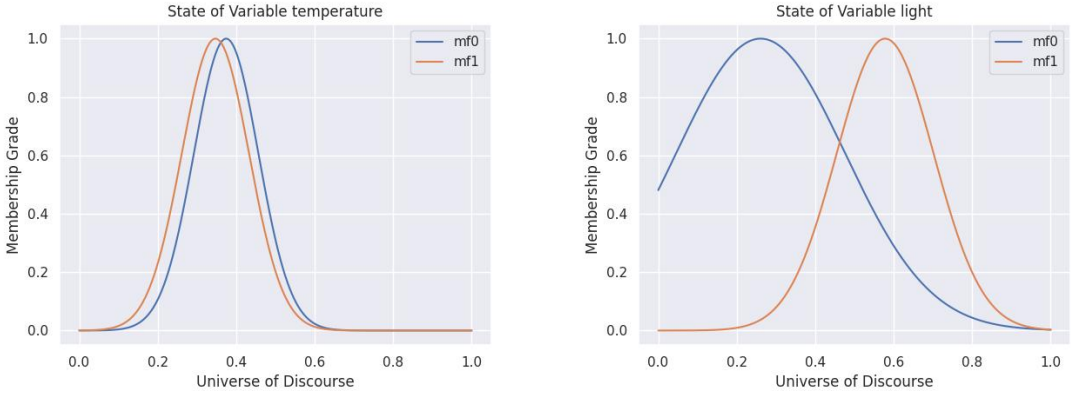
Εικόνα 4.20. Τιμές Πρόβλεψης vs Αληθινές Τιμές



Εικόνα 4.21. Τιμές Πρόβλεψης vs Αληθινές Τιμές (Στρογγυλοποιημένες)

4.2.2 Subtractive Clustering με ANFIS (TSK)

Στη συνέχεια θα παρουσιαστεί το Σύστημα Ασαφούς Συμπερασμού ANFIS που προκύπτει από τις συστάδες που ευρέθησαν. Οι μεταβλητές εισόδου είναι αρχικά όπως παρουσιάστηκαν προηγουμένως και οι συντελεστές στο τμήμα απόδοσης επιλέγονται σύμφωνα με τον τυχαία σύμφωνα με τον αλγόριθμο του Subtractive Clustering. Θα παρουσιαστούν οι τελικές συναρτήσεις συμμετοχής των μεταβλητών εισόδου, οι τελικοί συντελεστές του τμήματος απόδοσης και η αποτελεσματικότητα του συστήματος.



(α) Είσοδος #1: «Temperature» (β) Είσοδος #2: «Light»

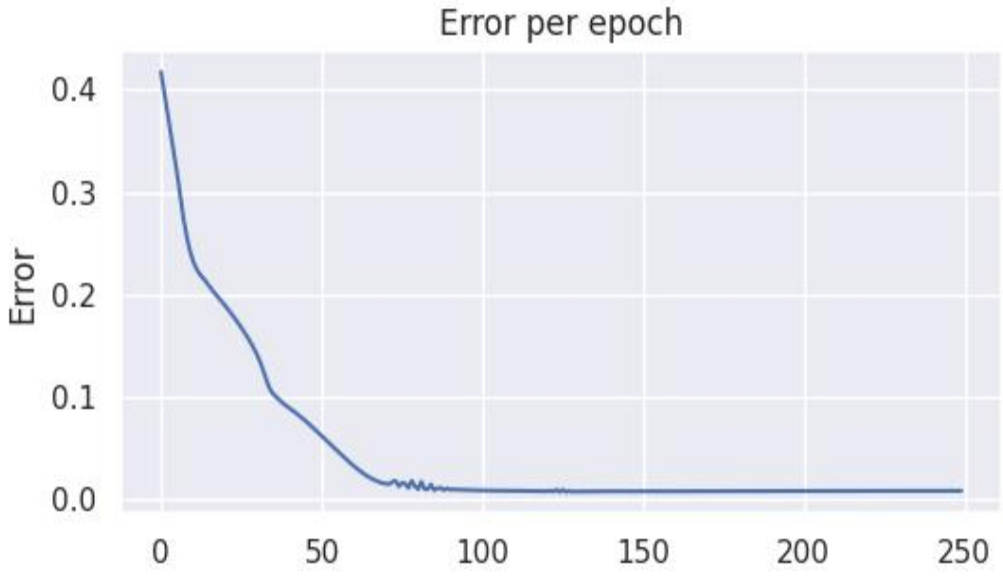
Εικόνα 4.22. Μεταβλητές Εισόδου

Απόδοση εξόδου TSK:

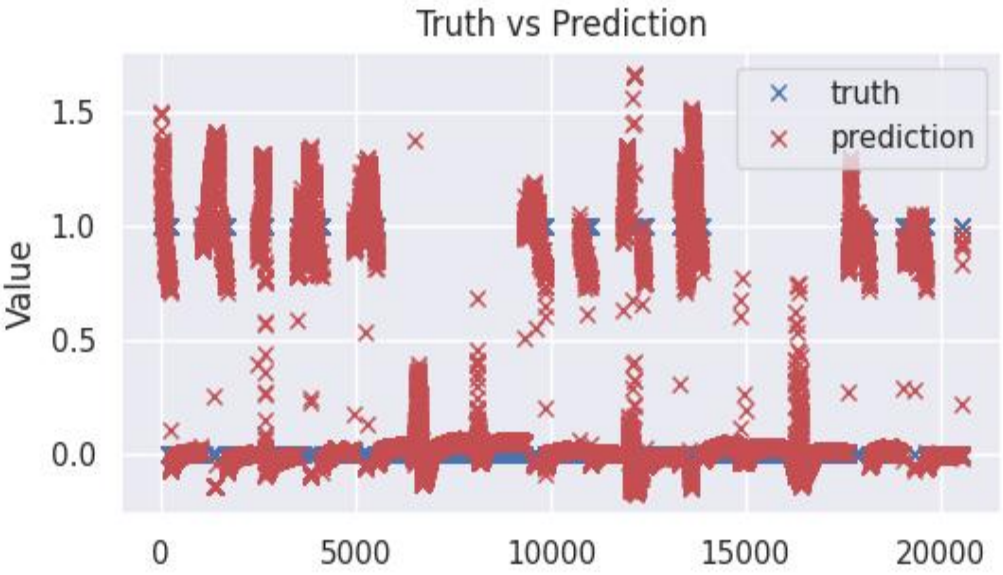
(Κάθε y_i αντιστοιχεί στην απόδοση του κανόνα R^i)

$$y_0 = 0.0673 - 0.2200x_0 - 0.2251x_1$$

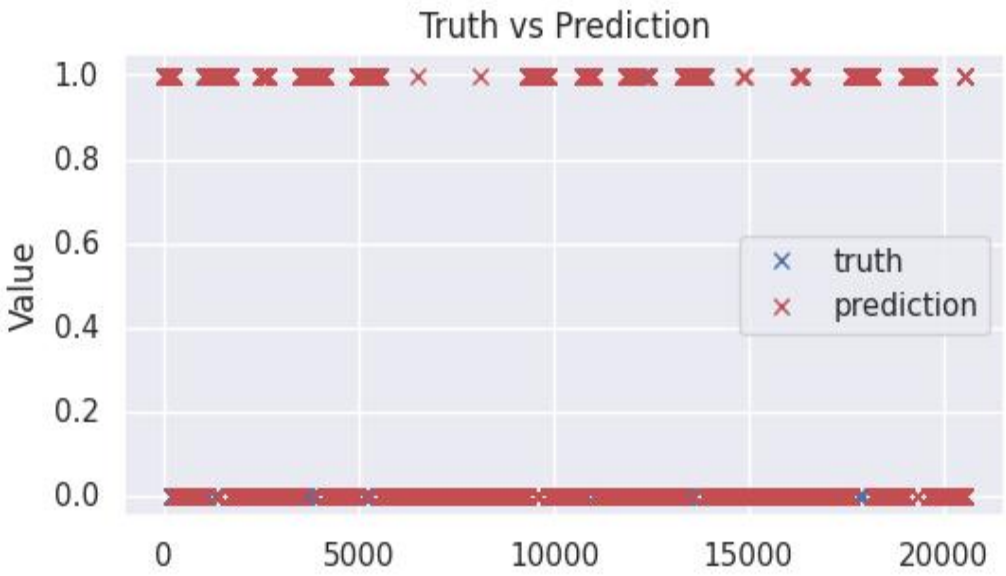
$$y_1 = 0.6825 + 0.8484x_0 + 0.9025x_1$$



Εικόνα 4.23. Σφάλμα «Mean Squared Error» ανά epoch



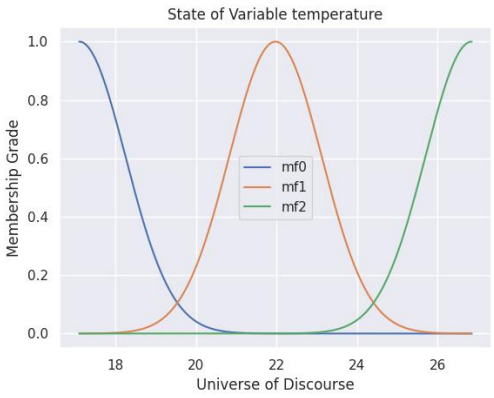
Εικόνα 4.24. Τιμές Πρόβλεψης vs Αληθινές Τιμές



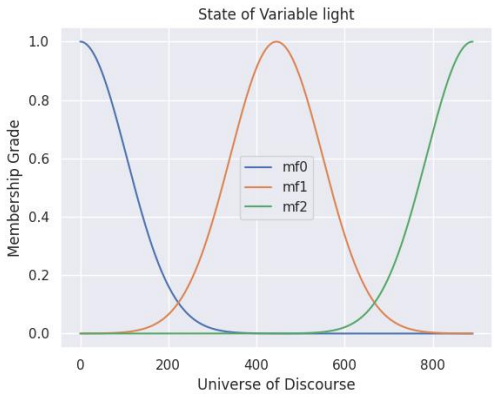
Εικόνα 4.25. Τιμές Πρόβλεψης vs Αληθινές Τιμές (Στρογγυλοποιημένες)

4.2.3 Grid Partitioning (Mamdani)

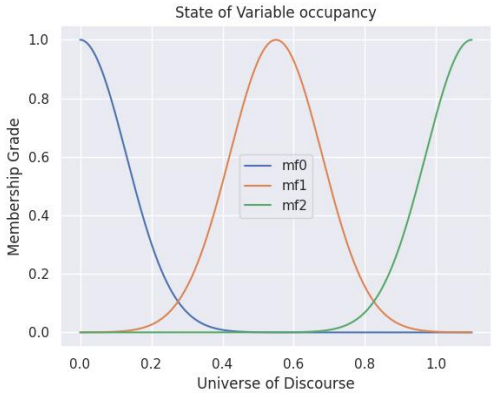
Στη συνέχεια θα παρουσιαστεί το Σύστημα Ασαφούς Συμπερασμού Mamdani που προέκυψε με τη χρήση της τεχνικής διαμοιρασμού του χώρου εισόδου Grid Partitioning.



(α) Είσοδος #1: «Temperature»

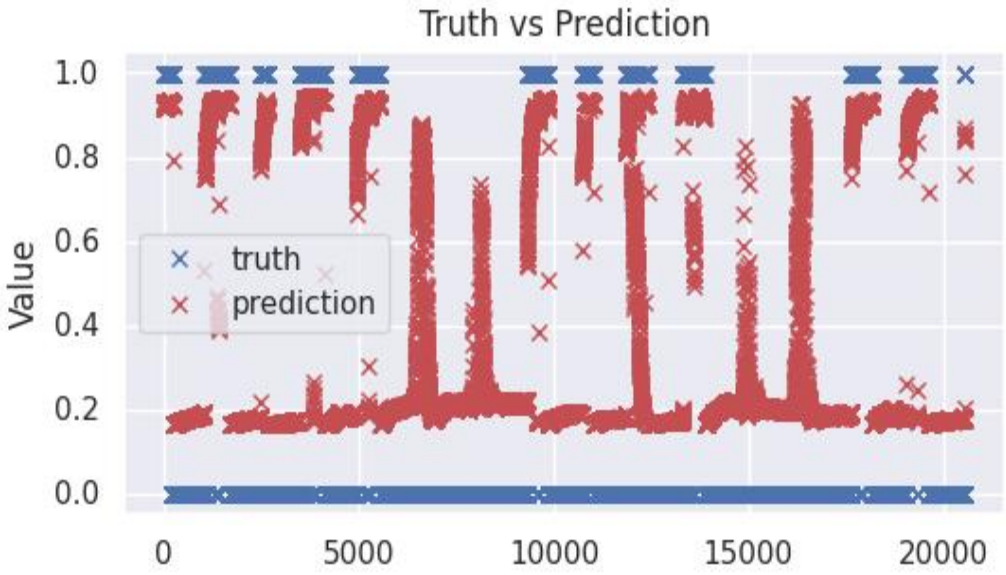


(β) Είσοδος #2: «Light»

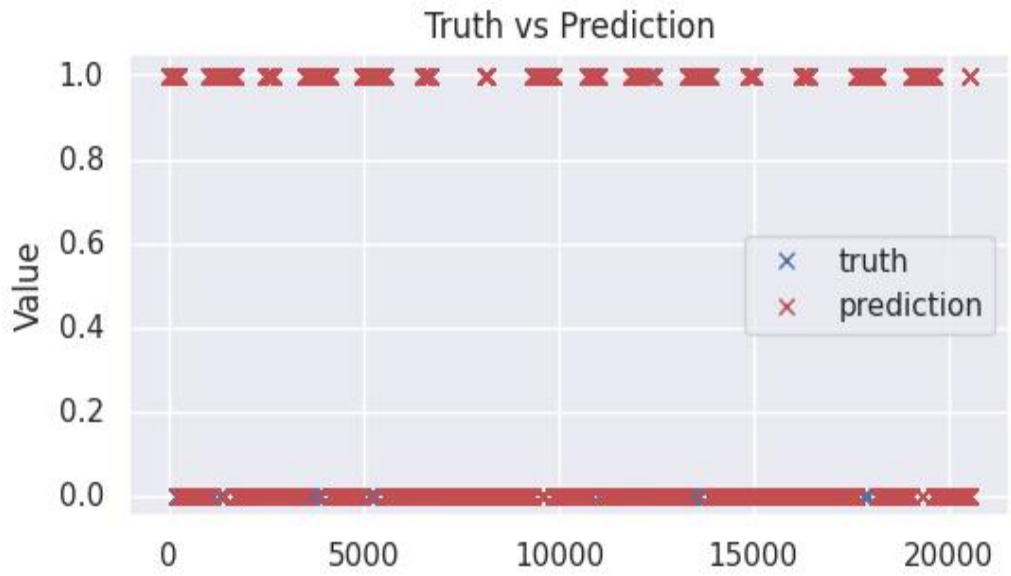


(γ) Έξοδος: «Occupancy»

Εικόνα 4.26. Μεταβλητές Εισόδου και Εξόδου



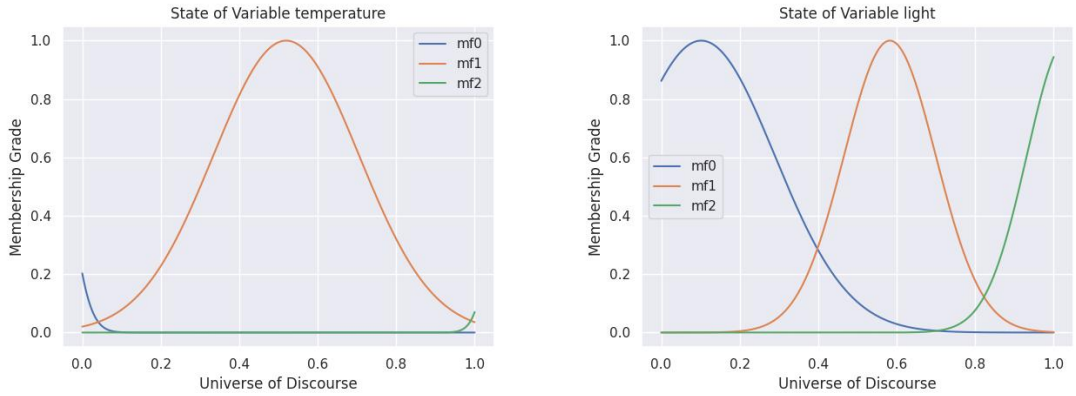
Εικόνα 4.27. Τιμές Πρόβλεψης vs Αληθινές Τιμές



Εικόνα 4.28. Τιμές Πρόβλεψης vs Αληθινές Τιμές (Στρογγυλοποιημένες)

4.2.4 Grid Partitioning με ANFIS (TSK)

Στη συνέχεια θα παρουσιαστεί το Σύστημα Ασαφούς Συμπερασμού Tagaki-Sugeno-Kang που προέκυψε με τη χρήση της τεχνικής διαμοιρασμού του χώρου εισόδου Grid Partitioning. Οι συναρτήσεις συμμετοχής για την είσοδο είναι αρχικά ίδιες με την προηγούμενη περίπτωση. Με την εκπαίδευση θα αλλάξουν και φαίνεται στις παρακάτω εικόνες το αποτέλεσμα.



(α) Είσοδος #1: «Temperature» (β) Είσοδος #2: «Light»

Εικόνα 4.29. Μεταβλητές Εισόδου

Απόδοση εξόδου TSK:
(Κάθε y_i αντιστοιχεί στην απόδοση του κανόνα R^i)

$$y_0 = -0.5951 - 0.9224x_0 - 0.6205x_1$$

$$y_1 = 0.9739 + 1.5264x_0 - 0.7610x_1$$

$$y_2 = 0.\mathrm{O}984 - 0.6573x_0 + 1.6017x_1$$

$$y_3 = 0.1766 - 0.6608x_0 + 1.6322x_1$$

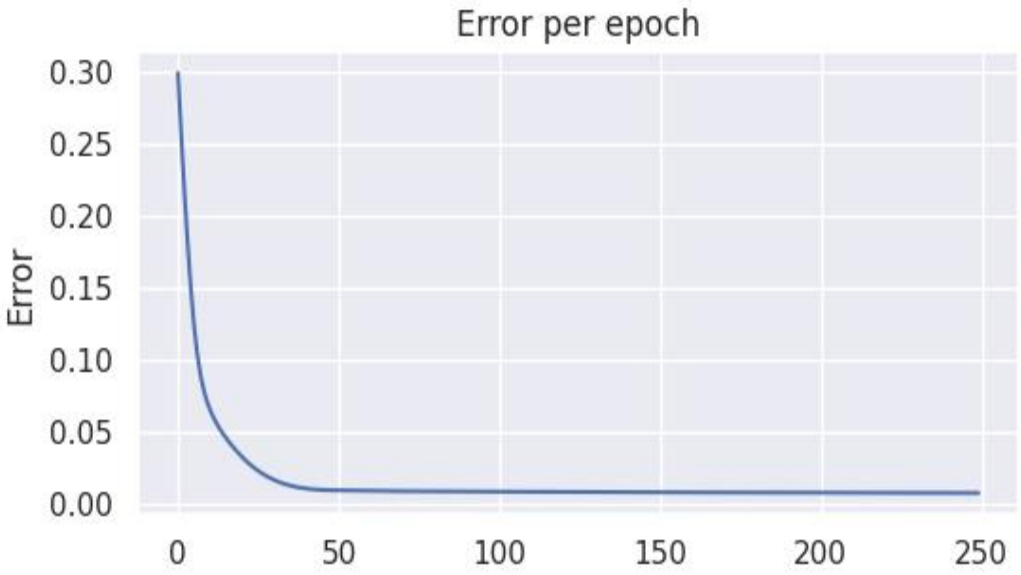
$$y_4 = 0.4433 + 0.2777x_0 + 1.2713x_1$$

$$y_5 = 1.3990 - 0.8680x_0 + 1.3835x_1$$

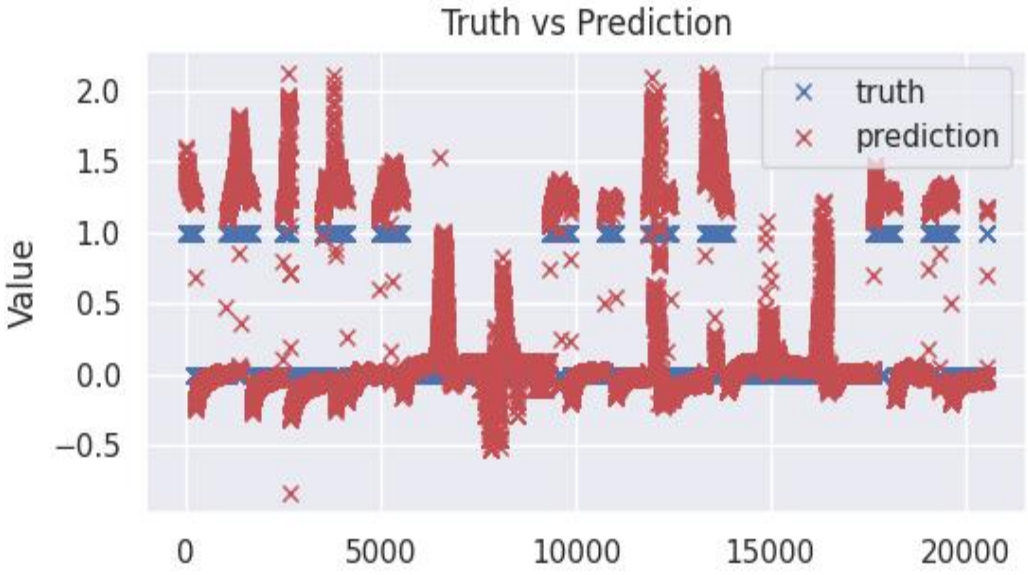
$$y_6 = 1.3280 + 1.5193x_0 - 0.6229x_1$$

$$y_7 = 1.1381 + 1.6921x_0 + 0.5869x_1$$

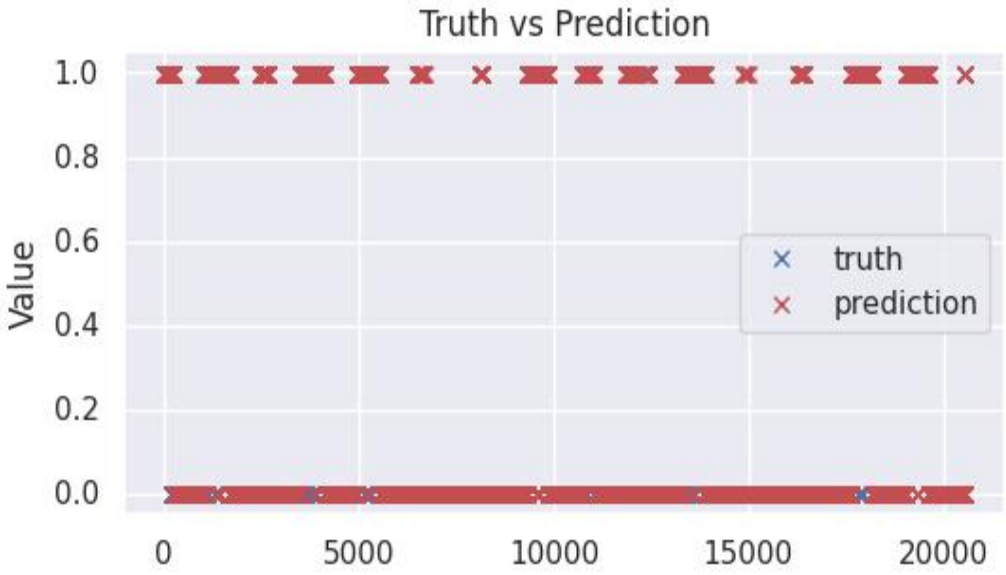
$$y_8 = -0.4290 - \mathrm{O}.8588x_0 - 0.9374x_1$$



Εικόνα 4.30. Σφάλμα «Mean Squared Error» ανά epoch



Εικόνα 4.31. Τιμές Πρόβλεψης vs Αληθινές Τιμές



Εικόνα 4.32. Τιμές Πρόβλεψης vs Αληθινές Τιμές (Στρογγυλοποιημένες)

4.2.5 Σύνοψη

Πίνακας 4.5. Πίνακας σφάλματος

Method	F1 Score	Precision Score	Recall Score	MSE Loss
SubClust (TSK)	0.91129	0.83778	0.99894	0.04483
SubClust with ANFIS	0.97463	0.95986	0.98986	0.01187
Grid Partitions (Mamdani)	0.92141	0.85505	0.99894	0.39288
Grid Partitions with ANFIS	0.94658	0.90082	0.99725	0.02594

4.3 Σχόλια

Παρατηρήσαμε και στα δύο σύνολα δεδομένων ότι η χρήση του ANFIS βοήθησε στην ακρίβεια των αποτελεσμάτων. Από την άλλη ο αλγόριθμος Subtractive Clustering φαίνεται να βοήθησε στο δεύτερο σύνολο, που ήταν πρόβλημα κατάταξης, σε σχέση με το Grid Partitioning.

Υπάρχει περιθώριο βελτίωσης όσον αφορά την επιλογή των μεταβλητών εισόδου, τον διαχωρισμό του συνόλου δεδομένων βάση κάποιων μοτίβων, π.χ. με βάση την εποχή, και στον τρόπο εκπαίδευσης του ANFIS, δηλαδή τις διάφορες τεχνικές που χρησιμοποιούνται στον κλάδο των τεχνητών νευρωνικών δικτύων για τη βελτιστοποίησή τους.

5

Συμπεράσματα

Κατά την εκπόνηση της εργασίας έγινε έντονα αισθητή η έλλειψη μιας ολοκληρωμένης υλοποίησης σε Python. Τα πακέτα Fuzzy Expert, Simpful και Scikit-Fuzzy, έχουν υλοποιήσει είτε το μοντέλο Mamdani είτε το μοντέλο Tagaki-Sugeno είτε και τα δύο, αλλά δεν έχουν ενσωματώσει το μοντέλο ANFIS και διαφέρουν πολύ στον τρόπο χρήσης μεταξύ τους. Οπότε είναι ξεκάθαρη η ανάγκη μιας υλοποίησης που να έχει ενσωματωμένο το ANFIS και να είναι απλή όσον αφορά τον κώδικα.

Το πακέτο που αναπτύχθηκε συμπληρώνει ένα σημαντικό κενό στην κοινότητα της Python ως προς τον κλάδο της Ασαφής Λογικής. Ο αλγόριθμος Subtractive Clustering και το νευροασαφές δίκτυο ANFIS, βοηθάνε στη βέλτιστη μοντελοποίηση και η έλλειψη μια υλοποίησης που να τρέχει στην τελευταία έκδοση της python αποτελούσε μεγάλο πρόβλημα για τα σύγχρονα πακέτα που θέλουν να αξιοποιήσουν τις παραπάνω μεθόδους. Βέβαια, υπάρχει περιθώριο βελτίωσης, ως προς τις τεχνικές της Ασαφής Λογικής και τις τεχνικές των Τεχνητών Νευρωνικών Δικτύων.

Παράρτημα Α

Κώδικας Παραγωγής Αποτελεσμάτων

Σε αυτό το παράρτημα παρουσιάζεται ο κώδικας Python που χρησιμοποιήθηκε για την παραγωγή των αποτελεσμάτων. Όπου το αρχείο data.csv, είναι το αντίστοιχο csv για τα σύνολα δεδομένων που χρησιμοποιήθηκαν.

A.1 Tetuan Power Consumption

```
# from datashader.mpl_ext import dsshow
from matplotlib import cm
from matplotlib.colors import Normalize
from scipy.interpolate import interpn
from scipy.stats import gaussian_kde

# from utils import saveFig
# import datashader as ds
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd


def normalize(df):
    return (df - df.min()) / (df.max() - df.min())


# def using_datashader(ax, x, y):
#
#    df = pd.DataFrame(dict(x=x, y=y))
```

```
#    dsartist = dsshow(
#        df,
#        ds.Point(”x”, ”y”),
#        ds.count(),
#        norm=”linear”,
#        aspect=”auto”,
#        ax=ax,
#    )
#
#    plt.colorbar(dsartist)


def density_scatter(x, y, ax=None, sort=True, bins=20, **kwargs):
    ”””
    Scatter plot colored by 2d histogram
    ”””
    if ax is None:
        fig, ax = plt.subplots()
    data, x_e, y_e = np.histogram2d(x, y, bins=bins, density=True)
    z = interpn(
        (0.5 * (x_e[1:] + x_e[:-1]), 0.5 * (y_e[1:] + y_e[:-1])),
        data,
        np.vstack([x, y]).T,
        method=”splinef2d”,
        bounds_error=False,
    )

    # To be sure to plot all data
    z[np.where(np.isnan(z))] = 0.0

    # Sort the points by density, so that the densest points are plotted
     ↪ last
    if sort:
        idx = z.argsort()
        x, y, z = x[idx], y[idx], z[idx]

    ax.scatter(x, y, c=z, **kwargs)
```

```
    norm = Normalize(vmin=np.min(z), vmax=np.max(z))
    cbar = fig.colorbar(cm.ScalarMappable(norm=norm), ax=ax)
    cbar.ax.set_ylabel("Density")

    return ax

def scatter_matrix(input_df: pd.DataFrame, output_df: pd.DataFrame):
    #fig, axs = plt.subplots(len(output_df.columns),len(input_df.columns))
    #if axs.ndim == 1:
    #    axs = axs.reshape(1,len(axs))
    for i, input_col in enumerate(input_df):
        for j, output_col in enumerate(output_df):
            #axs[j, i].scatter(input_df[input_col], output_df[output_col])
            plt.scatter(input_df[input_col], output_df[output_col])
            plt.show()

datadf = pd.read_csv("data.csv")

def scatter2(X: pd.DataFrame, Y: pd.DataFrame, axes=None, show=True):
    # ref: https://stackoverflow.com/questions/20105364/how-can-i-make-a-
    ↪ scatter-plot-colored-by-density
    ax = axes
    if axes is None:
        fig, ax = plt.subplots()
    xy = np.vstack([X.values, Y.values])
    z = gaussian_kde(xy)(xy)
    idx = z.argsort()
    x, y, z = X[idx], Y[idx], z[idx]
    ax.scatter(x,y, c=z, s=50)
    ax.set_xlabel(X.name)
    ax.set_ylabel(Y.name)
    ax.set_title(f"{X.name} vs {Y.name}")
    if show is True:
        fig.show()
    return ax

def proc_do(data):
    scatter2(data[0],data['Zone 1 Power Consumption'], show=False)
```

```
    plt.savefig(f"{data[0].name}-vs-{data[1].name}.jgp")

scatter2(datadf['Temperature'],datadf['Zone 1 Power Consumption'],
 ↪  show=False)
plt.savefig('temperature-to-power.jpg')
scatter2(datadf['Humidity'],datadf['Zone 1 Power Consumption'], show=False)
plt.savefig('Humidity-to-power.jpg')
scatter2(datadf['Wind Speed'],datadf['Zone 1 Power Consumption'],
 ↪  show=False)
plt.savefig('wind_speed-to-power.jpg')
scatter2(datadf['general diffuse flows'],datadf['Zone 1 Power Consumption'],
 ↪  show=False)
plt.savefig('general_diffuse_flows-to-power.jpg')
scatter2(datadf['diffuse flows'],datadf['Zone 1 Power Consumption'],
 ↪  show=False)
plt.savefig('general_diffue-to-power.jpg')

exit()

import matplotlib.style as mplstyle
mplstyle.use('fast')




input_df = datadf.drop(columns='Zone 1 Power Consumption')
output_df = datadf[['Zone 1 Power Consumption']]
scatter_matrix(input_df, output_df)

x = normalize(datadf["Temperature"])
y = normalize(datadf["Zone 1 Power Consumption"])

density_scatter(x, y)
plt.grid()
plt.xlabel("Normalized Temperature")
plt.ylabel("Normalized Zone 1 Power Consumption")
plt.savefig("fig1.jpg")
```

```
# xy = np.vstack([x, y])
# z = gaussian_kde(xy)(xy)
#
## Sort the points by density,
## so that the densest points are plotted last
# idx = z.argsort()
# x, y, z = x[idx], y[idx], z[idx]
#
# fig, ax = plt.subplots()
# ax.scatter(x, y, c=z, s=50)
# fig = plt.show()
# saveFig(fig)

x = normalize(datadf["Humidity"])
y = normalize(datadf["Zone 1 Power Consumption"])
density_scatter(x, y)
plt.grid()
plt.xlabel("Normalized Humidity")
plt.ylabel("Normalized Zone 1 Power Consumption")
plt.savefig("fig2.jpg")

pd.plotting.scatter_matrix(datadf, alpha=0.2)
plt.show()




#
# xy = np.vstack([x, y])
# z = gaussian_kde(xy)(xy)
#
## Sort the points by density,
## so that the densest points are plotted last
# idx = z.argsort()
# x, y, z = x[idx], y[idx], z[idx]
#
```

```
# fig, ax = plt.subplots()
# ax.scatter(x, y, c=z, s=50)
# fig = plt.show()
# saveFig(fig)
```

A.1.1 Subtractive Clustering χωρίς ANFIS

Η δημιουργία των clusters με την τεχνική Subtractive Clustering:

```
from fuzzyutils.clust import subclust_wrapper
import pandas as pd

datadf = pd.read_csv('data.csv')
subclust_wrapper(datadf, ['Temperature','Humidity'], ['Zone 1 Power
 ↪  Consumption'], savefig=”subclust-”, radiusA=0.4)
```

```
import shelve
from fuzzyutils.membership import *
import torch

with shelve.open('data') as db:
    C = db['Cnorm']
    S = db['Snorm']
    X = db['Xnorm']
    datadf = db['datadf']
    col = db['col']

temperature = {'name': 'temperature'}
humidity = {'name': 'light'}
power = {'name': 'power'}

temperature['memfs'] = clusters_gauss(C[:,0], S[0,0])
humidity['memfs'] = clusters_gauss(C[:, 1], S[0,0])
power['memfs'] = PolyMF(0, len(C))
power['memfs'].set_bias(C[:,-1])

plot_fuzzy(temperature)
fig = plt.gcf()
fig.savefig('notrain-temperature.jpg')
```

```
plt.show()
plot_fuzzy(humidity)
fig = plt.gcf()
fig.savefig('notrain-humidity.jpg')
plt.show()
plot_poly(power)
fig = plt.gcf()
fig.savefig('notrain-power.jpg')
plt.show()

model = AnfisNet([temperature, humidity], power, rules='clustering')

DTYPE = torch.float32
x = torch.tensor(X[:,:-1], dtype=DTYPE)
y = torch.tensor(X[:,-1:], dtype=DTYPE)

with torch.no_grad():
    y_pred = model(x)

with shelve.open('data') as db:
    db['notrain-datadf'] = datadf
    db['notrain-true'] = y.numpy()
    db['notrain-pred'] = y_pred.numpy()
```

```
import shelve
import matplotlib.pyplot as plt
import seaborn as sns
from fuzzyutils.plots import *
from sklearn.metrics import mean_squared_error

sns.set_theme()

with shelve.open('data') as db:
    datadf = db['notrain-datadf']
    pred = db['notrain-pred']
    true = db['notrain-true']
```

```
plotPrediction(true,pred)
fig = plt.gcf()
fig.savefig('notrain-prediction.jpg')
plt.show()

errorDistribution(true-pred)
fig = plt.gcf()
fig.savefig('notrain-distribution.jpg')
plt.show()

with open(”notrain-mse.txt”, ”w”) as file:
    file.write(str(mean_squared_error(true,pred)))
```

A.1.2 Subtractive Clustering με ANFIS

```
from sklearn.preprocessing import MinMaxScaler
import pandas as pd
import shelve
import torch
from copy import deepcopy
from fuzzyutils.membership import *
from fuzzyutils.plots import *

with shelve.open('data') as db:
    C = db['Cnorm']
    S = db['Snorm']
    X = db['Xnorm']
    datadf = db['datadf']
    col = db['col']

temperature = {'name': 'temperature'}
humidity = {'name': 'humidity'}
power = {'name': 'power'}

temperature['memfs'] = clusters_gauss(C[:,0], S[0,0])
light['memfs'] = clusters_gauss(C[:, 1], S[0,0])
power['memfs'] = PolyMF(2, len(C))
power['memfs'].set_bias(C[:,-1])
```

```
model = AnfisNet([temperature, humidity], power, rules='clustering')

DTYPE = torch.float32
x = torch.tensor(X[:,:-1], dtype=DTYPE)
y = torch.tensor(X[:,-1:], dtype=DTYPE)
n = int(len(x)*0.7)
all_dataset = torch.utils.data.TensorDataset(x,y)
train_dataset = torch.utils.data.TensorDataset(x[:n],y[:n])
test_dataset = torch.utils.data.TensorDataset(x[n:],y[n:])
dataloader = torch.utils.data.DataLoader(train_dataset, batch_size=2048,
 ↪  shuffle=False)

errors = []  # Keep a list of these for plotting afterwards
# optimizer = torch.optim.Adam(model.parameters(), lr=1e-4)
loss = torch.nn.MSELoss()
#optimizer = torch.optim.SGD(model.parameters(), lr=0.0005)
#optimizer = torch.optim.Rprop(model.parameters(), lr=0.0005)
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)
epoch = 250
min_state = deepcopy(model.state_dict())
min_error = float('inf')
min_y_pred = None

for t in range(epoch):
    # Process each mini-batch in turn:
    with torch.autograd.set_detect_anomaly(True):
        for x, y_actual in dataloader:
            optimizer.zero_grad()
            y_pred = model(x)
            #model.fit_coeffs(x, y_pred)
            l = loss(y_pred, y_actual) # get loss
            l.backward()     # backwards propagation based on loss function
            optimizer.step() # Update parameters based on backward pass

    with torch.no_grad():
        x,y_actual = test_dataset[:]
        y_pred = model(x)
        mse = loss(y_pred, y_actual)
```

```
        if mse < min_error:
            min_y_pred = y_pred.clone()
            min_state = deepcopy(model.state_dict())
            min_error = l
        errors.append(mse)
    # Epoch ending, so now fit the coefficients based on all data:
    #x, y_actual = data.dataset.tensors
    #with torch.no_grad():
    #    model.fit_coeff(x, y_actual)
    # Get the error rate for the whole batch:
    # Print some progress information as the net is trained:
        if t/epoch*100 % 2 == 0:
            print(f"epoch: {t/epoch*100:5.2f} %, MSE Loss is: {mse:.8f}")

errorEpochs(errors)
fig = plt.gcf()
fig.savefig('train-epochs.jpg')
plt.show()

model.load_state_dict(min_state)
with torch.no_grad():
    x, y_actual = all_dataset[:]
    y_pred = model(x)

for var in model.in_variables_def:
    plot_fuzzy(var)
    fig = plt.gcf()
    fig.savefig(f'train-{var["name"]}.jpg')
    plt.show()

plot_poly(model.out_variable_def)
fig = plt.gcf()
fig.savefig('train-occupancy.jpg')
plt.show()

with shelve.open('data') as db:
    db['train-datadf'] = datadf
    db['train-pred'] = y_pred.numpy()
```

```
    db['train-true'] = y_actual.numpy()
```

```
import shelve
import matplotlib.pyplot as plt
import seaborn as sns
from fuzzyutils.plots import *
from sklearn.metrics import mean_squared_error

sns.set_theme()

with shelve.open('data') as db:
    datadf = db['train-datadf']
    pred = db['train-pred']
    true = db['train-true']

plotPrediction(true,pred)
fig = plt.gcf()
fig.savefig('train-prediction.jpg')
plt.show()

errorDistribution(true-pred)
fig = plt.gcf()
fig.savefig('train-distribution.jpg')
plt.show()

with open("train-mse.txt", "w") as file:
    file.write(str(mean_squared_error(true,pred)))
```

A.1.3 Grid Partitioning χωρίς ANFIS

```
# This file is licensed under GPLv3
import pandas as pd
import numpy as np
from simpful import FuzzySystem
import multiprocessing as mp
import shelve
from fuzzyutils.membership import *
from fuzzyutils.plots import *
from fuzzyutils.simpful import *
```

```
datadf = pd.read_csv(”data.csv”)
temperature_df = datadf[”Temperature”]
humidity_df = datadf[”Humidity”]
power_df = datadf[”Zone 1 Power Consumption”]

temperature = {'name': 'temperature', 'domain': (temperature_df.min()*.9,
 ↪  temperature_df.max()*1.1)}
humidity = {'name': 'humidity', 'domain': (humidity_df.min()*.9,
 ↪  humidity_df.max()*1.1)}
power = {'name': 'power', 'domain': (power_df.min()*.9, power_df.max()*1.1)}

temperature['memfs'] = grid_gauss(3, domain=temperature['domain'])
humidity['memfs'] = grid_gauss(3, domain=humidity['domain'])
power['memfs'] = grid_gauss(5, domain=power['domain'])

plot_fuzzy(temperature)
fig = plt.gcf()
fig.savefig('notrain-temperature.jpg')
plt.show()
plot_fuzzy(humidity)
fig = plt.gcf()
fig.savefig('notrain-humidity.jpg')
plt.show()
plot_fuzzy(power)
fig = plt.gcf()
fig.savefig('notrain-power.jpg')
plt.show()

fis = FuzzySystem()
lv = variable_to_simpful(temperature)
fis.add_linguistic_variable(name=lv._concept, LV=lv)
lv = variable_to_simpful(humidity)
fis.add_linguistic_variable(name=lv._concept, LV=lv)
lv = variable_to_simpful(power)
fis.add_linguistic_variable(name=lv._concept, LV=lv)
```

```
rules_array = gen_grid_rules([temperature, humidity, power],
 ↪  datadf[['Temperature','Humidity','Zone 1 Power Consumption']].values)
rules_df = pd.DataFrame(
    rules_array, columns=[”Temperature”, ”Humidity”, ”Power”]
)

rules_fuzzy = []
for idx, rule in rules_df.iterrows():
    rules_fuzzy.append(
        f”IF (temperature IS mf{rule['Temperature']}) AND (humidity IS
         ↪  mf{rule['Humidity']}) THEN (power IS mf{rule['Power']})”
    )

fis.add_rules(rules_fuzzy)

pool_data = datadf[[”Temperature”, ”Humidity”]].values


def process_func(row: np.ndarray):
    fis.set_variable(”temperature”, row[0])
    fis.set_variable(”humidity”, row[1])
    return fis.inference()[”power”]


pool_cpus = mp.cpu_count() - 1
with mp.Pool(processes=pool_cpus) as pool:
    result = pool.map(process_func, pool_data)

with shelve.open(”data”) as db:
    db[”notrain-pred”] = np.array(result)
    db[”notrain-true”] = datadf[['Zone 1 Power
     ↪  Consumption']].values.squeeze()
    db[”notrain-fis”] = fis
    db[”notrain-datadf”] = datadf

with open(”notrain-rules.txt”, ”w”) as file:
    file.write(str(fis.get_rules()))
```

```
import shelve
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import MinMaxScaler
from fuzzyutils.plots import *

sns.set_theme()

with shelve.open('data') as db:
    datadf = db['notrain-datadf']
    pred = db['notrain-pred']
    true = db['notrain-true']

# Scale output from Simpful
scaler = MinMaxScaler()
scaler.fit(true.reshape(-1,1))
true = scaler.transform(true.reshape(-1,1))
pred = scaler.transform(pred.reshape(-1,1))

plotPrediction(true,pred)
fig = plt.gcf()
fig.savefig('notrain-prediction.jpg')
plt.show()

errorDistribution(true-pred)
fig = plt.gcf()
fig.savefig('notrain-distribution.jpg')
plt.show()

with open(”notrain-mse.txt”, ”w”) as file:
    file.write(str(mean_squared_error(true,pred)))
```

A.1.4 Grid Partitioning με ANFIS

```
from sklearn.preprocessing import MinMaxScaler
import pandas as pd
import shelve
```

```
import torch
from copy import deepcopy
from fuzzyutils.membership import *
from fuzzyutils.plots import *

datadf = pd.read_csv('data.csv')

scaler = MinMaxScaler()
data = datadf[['Temperature','Humidity','Zone 1 Power Consumption']]
scaler.fit(data)
normData = scaler.transform(data)
X = normData[:,:2]
Y = normData[:,2:]

temperature = {'name': 'temperature'}
temperature['memfs'] = grid_gauss(3)
humidity = {'name': 'humidity'}
humidity['memfs'] = grid_gauss(3)
power = {'name': 'power'}
power['memfs'] = PolyMF(2, 9)

model = AnfisNet([temperature, humidity], power)

DTYPE = torch.float32
x = torch.tensor(X, dtype=DTYPE)
y = torch.tensor(Y, dtype=DTYPE)
n = int(len(x)*0.7)
all_dataset = torch.utils.data.TensorDataset(x,y)
train_dataset = torch.utils.data.TensorDataset(x[:n],y[:n])
test_dataset = torch.utils.data.TensorDataset(x[n:],y[n:])
dataloader = torch.utils.data.DataLoader(train_dataset, batch_size=2048,
 ↪  shuffle=False)

errors = []  # Keep a list of these for plotting afterwards
# optimizer = torch.optim.Adam(model.parameters(), lr=1e-4)
loss = torch.nn.MSELoss()
#optimizer = torch.optim.SGD(model.parameters(), lr=0.0005)
#optimizer = torch.optim.Rprop(model.parameters(), lr=0.001)
```

```
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)
epoch = 250
min_state = deepcopy(model.state_dict())
min_error = float('inf')
min_y_pred = None

for t in range(epoch):
    # Process each mini-batch in turn:
    with torch.autograd.set_detect_anomaly(True):
        for x, y_actual in dataloader:
            optimizer.zero_grad()
            y_pred = model(x)
            l = loss(y_pred, y_actual) # get loss
            l.backward()     # backwards propagation based on loss function
            optimizer.step() # Update parameters based on backward pass

    with torch.no_grad():
        x,y_actual = test_dataset[:]
        y_pred = model(x)
        mse = loss(y_pred, y_actual)
        if mse < min_error:
            min_y_pred = y_pred.clone()
            min_state = deepcopy(model.state_dict())
            min_error = l
        errors.append(mse)
    # Epoch ending, so now fit the coefficients based on all data:
    #x, y_actual = data.dataset.tensors
    #with torch.no_grad():
    #    model.fit_coeff(x, y_actual)
    # Get the error rate for the whole batch:
    # Print some progress information as the net is trained:
        if t/epoch*100 % 2 == 0:
            print(f"epoch: {t/epoch*100:5.2f} %, MSE Loss is: {mse:.8f}")

errorEpochs(errors)
fig = plt.gcf()
fig.savefig('train-epochs.jpg')
plt.show()
```

```
model.load_state_dict(min_state)
with torch.no_grad():
    x, y_actual = all_dataset[:]
    y_pred = model(x)

for var in model.in_variables_def:
    plot_fuzzy(var)
    fig = plt.gcf()
    fig.savefig(f'train-{var[”name”]}.jpg')
    plt.show()

plot_poly(model.out_variable_def)
fig = plt.gcf()
fig.savefig('train-occupancy.jpg')
plt.show()

with shelve.open('data') as db:
    db['train_datadf'] = datadf
    db['train_pred'] = y_pred.numpy()
    db['train_true'] = y_actual.numpy()
```

```
import shelve
import matplotlib.pyplot as plt
import seaborn as sns
from fuzzyutils.plots import *
from sklearn.metrics import mean_squared_error

sns.set_theme()

with shelve.open('data') as db:
    datadf = db['train_datadf']
    pred = db['train_pred']
    true = db['train_true']

plotPrediction(true,pred)
fig = plt.gcf()
fig.savefig('train-prediction.jpg')
```

```
plt.show()

errorDistribution(true-pred)
fig = plt.gcf()
fig.savefig('train-distribution.jpg')
plt.show()

with open(”train-mse.txt”, ”w”) as file:
    file.write(str(mean_squared_error(true,pred)))
```

A.2 Office Occupancy

A.2.1 Subtractive Clustering χωρίς ANFIS

Κώδικας για τη διερευνητική ανάλυση.

```
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from datetime import datetime

# Header
# ”date”,”Temperature”,”Humidity”,”Light”,”CO2”,”HumidityRatio”,”Occupancy”
df = pd.read_csv('data.csv')

sns.displot(df, x='Temperature', hue='Occupancy')
plt.title(”Temperature Distribution”)
plt.tight_layout()
plt.savefig('fig1.jpg')

sns.displot(df, x='Humidity', hue='Occupancy')
plt.title(”Humidity Distribution”)
plt.tight_layout()
plt.savefig('fig2.jpg')

sns.displot(df, x='Light', hue='Occupancy')
plt.title(”Light Distribution”)
plt.tight_layout()
plt.savefig('fig3.jpg')
```

```
sns.displot(df, x='CO2', hue='Occupancy')
plt.title(”CO2 Distribution”)
plt.tight_layout()
plt.savefig('fig4.jpg')

sns.displot(df, x='HumidityRatio', hue='Occupancy')
plt.title(”HumidityRatio Distribution”)
plt.tight_layout()
plt.savefig('fig5.jpg')

df['date'] = df['date'].apply(lambda t: datetime.strptime(t, '%Y-%m-%d
 ↪  %H:%M:%S'))
df['date'] = df['date'].apply(lambda t: t.hour*60+t.minute)
sns.displot(df, x='date', hue='Occupancy')
plt.title(”Hourly (clock time) Distribution”)
plt.tight_layout()
plt.savefig('fig6.jpg')
```

Η δημιουργία των clusters με την τεχνική Subtractive Clustering:

```
from fuzzyutils.clust import subclust_wrapper
import pandas as pd

datadf = pd.read_csv('data.csv')
subclust_wrapper(datadf, ['Temperature','Light'], ['Occupancy'],
 ↪  savefig=”subclust-”, radiusA=0.4)
```

```
import shelve
from fuzzyutils.membership import *
import torch

with shelve.open('data') as db:
    C = db['Cnorm']
    S = db['Snorm']
    X = db['Xnorm']
    datadf = db['datadf']
    col = db['col']
```

```
temperature = {'name': 'temperature'}
light = {'name': 'light'}
occupancy = {'name': 'occupancy'}

temperature['memfs'] = clusters_gauss(C[:,0], S[0,0])
light['memfs'] = clusters_gauss(C[:, 1], S[0,0])
occupancy['memfs'] = PolyMF(0, len(C))
occupancy['memfs'].set_bias(C[:,-1])

plot_fuzzy(temperature)
fig = plt.gcf()
fig.savefig('notrain-temperature.jpg')
plt.show()
plot_fuzzy(light)
fig = plt.gcf()
fig.savefig('notrain-light.jpg')
plt.show()
plot_poly(occupancy)
fig = plt.gcf()
fig.savefig('notrain-occupancy.jpg')
plt.show()

model = AnfisNet([temperature, light], occupancy, rules='clustering')

DTYPE = torch.float32
x = torch.tensor(X[:,:-1], dtype=DTYPE)
y = torch.tensor(X[:,-1:], dtype=DTYPE)

with torch.no_grad():
    y_pred = model(x)

with shelve.open('data') as db:
    db['notrain-datadf'] = datadf
    db['notrain-true'] = y.numpy()
    db['notrain-pred'] = y_pred.numpy()
```

```
import shelve
```

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import mean_squared_error
from sklearn.metrics import f1_score, precision_score, recall_score
from fuzzyutils.plots import *

sns.set_theme()

with shelve.open('data') as db:
    datadf = db['notrain-datadf']
    pred = db['notrain-pred']
    true = db['notrain-true']

plotPredictionPoints(true,pred)
fig = plt.gcf()
fig.savefig('notrain-prediction1.jpg')
plt.show()

pred[pred < 0.5] = 0
pred[pred >= 0.5] = 1

plotPredictionPoints(true,pred)
fig = plt.gcf()
fig.savefig('notrain-prediction2.jpg')
plt.show()

with open("notrain-mse.txt", "w") as file:
    file.write(f"mse, {str(mean_squared_error(true,pred))}\n")
    file.write(f"f1_score, {str(f1_score(true,pred))}\n")
    file.write(f"precision_score, {str(precision_score(true,pred))}\n")
    file.write(f"recall_score, {str(recall_score(true,pred))}\n")
```

A.2.2 Subtractive Clustering με ANFIS

```
from sklearn.preprocessing import MinMaxScaler
import pandas as pd
import shelve
import torch
from copy import deepcopy
```

```
from fuzzyutils.membership import *
from fuzzyutils.plots import *

with shelve.open('data') as db:
    C = db['Cnorm']
    S = db['Snorm']
    X = db['Xnorm']
    datadf = db['datadf']
    col = db['col']

temperature = {'name': 'temperature'}
light = {'name': 'light'}
occupancy = {'name': 'occupancy'}

temperature['memfs'] = clusters_gauss(C[:,0], S[0,0])
light['memfs'] = clusters_gauss(C[:, 1], S[0,0])
occupancy['memfs'] = PolyMF(2, len(C))
occupancy['memfs'].set_bias(C[:,-1])

model = AnfisNet([temperature, light], occupancy, rules='clustering')

DTYPE = torch.float32
x = torch.tensor(X[:,:-1], dtype=DTYPE)
y = torch.tensor(X[:,-1:], dtype=DTYPE)
n = int(len(x)*0.7)
all_dataset = torch.utils.data.TensorDataset(x,y)
train_dataset = torch.utils.data.TensorDataset(x[:n],y[:n])
test_dataset = torch.utils.data.TensorDataset(x[n:],y[n:])
dataloader = torch.utils.data.DataLoader(train_dataset, batch_size=2048,
 ↪  shuffle=False)

errors = []  # Keep a list of these for plotting afterwards
# optimizer = torch.optim.Adam(model.parameters(), lr=1e-4)
loss = torch.nn.MSELoss()
#optimizer = torch.optim.SGD(model.parameters(), lr=0.0005)
#optimizer = torch.optim.Rprop(model.parameters(), lr=0.0005)
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)
epoch = 250
```

```
min_state = deepcopy(model.state_dict())
min_error = float('inf')
min_y_pred = None

for t in range(epoch):
    # Process each mini-batch in turn:
    with torch.autograd.set_detect_anomaly(True):
        for x, y_actual in dataloader:
            optimizer.zero_grad()
            y_pred = model(x)
            #model.fit_coeffs(x, y_pred)
            l = loss(y_pred, y_actual) # get loss
            l.backward()     # backwards propagation based on loss function
            optimizer.step() # Update parameters based on backward pass

    with torch.no_grad():
        x,y_actual = test_dataset[:]
        y_pred = model(x)
        mse = loss(y_pred, y_actual)
        if mse < min_error:
            min_y_pred = y_pred.clone()
            min_state = deepcopy(model.state_dict())
            min_error = l
        errors.append(mse)
    # Epoch ending, so now fit the coefficients based on all data:
    #x, y_actual = data.dataset.tensors
    #with torch.no_grad():
    #    model.fit_coeff(x, y_actual)
    # Get the error rate for the whole batch:
    # Print some progress information as the net is trained:
        if t/epoch*100 % 2 == 0:
            print(f"epoch: {t/epoch*100:5.2f} %, MSE Loss is: {mse:.8f}")

errorEpochs(errors)
fig = plt.gcf()
fig.savefig('train-epochs.jpg')
plt.show()
```

```
model.load_state_dict(min_state)
with torch.no_grad():
    x, y_actual = all_dataset[:]
    y_pred = model(x)

for var in model.in_variables_def:
    plot_fuzzy(var)
    fig = plt.gcf()
    fig.savefig(f'train-{var[”name”]}.jpg')
    plt.show()

plot_poly(model.out_variable_def)
fig = plt.gcf()
fig.savefig('train-occupancy.jpg')
plt.show()



with shelve.open('data') as db:
    db['train-datadf'] = datadf
    db['train-pred'] = y_pred.numpy()
    db['train-true'] = y_actual.numpy()
```

```
import shelve
import matplotlib.pyplot as plt
import seaborn as sns
from fuzzyutils.plots import *
from sklearn.metrics import mean_squared_error
from sklearn.metrics import f1_score, precision_score, recall_score

sns.set_theme()

with shelve.open('data') as db:
    datadf = db['train-datadf']
    pred = db['train-pred']
    true = db['train-true']

plotPredictionPoints(true,pred)
fig = plt.gcf()
```

```
fig.savefig('train-prediction1.jpg')
plt.show()

pred[pred < 0.5] = 0
pred[pred >= 0.5] = 1

plotPredictionPoints(true,pred)
fig = plt.gcf()
fig.savefig('train-prediction2.jpg')
plt.show()

with open("train-mse.txt", "w") as file:
    file.write(f"mse, {str(mean_squared_error(true,pred))}\n")
    file.write(f"f1_score, {str(f1_score(true,pred))}\n")
    file.write(f"precision_score, {str(precision_score(true,pred))}\n")
    file.write(f"recall_score, {str(recall_score(true,pred))}\n")
```

A.2.3 Grid Partitioning χωρίς ANFIS

```
# This file is licensed under GPLv3
import pandas as pd
import numpy as np
from simpful import FuzzySystem
import multiprocessing as mp
import shelve
from fuzzyutils.membership import *
from fuzzyutils.plots import *
from fuzzyutils.simpful import *

datadf = pd.read_csv("data.csv")
temperature_df = datadf["Temperature"]
light_df = datadf["Light"]
occupancy_df = datadf["Occupancy"]

temperature = {'name': 'temperature', 'domain': (temperature_df.min()*.9,
 ↪  temperature_df.max()*1.1)}
light = {'name': 'light', 'domain': (light_df.min()*.9, light_df.max()*1.1)}
occupancy = {'name': 'occupancy', 'domain': (occupancy_df.min()*.9,
 ↪  occupancy_df.max()*1.1)}
```

```
temperature['memfs'] = grid_gauss(3, domain=temperature['domain'])
light['memfs'] = grid_gauss(3, domain=light['domain'])
occupancy['memfs'] = grid_gauss(3, domain=occupancy['domain'])

plot_fuzzy(temperature)
fig = plt.gcf()
fig.savefig('notrain-temperature.jpg')
plt.show()
plot_fuzzy(light)
fig = plt.gcf()
fig.savefig('notrain-light.jpg')
plt.show()
plot_fuzzy(occupancy)
fig = plt.gcf()
fig.savefig('notrain-occupancy.jpg')
plt.show()

fis = FuzzySystem()
lv = variable_to_simpful(temperature)
fis.add_linguistic_variable(name=lv._concept, LV=lv)
lv = variable_to_simpful(light)
fis.add_linguistic_variable(name=lv._concept, LV=lv)
lv = variable_to_simpful(occupancy)
fis.add_linguistic_variable(name=lv._concept, LV=lv)


rules_array = gen_grid_rules([temperature, light, occupancy],
 ↪  datadf[['Temperature','Light','Occupancy']].values)
rules_df = pd.DataFrame(
    rules_array, columns=[”Temperature”, ”Light”, ”Occupancy”]
)

rules_fuzzy = []
for idx, rule in rules_df.iterrows():
    rules_fuzzy.append(
        f”IF (temperature IS mf{rule['Temperature']}) AND (light IS
         ↪  mf{rule['Light']}) THEN (occupancy IS mf{rule['Occupancy']})”
```

```
    )

fis.add_rules(rules_fuzzy)

pool_data = datadf[[”Temperature”, ”Light”]].values


def process_func(row: np.ndarray):
    fis.set_variable(”temperature”, row[0])
    fis.set_variable(”light”, row[1])
    return fis.inference()[”occupancy”]


pool_cpus = mp.cpu_count() - 1
with mp.Pool(processes=pool_cpus) as pool:
    result = pool.map(process_func, pool_data)

with shelve.open(”data”) as db:
    db[”notrain-pred”] = np.array(result)
    db[”notrain-true”] = datadf[['Occupancy']].values.squeeze()
    db[”notrain-fis”] = fis
    db[”notrain-datadf”] = datadf

with open(”notrain-rules.txt”, ”w”) as file:
    file.write(str(fis.get_rules()))
```

```
import shelve
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import f1_score, precision_score, recall_score
from fuzzyutils.plots import *

sns.set_theme()

with shelve.open('data') as db:
    datadf = db['notrain-datadf']
```

```
    pred = db['notrain-pred']
    true = db['notrain-true']

# Scale output from Simpful
scaler = MinMaxScaler()
scaler.fit(true.reshape(-1,1))
true = scaler.transform(true.reshape(-1,1))
pred = scaler.transform(pred.reshape(-1,1))

plotPredictionPoints(true,pred)
fig = plt.gcf()
fig.savefig('notrain-prediction1.jpg')
plt.show()

pred[pred < 0.5] = 0
pred[pred >= 0.5] = 1

plotPredictionPoints(true,pred)
fig = plt.gcf()
fig.savefig('notrain-prediction2.jpg')
plt.show()

with open("notrain-mse.txt", "w") as file:
    file.write(f"mse, {str(mean_squared_error(true,pred))}\n")
    file.write(f"f1_score, {str(f1_score(true,pred))}\n")
    file.write(f"precision_score, {str(precision_score(true,pred))}\n")
    file.write(f"recall_score, {str(recall_score(true,pred))}\n")
```

A.2.4 Grid Partitioning με ANFIS

```
from sklearn.preprocessing import MinMaxScaler
import pandas as pd
import shelve
import torch
from copy import deepcopy
from fuzzyutils.membership import *
from fuzzyutils.plots import *

datadf = pd.read_csv('data.csv')
```

```
scaler = MinMaxScaler()
data = datadf[['Temperature','Light','Occupancy']]
scaler.fit(data)
normData = scaler.transform(data)
X = normData[:,:2]
Y = normData[:,2:]

temperature = {'name': 'temperature'}
temperature['memfs'] = grid_gauss(3)
light = {'name': 'light'}
light['memfs'] = grid_gauss(3)
occupancy = {'name': 'occupancy'}
occupancy['memfs'] = PolyMF(2, 9)

plot_fuzzy(temperature)
plt.show()
plot_fuzzy(light)
plt.show()
plot_poly(occupancy)
fig = plt.gcf()
fig.savefig('train-occupancy-init.jpg')
plt.show()

model = AnfisNet([temperature, light], occupancy)

DTYPE = torch.float32
x = torch.tensor(X, dtype=DTYPE)
y = torch.tensor(Y, dtype=DTYPE)
n = int(len(x)*0.7)
all_dataset = torch.utils.data.TensorDataset(x,y)
train_dataset = torch.utils.data.TensorDataset(x[:n],y[:n])
test_dataset = torch.utils.data.TensorDataset(x[n:],y[n:])
dataloader = torch.utils.data.DataLoader(train_dataset, batch_size=2048,
 ↪  shuffle=False)

errors = []  # Keep a list of these for plotting afterwards
# optimizer = torch.optim.Adam(model.parameters(), lr=1e-4)
```

```
loss = torch.nn.MSELoss()
#optimizer = torch.optim.SGD(model.parameters(), lr=0.0005)
#optimizer = torch.optim.Rprop(model.parameters(), lr=0.001)
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)
epoch = 250
min_state = deepcopy(model.state_dict())
min_error = float('inf')
min_y_pred = None

for t in range(epoch):
    # Process each mini-batch in turn:
    with torch.autograd.set_detect_anomaly(True):
        for x, y_actual in dataloader:
            optimizer.zero_grad()
            y_pred = model(x)
            #model.fit_coeffs(x, y_pred)
            l = loss(y_pred, y_actual) # get loss
            l.backward()     # backwards propagation based on loss function
            optimizer.step() # Update parameters based on backward pass

    with torch.no_grad():
        x,y_actual = test_dataset[:]
        y_pred = model(x)
        mse = loss(y_pred, y_actual)
        if mse < min_error:
            min_y_pred = y_pred.clone()
            min_state = deepcopy(model.state_dict())
            min_error = l
        errors.append(mse)
    # Epoch ending, so now fit the coefficients based on all data:
    #x, y_actual = data.dataset.tensors
    #with torch.no_grad():
    #    model.fit_coeff(x, y_actual)
    # Get the error rate for the whole batch:
    # Print some progress information as the net is trained:
        if t/epoch*100 % 2 == 0:
            print(f"epoch: {t/epoch*100:5.2f} %, MSE Loss is: {mse:.8f}")
```

```
errorEpochs(errors)
fig = plt.gcf()
fig.savefig('train-epochs.jpg')
plt.show()

model.load_state_dict(min_state)
with torch.no_grad():
    x, y_actual = all_dataset[:]
    y_pred = model(x)

for var in model.in_variables_def:
    plot_fuzzy(var)
    fig = plt.gcf()
    fig.savefig(f'train-{var[”name”]}.jpg')
    plt.show()

plot_poly(model.out_variable_def)
fig = plt.gcf()
fig.savefig('train-occupancy.jpg')
plt.show()

with shelve.open('data') as db:
    db['train_datadf'] = datadf
    db['train_pred'] = y_pred.numpy()
    db['train_true'] = y_actual.numpy()
```

```
import shelve
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from fuzzyutils.plots import *
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import f1_score, precision_score, recall_score

sns.set_theme()

with shelve.open('data') as db:
```

```
    datadf = db['train_datadf']
    pred = db['train_pred']
    true = db['train_true']

plotPredictionPoints(true,pred)
fig = plt.gcf()
fig.savefig('train-prediction1.jpg')
plt.show()

pred[pred < 0.5] = 0
pred[pred >= 0.5] = 1

plotPredictionPoints(true,pred)
fig = plt.gcf()
fig.savefig('train-prediction2.jpg')
plt.show()

with open(”train-mse.txt”, ”w”) as file:
    file.write(f”mse, {str(mean_squared_error(true,pred))}\n”)
    file.write(f”f1_score, {str(f1_score(true,pred))}\n”)
    file.write(f”precision_score, {str(precision_score(true,pred))}\n”)
    file.write(f”recall_score, {str(recall_score(true,pred))}\n”)
```

Παράρτημα Β

Κλάσεις και Συναρτήσεις

Τα παρακάτω αρχεία πρέπει να βρίσκονται σε έναν φάκελο που λέγεται «fuzzyutils» και σε αυτό το φάκελο θα πρέπει να υπάρχει επίσης ένα αρχείο «__init__.py» με τα ακόλουθα περιεχόμενα:

```
import os, sys; sys.path.append(os.path.dirname(os.path.realpath(__file__)))
```

Β.1 Συναρτήσεις Συμμετοχής, ANFIS και Subtractive Clustering

Το αρχείο «membership.py» περιέχει τον κώδικα για τις λεκτικές μεταβλητές, συναρτήσεις συμμετοχής και το ANFIS.

```
import itertools
import numpy as np
import torch
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

sns.set_theme()


DTYPE = torch.float32



class GaussMF(torch.nn.Module):
    def __init__(self, mu, sigma, dtype=torch.float32):
```

```
        super(GaussMF, self).__init__()
        self.mu = torch.nn.Parameter(torch.tensor(mu, dtype=dtype))
        self.sigma = torch.nn.Parameter(torch.tensor(sigma, dtype=dtype))

    def forward(self, x: torch.Tensor) -> torch.Tensor:
        """
        x.shape: (batch_size, 1) or (batch_size)
        y.shape: (batch_size, 1) or (batch_size)
        """
        if not isinstance(x, torch.Tensor):
            x = torch.tensor(x, dtype=torch.float32)
        val = torch.exp(-(((x - self.mu) / self.sigma) ** 2))
        return val


class PolyMF(torch.nn.Module):
    def __init__(self, n_input, n_rules, domain=(-1, 2)):
        super(PolyMF, self).__init__()
        range_ = abs(domain[0] - domain[1])
        mf = torch.rand([n_rules, n_input + 1], dtype=DTYPE) * range_ -
        ↪  abs(domain[0])
        self.mf = torch.nn.Parameter(mf)

    def forward(self, x: torch.Tensor) -> torch.Tensor:
        """
        x.shape: (batch_size, 1) or (batch_size)
        y.shape: (batch_size, 1) or (batch_size)
        """
        if self.mf.shape[-1] == 1:
            return self.mf.repeat((x.shape[0], 1, 1)).squeeze()
        x_padded = torch.nn.functional.pad(x, (0, 1), value=1)
        return x_padded @ self.mf.T

    def set_bias(self, bias):
        if not isinstance(bias, torch.Tensor):
            bias = torch.tensor(bias, dtype=DTYPE)
        with torch.no_grad():
            self.mf[:, -1] = bias
```

```
def grid_gauss(n, domain=(0, 1)):
    mean = (domain[1] - domain[0]) / (n - 1)
    sigma = mean / 3
    res = torch.nn.ModuleList()
    for i in range(n):
        res.append(GaussMF(domain[0] + mean * i, sigma))
    return res


def clusters_gauss(centers, sigma):
    res = torch.nn.ModuleList()
    for i, c in enumerate(centers):
        res.append(GaussMF(c, sigma))
    return res


def poly(n_input, n_rules, domain=(-1, 2)):
    range_ = abs(domain[0] - domain[1])
    return torch.rand([n_rules, n_input + 1], dtype=torch.float32) * range_
    ↪  - abs(
        domain[0]
    )


def plot_fuzzy(variable_def):
    if ”domain” not in variable_def.keys():
        variable_def[”domain”] = (0, 1)
    x = torch.linspace(
        variable_def[”domain”][0], variable_def[”domain”][1], 10001,
        ↪  dtype=DTYPE
    )
    with torch.no_grad():
        for mf in variable_def[”memfs”]:
            plt.plot(x, mf(x))
    plt.title(f”State of Variable {variable_def['name']}”)
    plt.xlabel(”Universe of Discourse”)
```

```
    plt.ylabel("Membership Grade")
    legend = [f"mf{i}" for i in range(len(variable_def["memfs"]))]
    plt.legend(legend)


def plot_poly(variable_def):
    mf = variable_def["memfs"].mf
    n_input = mf.shape[1] - 1
    n_rules = mf.shape[0]
    fig, ax = plt.subplots()
    ax.axis("off")
    ax.axis("tight")

    columns = [f"a{j}" for j in range(n_input)]
    columns.append("b")
    rows = [f"rule{j}" for j in range(n_rules)]

    with torch.no_grad():
        df = pd.DataFrame(mf, columns=columns, index=rows)

    table = ax.table(
        cellText=df.values,
        colLabels=df.columns,
        rowLabels=rows,
        loc="center",
    )
    ax.set_title(f"State of Variable {variable_def['name']}")

    fig.tight_layout()

def plot_poly2(variable_def):
    mf = variable_def["memfs"].mf
    n_input = mf.shape[1] - 1
    n_rules = mf.shape[0]
    fig, ax = plt.subplots(figsize=(1,1))
    ax.axis("off")
    ax.axis("tight")
```

```
    step = 0.8/n_rules
    height=0.8
    for i in range(n_rules):
        text = f"$y_{i} = {mf[i,-1]} "
        for j in range(n_input):
            text += f"+ {mf[i,j]}x_{j}"
        text += "$"
        fig.text(.1,height,text)
        height-=step
    fig.tight_layout()
    ax.set_title(f"State of Variable {variable_def['name']}")


def gen_grid_rules(variables, data, n_output=1):
    n_input = len(variables) - n_output

    max_values = []
    max_idxs = []
    for i, variable in enumerate(variables):
        mfs_values = [mf(data[:, i]) for mf in variable["memfs"]]
        mfs_values = torch.vstack(mfs_values)
        max_v, max_i = mfs_values.max(axis=0)
        max_values.append(max_v)
        max_idxs.append(max_i)

    max_values = torch.vstack(max_values).T
    max_idxs = torch.vstack(max_idxs).T

    uniq = torch.unique(max_idxs, dim=0, return_counts=True)
    uniq_idxs = uniq[0]
    uniq_count = uniq[1]

    uniq_premise = torch.unique(uniq_idxs[:, :n_input], dim=0)
    rules = []
    for premise in uniq_premise:
        idx = torch.where(torch.all(uniq_idxs[:, :n_input] == premise,
        ↪  dim=1))
        chosen_idx = uniq_count[idx].argmax()
```

```
        rules.append(uniq_idxs[idx[0][chosen_idx]])
    return np.vstack(rules)


def fz_forward(self, x):
    """
    x.shape: (batch_size, 1)
    y.shape: (batch_size, n_mfs)
    """
    mfs = [mf(x).unsqueeze(1) for mf in self.mfs.values()]
    return torch.hstack(mfs)


class AnfisNet(torch.nn.Module):
    def __init__(
        self,
        in_variables,
        out_variable,
        rules="grid",
    ):
        super(AnfisNet, self).__init__()
        self.in_variables_def = in_variables
        self.in_variables = torch.nn.ModuleList()
        for i, variable in enumerate(self.in_variables_def):
            self.in_variables.append(variable["memfs"])
        self.out_variable_def = out_variable
        self.out_variable = out_variable["memfs"]
        self.n_input = len(self.in_variables_def)

        # Create indexes
        mfs_size = [len(mfs) for mfs in self.in_variables]
        self.mfs_size = mfs_size
        self.max_size = max(mfs_size)

        if rules == "grid":
            idxs = list(itertools.product(*[range(n) for n in mfs_size]))
        elif rules == "clustering":
            idxs = [(i, i) for i in range(max(mfs_size))]
```

```
        else:
            idxs = []
        idxs = torch.tensor(idxs)
        self.idxs_ = idxs.clone()
        self.n_rules = idxs.shape[0]
        for i in range(1, idxs.shape[-1]):
            n_max = idxs[:, i - 1].max() + 1
            idxs[:, i] += n_max
        self.idxs = idxs.flatten()

        self.original_state = self.state_dict()

    def forward(self, x):
        """
        x.shape: (batch_size, n_input)
        y.shape: (batch_size, n_output)
        Note: Variables are saved for debugging
        """
        # Fuzzify
        x_fuz = []
        for i, variable in enumerate(self.in_variables):
            res_var = torch.hstack([mf(x[:, i]).unsqueeze(1) for mf in
             ↪  variable])
            if res_var.shape[1] < self.max_size:
                n = self.max_size - res_var.shape[0]
                res_var = torch.nn.functional.pad(res_var, (0, n), value=1)
            x_fuz.append(res_var)
        x_fuz = torch.hstack(x_fuz)

        # Get Product
        x_prod = x_fuz[:, self.idxs]
        x_prod = x_prod.view(x.shape[0], self.n_rules, x.shape[1])
        x_prod = x_prod.prod(2)

        # Normalize Weights
        x_weights = torch.nn.functional.normalize(x_prod, p=1, dim=1)

        # Get Output
```

```
        tsk = self.out_variable(x)
        return (x_weights * tsk).sum(1).unsqueeze(1)

    def predict(self, x):
        with torch.no_grad():
            return self(x)

    def plot(self):
        for var in self.in_variables_def:
            plot_fuzzy(var)
        plot_poly(self.out_variable_def)

    def input_variables(self):
        return self.in_variables
```

Το αρχείο «simpful.py» περιέχει κώδικα για την αλληλεπίδραση με το πακέτο simpful. Ο κώδικας σε αυτό το αρχείο προστατεύεται από την άδεια χρήσης GPLv3.

```
# This file is licensed under GPLv3

def variable_to_simpful(variable_def):
    from simpful import LinguisticVariable, GaussianFuzzySet

    fuzzy_sets = []
    for i, mf in enumerate(variable_def['memfs']):
        fuzzy_sets.append(GaussianFuzzySet(mf.mu.item(), mf.sigma.item(),
        ↪ f"mf{i}"))
    return LinguisticVariable(
        fuzzy_sets, concept=variable_def['name'],
        ↪ universe_of_discourse=variable_def['domain']
    )
```

Το αρχείο «clust.py» περιέχει τον κώδικα για τον αλγόριθμο Subtractive Clustering.

```
from math import sqrt
from scipy.spatial.distance import cdist
from .plots import plotClusters
import logging
```

```
import numpy as np
import pandas as pd


def subclust_wrapper(
    datadf: pd.DataFrame, input_columns, output_columns, savefig=None,
    ↪ **subclust_args
):
    """
    Convenience function that automatically shows plots and saves data.
    For more fine grained control, use subclust function.
    """
    from sklearn.preprocessing import MinMaxScaler
    import shelve

    col = input_columns + output_columns

    X = datadf[col]
    scaler = MinMaxScaler()
    scaler.fit(X.values)
    Xnorm = scaler.transform(X.values)

    Cnorm, Snorm = subclust(Xnorm, **subclust_args)

    C = scaler.inverse_transform(Cnorm)
    S = scaler.inverse_transform(Snorm)
    #print(f"{len(C)=}")
    #print(f"{C=}")
    #print(f"{S=}")
    #print(f"{Snorm=}")

    plotClusters(datadf, C, S, input_columns, output_columns,
    ↪ savefig=savefig)

    with shelve.open("data") as db:
        db["Cnorm"] = Cnorm
        db["Snorm"] = Snorm
        db["C"] = C
```

```
        db["S"] = S
        db["Xnorm"] = Xnorm
        db["scaler"] = scaler
        db["col"] = col
        db["datadf"] = datadf

    return C, S, Cnorm, Snorm, scaler


def subclust(
    data: np.ndarray,
    radiusA=0.5,
    radiusB=None,
    pmin=None,
    k=2 * sqrt(2),
    eCeil=0.5,
    eFloor=0.15,
    log=None,
):
    points = np.array(data, ndmin=2)
    if pmin:
        k = 1 / sqrt(1 - pmin)
    radiusB = radiusB or 1.25 * radiusA
    A = -1 / (2 * (radiusA / k) ** 2)
    B = -1 / (2 * (radiusB / k) ** 2)
    centers = []
    centersPotential = []
    logging.basicConfig(level=log)

    # Step 1; Normalize data to a hypercube
    # points = normalize(data, norm='max', axis=0)

    # Step 2; first calculation of  points' Potential (as a cluster center)
    # pdist uses very much memory
    # splitting the array does not help either
    logging.info("Step 2 Init")
    potential = [np.exp(A * cdist([p], points) ** 2).sum() for p in points]
    logging.info("Step 2 Completed")
```

```
    logging.info(f"{potential[:100]=}")

    # Step 3; select the highest potential as center
    logging.info("Step 3 Init")
    maxi = np.argmax(potential)
    maxPotential = potential[maxi]
    centers.append(points[maxi])
    centersPotential.append(maxPotential)
    logging.info(f"{centers=}")
    logging.info(f"{centersPotential=}")
    logging.info("Step 3 Completed")

    acceptanceThreshold = eCeil * centersPotential[0]
    rejectionThreshold = eFloor * centersPotential[0]
    logging.info(f"{acceptanceThreshold=}")
    logging.info(f"{rejectionThreshold=}")

    logging.info("Step 4 & 5 Init")
    while True:
        potential = potential - centersPotential[-1] * np.exp(
            B * cdist([centers[-1]], points)[0] ** 2
        )
        maxi = np.argmax(potential)
        maxPotential = potential[maxi]
        logging.info(f"{maxi=}")
        logging.info(f"{maxPotential=}")

        if maxPotential > acceptanceThreshold:
            centers.append(points[maxi])
            centersPotential.append(maxPotential)
        elif maxPotential < rejectionThreshold:
            break
        else:
            d_min = np.min(cdist([points[maxi]], centers))
            if d_min / radiusA + maxPotential / centersPotential[0] >= 1:
                centers.append(points[maxi])
                centersPotential.append(maxPotential)
            else:
```

```
                potential[maxi] = 0
                maxi = np.argmax(potential)
                maxPotential = potential[maxi]
                centers.append(points[maxi])
                centersPotential.append(maxPotential)

        logging.info(f"{centers=}")
        logging.info(f"{centersPotential=}")

    logging.info("Step 4 & 5 Completed")

    centers = np.array(centers)
    return centers, np.tile(radiusA / k, centers.shape)


if __name__ == "__main__":
    # Testing
    d = np.array(
        [
            [0.0, 0.0, 0.0],
            [0.1, 0.1, 0.1],
            [0.1, 0.1, 0.1],
            [0.1, 0.1, 0.1],
            [0.1, 0.1, 0.1],
            [0.5, 0.5, 0.5],
            [0.9, 0.9, 0.9],
            [0.9, 0.9, 0.9],
            [0.9, 0.9, 0.9],
            [0.9, 0.9, 0.9],
            [0.9, 0.9, 0.9],
            [0.9, 0.9, 0.9],
            [0.9, 0.9, 0.9],
            [0.9, 0.9, 0.9],
            [0.9, 0.9, 0.9],
            [0.9, 0.9, 0.9],
            [0.9, 0.9, 0.9],
            [0.9, 0.9, 0.9],
        ]
```

```
    )

    C, S = subclust(d)
    print(C)
    print(S)
```

B.2 Plotting

Το αρχείο «plots.py» περιέχει συναρτήσεις για τη δημιουργία διαγραμμάτων.

```
import matplotlib.pyplot as plt
from matplotlib.patches import Ellipse
import seaborn as sns

sns.set_theme()


def plotPrediction(y_true, y_pred):
    fig, ax = plt.subplots(figsize=(6,3))
    n = range(len(y_pred))
    plt.plot(n, y_true)
    plt.plot(n, y_pred, "r")
    plt.xlabel("Iterations")
    plt.ylabel("Value")
    plt.legend(["truth", "prediction"])
    plt.title("Truth vs Prediction")


def plotPredictionPoints(y_true, y_pred):
    fig, ax = plt.subplots(figsize=(6,3))
    n = range(len(y_pred))
    plt.plot(n, y_true, "x")
    plt.plot(n, y_pred, "rx")
    plt.xlabel("Iterations")
    plt.ylabel("Value")
    plt.legend(["truth", "prediction"])
    plt.title("Truth vs Prediction")
```

```
def scatterPrediction(y_true, y_pred):
    fig, ax = plt.subplots(figsize=(6,3))
    plt.scatter(y_pred, y_true)
    plt.xlabel("Predicted Values")
    plt.ylabel("True Values")
    plt.title("Truth vs Prediction")
    p1 = max(max(y_pred), max(y_true))
    p2 = min(min(y_pred), min(y_true))
    plt.plot([p1, p2], [p1, p2], "r-")


def errorDistribution(errors, values=None):
    fig, ax = plt.subplots(figsize=(6,3))
    sns.histplot(errors, stat="probability")  # density=True, rwidth=500)
    plt.ylabel('Percentage')
    plt.title("Error distribution")


def errorCummulative(errors, values=None):
    fig, ax = plt.subplots(figsize=(6,3))
    sns.ecdfplot(errors)
    plt.title("Error distribution")


def errorEpochs(errors):
    fig, ax = plt.subplots(figsize=(6,3))
    plt.title("Error per epoch")
    plt.xlabel("Epoch")
    plt.ylabel("Error")
    plt.plot(range(len(errors)), errors)


def plotClusters(datadf, C, S, input_columns, output_columns, savefig=None):
    n_input = len(input_columns)
    # Visualise clusters
    for i, input_column in enumerate(input_columns):
        for j, output_column in enumerate(output_columns):
```

```
            fig, ax = plt.subplots()
            plt.scatter(datadf[input_column], datadf[output_column])
            plt.grid()
            ax.set_xlabel(input_column)
            ax.set_ylabel(output_column)
            ax.set_title("Clusters from substractive clustering")
            j = n_input + j
            plt.plot(C[:, i], C[:, j], "rx")
            for k in range(len(C)):
                ax.add_patch(
                    Ellipse(
                        (C[k, i], C[k, j]),
                        S[k, i],
                        S[k, j],
                        edgecolor=(1, 0, 0),
                        fc="None",
                        lw=2,
                    )
                )
                ax.add_patch(
                    Ellipse(
                        (C[k, i], C[k, j]),
                        2 * S[k, i],
                        2 * S[k, j],
                        edgecolor=(1, 0.5, 0.5),
                        fc="None",
                        lw=2,
                    )
                )
            if savefig is not None:
                fig.savefig(savefig + input_column + ".jpg",
                 ↪  bbox_inches="tight")
            plt.show()
```

Βιβλιογραφία & Δικτυογραφία

[1] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. Springer, 1981, σ. 256.

[2] L. Candanedo, *Occupancy Detection*, UCI Machine Learning Repository, 2016. DOI: 10.24432/C5X01N.

[3] Chiu και S. L., «Fuzzy Model Identification Based on Cluster Estimation,» *Journal of Intelligent and Fuzzy Systems*, τόμ. 2, αρθμ. 3, σσ. 267–278, 1994. DOI: 10.3233/IFS-1994-2306.

[4] R. Collobert, S. Bengio και J. Mariethoz, *Pytorch*, Github, έκδ. 2.1. διεύθν.: https://pytorch.org/.

[5] D. Cournapeau, *Scikit-Learn*, Github, έκδ. 1.3.2. διεύθν.: https://scikit-learn.org.

[6] S. Guillaume, «Designing fuzzy inference systems from data: An interpretability-oriented review,» *IEEE Transactions on Fuzzy Systems*, τόμ. 9, αρθμ. 3, σσ. 426–443, 2001. DOI: 10.1109/91.928739.

[7] J. D. Hunter, *Matplotlib*, Github, έκδ. 3.8. διεύθν.: https://matplotlib.org/.

[8] Jang, J.-S.R., «ANFIS: adaptive-network-based fuzzy inference system,» *IEEE Transactions on Systems, Man, and Cybernetics*, τόμ. 23, αρθμ. 3, σσ. 665–685, 1993. DOI: 10.1109/21.256541.

[9] J. JWarner, *Scikit-Fuzzy*, Github, έκδ. d7551b6. διεύθν.: https://github.com/scikit-fuzzy/scikit-fuzzy.

[10] G. Lenhard και D. Maringer, *sanfis: State-Adaptive Neuro-Fuzzy Inference System*, Github, έκδ. d6850bd. διεύθν.: https://github.com/gregorLen/S-ANFIS-PyTorch.

[11] E. Mamdani και S. Assilian, «An experiment in linguistic synthesis with a fuzzy logic controller,» *International Journal of Man-Machine Studies*, τόμ. 7, αρθμ. 1, σσ. 1–13, 1975. DOI: 10.1016/S0020-7373(75)80002-2.

[12] W. McKinney, *Pandas*, Github, έκδ. 2.1. διεύθν.: https://pandas.pydata.org/.

[13] T. Oliphant, *Numpy*, Github, έκδ. 1.26. διεύθν.: https://numpy.org/.

[14] J. Power, *anfis-pytorch*, Github, έκδ. 060acad. διεύθν.: https://github.com/jfpower/anfis-pytorch.

[15] G. van Rossum, *Python Programming Language*, Github, έκδ. 3.11.5. διεύθν.: https://www.python.org/.

[16] A. Salam και A. El Hibaoui, *Power consumption of Tetouan city*, UCI Machine Learning Repository, 2023. DOI: 10.24432/C5B034.

[17] Spolaor,S, Fuchs C., Cazzaniga P., Kaymak U., Besozzi D. και Nobile M.S., *Simpful*, Github, έκδ. 785d7c5. διεύθν.: https://github.com/aresio/simpful.

[18] M. Sugeno, *Industrial Applications of Fuzzy Control Profile image of Michio Sugeno*. Elsevier Science Inc., 1987, σ. 278.

[19] J. D. Velásquez-Henao, *fuzzy-expert*, Github, έκδ. b27f55a. διεύθν.: https://github.com/jdvelasq/fuzzy-expert.

[20] M. L. Waskom, *Seaborn*, Github, έκδ. 0.13.0. διεύθν.: https://seaborn.pydata.org/.

[21] L. Zadeh, «Fuzzy sets,» *Information and Control*, τόμ. 8, αρθμ. 3, σσ. 338–353, 1965. DOI: 10.1016/S0019-9958(65)90241-X.

[22] Π. Μαστοροκώστας, *Ασαφή και Νευροασαφή συστήματα - Ασαφής Βαθειά Μάθηση*. Κάλλιπος, 2022, σ. 382.