ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ ΜΕΤΑΛΛΕΙΩΝ- ΜΕΤΑΛΛΟΥΡΓΩΝ
ΤΟΜΕΑΣ ΜΕΤΑΛΛΟΥΡΓΙΑΣ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΛΙΚΩΝ

ΑΡΙΣΤΟΠΟΙΗΣΗ ΔΙΕΡΓΑΣΙΑΣ ΚΑΤΑΒΥΘΙΣΗΣ ΕΝΥΔΡΗΣ ΑΛΟΥΜΙΝΑΣ ΑΠΟ ΑΡΓΙΛΙΚΑ ΔΙΑΛΥΜΑΤΑ ΜΕ ΤΕΧΝΙΚΕΣ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Καλόγνωμος Αλέξανδρος

Επιβλέπων: Πάνιας Δημήτριος
Καθηγητής

ΑΘΗΝΑ, ΙΟΥΛΙΟΣ 2022

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ ΜΕΤΑΛΛΕΙΩΝ- ΜΕΤΑΛΛΟΥΡΓΩΝ
ΤΟΜΕΑΣ ΜΕΤΑΛΛΟΥΡΓΙΑΣ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΛΙΚΩΝ

ΑΡΙΣΤΟΠΟΙΗΣΗ ΔΙΕΡΓΑΣΙΑΣ ΚΑΤΑΒΥΘΙΣΗΣ ΕΝΥΔΡΗΣ ΑΛΟΥΜΙΝΑΣ ΑΠΟ ΑΡΓΙΛΙΚΑ ΔΙΑΛΥΜΑΤΑ ΜΕ ΤΕΧΝΙΚΕΣ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Καλόγνωμος Αλέξανδρος

Επιβλέπων: Πάνιας Δημήτριος
Καθηγητής

Εγκρίθηκε από την τριμελή επιτροπή στις 18/7/2022

Δημήτριος Πάνιας, Καθηγητής

Άνθιμος Ξενίδης, Καθηγητής

Σπύρος Παπαευθυμίου, Αναπληρωτής Καθηγητής

ΑΘΗΝΑ, ΙΟΥΛΙΟΣ 2022

Ευχαριστίες
Μ. Μαυροφοράκης, Μ. Βαφείας, Δ. Πάνιας, Ν. Κλάδης

Περιεχόμενα

1. Παραλλαγή της Μεθόδου Bayer με Καταβύθιση Βαιμίτη

1.1 Εισαγωγή

Το 2003 είχε πραγματοποιηθεί μία σειρά από εργαστηριακά πειράματα όσον αφορά την κινητική της αντίδρασης καταβύθισης βαιμίτη από υπέρκορα αργιλικά διαλύματα καυστικού νατρίου. Σκοπός των πειραμάτων ήταν να σχεδιαστεί το κατάλληλο αναλυτικό μοντέλο συναρτήσει της θερμοκρασίας (T), της συγκέντρωσης σε ισοδύναμο οξείδιο του νατρίου, του λόγου φύτρων βαιμίτη, το οποίο να υπολογίζει την απόδοση της αντίδρασης ως προς τον χρόνο. Στόχος ήταν να ερευνηθεί η πιθανότητα εφαρμογής μιας παραλλαγής της μεθόδου Bayer, όπου να καταβυθίζεται βαιμίτης αντί για γιββσίτης, λόγω της μικρότερης απαίτησης σε ενέργεια για παραγωγή αλούμινας σε επόμενο στάδιο της μεθόδου Bayer.

1.2 Παραγωγή Αλούμινας

Η αλούμινα παράγεται σχεδόν αποκλειστικά με την μέθοδο Bayer. Άλλες μέθοδοι παραγωγής αλούμινας, δευτερεύουσας σημασίας όμως, είναι η μέθοδος Deville-Pechiney, ο συνδυασμός των μεθόδων Bayer-Deville, οι μέθοδοι Pedersen, Blanc, Aluminium Co [1], με το βασικό μετάλλευμα από το οποίο εξάγεται η αλούμινα να είναι ο βωξίτης [2].

Με την συμβατική χρήση τις μεθόδου Bayer η μέση ενεργειακή κατανάλωση υπολογίζεται σε 11.6 GJ/t παραγόμενης άνυδρης αλούμινας. Το πείραμα εξέτασε την εναλλακτική να καταβυθιστεί βαιμίτης στους 90°C αντί για γιββσίτη και αντίστοιχα κάνοντας χρήση φύτρων βαιμίτη αντί για φύτρα γιββσίτη, οπότε και η παράγωγη της άνυδρης αλούμινας θα προκύψει από την πύρωση του καταβυθισμένου βαιμίτη. Προβλέπεται η εξοικονόμηση ενέργειας να είναι της τάξης του 1.8 GJ/t παραγόμενης αλούμινας, δηλαδή μείωση κατά 60% στην ενέργεια του σταδίου πύρωσης και κατά 15% στην συνολική ενεργειακή κατανάλωση της διεργασίας. Αυτή η εξοικονόμηση στην ενέργεια οφείλεται στην διαφορά ενθαλπίας των δύο αντιδράσεων πύρωσης, όπως φαίνεται παρακάτω, και ανέρχεται 1.1 GJ/t αλούμινας [2].

$$\begin{array}{lll} \text{Γιββσίτης:} & Al_2O_3 \cdot 3H_2O \rightarrow Al_2O_3 + 3H_2O & \Delta H = 187\,kJ/mole \quad Al_2O_3 \\ \text{Βαιμίτης:} & Al_2O_3 \cdot H_2O \rightarrow Al_2O_3 + H_2O & \Delta H = 72\,kJ/mole \quad Al_2O_3 \end{array} \tag{1.1}$$

Ο Βαιμίτης έχει μόνο ένα mole νερού σε αντίθεση με τον γιββσίτη που περιέχει τρία, οπότε η μάζα του υλικού που τροφοδοτείται στην κάμινο πύρωσης είναι μόλις το 80% και σαν αποτέλεσμα ελαττώνεται ακόμα περισσότερα η ενεργειακή απαίτηση κατά 0.7 GJ/t παραγόμενης αλούμινας.

Στο 3 φαίνεται το διάγραμμα ροής της μεθόδου Bayer. Για περισσότερες πληροφορίες ο αναγνώστης μπορεί να ανατρέξει στην σχετική βιβλιογραφική αναφορά [3].

Table 1.1: Ορυκτά Αλουμινίου [2]

Όνομα	Ορυκτολογικός τύπος
Κορούνδιο	$\alpha\text{-}Al_2O_3$
Γιββσίτης	$\gamma\text{-}Al_2O_3 \cdot 3H_2O$
Βαιμίτης	$\gamma\text{-}Al_2O_3 \cdot H_2O$
Διάσπορο	$\alpha\text{-}Al_2O_3 \cdot H_2O$
Σπινέλλιος	$Al_2O_3 \cdot MgO$
Σιλλιμανίτης	$Al_2O_3 \cdot SiO_2$
Κρυόλιθος	$AlF_3 \cdot 3NaF$
Καολινίτης	$Al_2O_3 \cdot 2SiO_2 \cdot 2H_2O$
Αλουνίτης	$3Al_2O_3 \cdot K_2O \cdot 4SO_3 \cdot 6H_2O$
Λευκίτης	$Al_2O_3 \cdot K_2O \cdot 4SiO_2$

Table 1.2: Τυπική Σύσταση Βωξίτη [4]

Συστατικό	Ποσοστό κατά Βάρος %
Al_2O_3	30-60
Fe_2O_3	1-30
SiO_2	<0.5-10
TiO_2	<0.5-10
C	0.02 – 0.40
P_2O_5	0.02-1
CaO	0.1-2
V_2O_5	0.01 – 0.10
ZnO	0.002 – 0.10
Ga_2O_3	0.004 – 0.013
Cr_2O_3	0.003 – 0.30
S	0.02-0.1
F	0.01 – 0.10

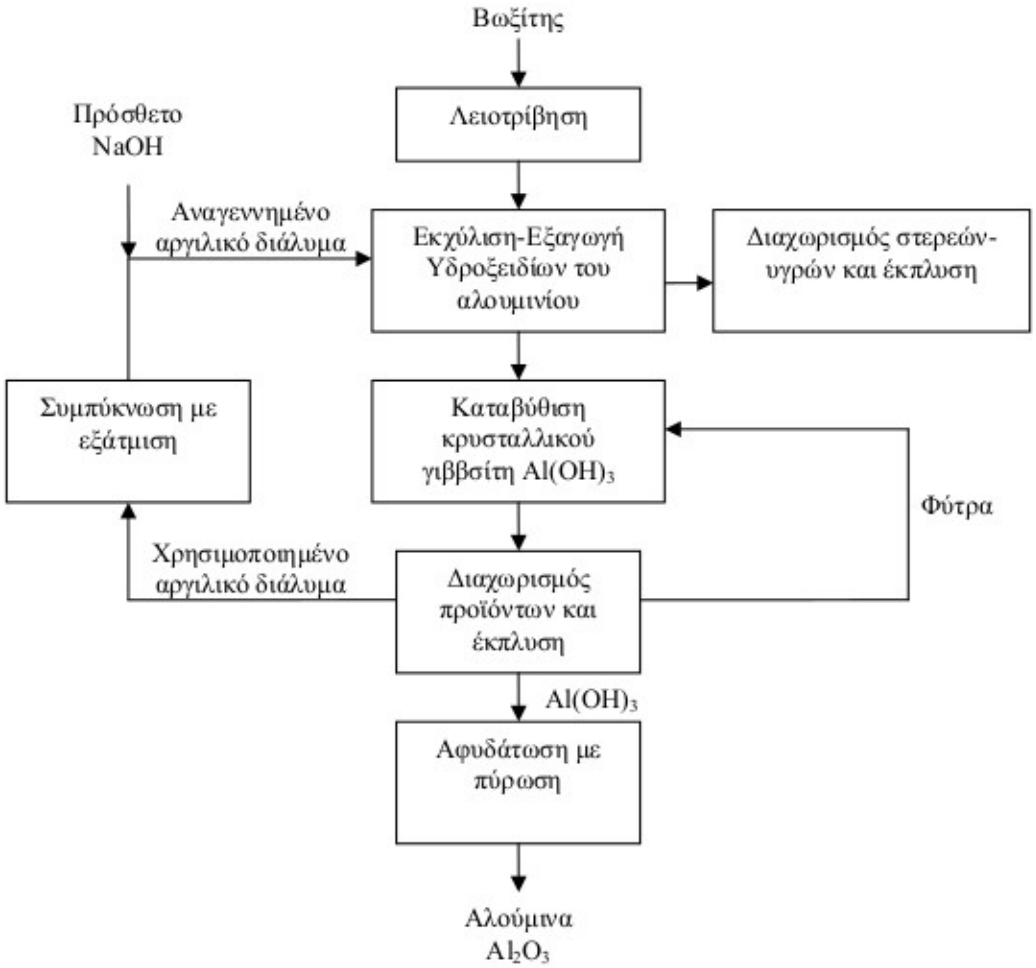

Figure 1.1: Μέθοδος Bayer [2]

1.3 Διαλυτότητα Γιββσίτη

Η διαλυτότητα του γιββσίτη σε θερμοκρασίες [25,100]°C έχει μελετηθεί εκτενώς, εφόσον αποτελεί ένα στάδιο της μεθόδου Bayer και επικρατεί η θεωρία ότι μπορεί να γίνει μία καλή προσέγγιση της διαλυτότητας του $Al(OH)_3$ σε διαλύματα καυστικού νατρίου από την σταθερά ισορροπίας της αντίδρασης.

$$Al(OH)_3 + OH^- \leftrightarrow AlO_2^- + 2H_2O \tag{1.2}$$

Με την σταθερά ισορροπίας για το σύστημα Al_2O_3 – Na_2O-H_2O να είναι

$$K = \frac{[AlO_2^-]\cdot(\alpha_{H_2O})^2}{[OH^-]} \tag{1.3}$$

$[AlO_2^-],[OH^-]$: μοριακές συγκεντρώσεις του αργιλικού ιόντος και του υδροξυλίου
(α_{H_2O}): η ενεργότητα του νερού στο διάλυμα του καυστικού νατρίου

Ο Misra στο [5] κάνει την παραδοχή 4 και καταλήγει στην εμπειρική σχέση 4, από την οποία προκύπτει η σχέση 4. Αυτή η σχέση θα χρησιμοποιηθεί για την χάραξη του καμπυλών διαλυτότητας γιββσίτη σε διαλύματα [70, 90, 110, 130] g/L Na_2O αντίστοιχα.

$$R_{eq} = \frac{Al_2O_3}{Na_2O} \tag{1.4}$$

$$lnR_{eq} = 6.2106 - \frac{2486.7}{(t+273)} + \frac{1.08752\cdot C}{(t+273)} \tag{1.5}$$

$$[Al_2O_3] = [Na_2O]\cdot e^{lnR_{eq}} \rightarrow$$
$$[Al_2O_3] = [Na_2O]\cdot R_{eq} \tag{1.6}$$

1.4 Διαλυτότητα Βαιμίτη

Η διαλυτότητα του βαιμίτη σε διαλύματα καυστικού νατρίου έχει μελετηθεί σε θερμοκρασίες [30-150]°C [6], [7] και έχει αναπτυχθεί το αντίστοιχο μοντέλο που την περιγράφει [8].

$$C_{eq} = A_1\cdot 10^{-6}\cdot T^3 + A_2\cdot 10^{-3}\cdot T^2 + A_3\cdot 10^{-2}\cdot T + A_4$$

$T \in [30, 150]°C$ και $C_{inNa2O} \in [60-140]$ g/L
C_{eq}: διαλυτότητα του βαιμίτη σε g/L
T: θερμοκρασία σε °C
C_{Na2O}: αρχική συγκέντρωση καυστικού νατρίου σε g/L Na_2O (1.7)

$$
\begin{aligned}
A_1 &= -0.0618925 \cdot C_{Na2O} + 1.36953 \\
A_2 &= 0.02301 \cdot C_{Na2O} + 0.1707 \\
A_3 &= 2.498 \cdot 10^{-6} \cdot C_{Na2O}^3 - 3.106 \cdot 10^{-4} \cdot C_{Na2O}^2 + 5.483 \cdot 10^{-2} \cdot C_{Na2O} - 1.332 \\
A_4 &= 3.236 \cdot 10^{-6} \cdot C_{Na2O}^3 - 7.887 \cdot 10^{-4} \cdot C_{Na2O}^2 + 1.584 \cdot 10^{-1} \cdot C_{Na2O} - 2.518
\end{aligned}
\tag{1.8}
$$

Από την σχέση 4 υπολογίζονται οι καμπύλες της διαλυτότητα βαιμίτη σε διάλυμα καυστικού νατρίου [70, 90, 110, 130] g/L και γίνεται η σύγκριση με τις αντίστοιχες καμπύλες διαλυτότητας του γιββσίτη. Συγκρίνοντας τις καμπύλες διαλυτότητας των δύο φάσεων γίνεται εμφανής η διαφορά στην διαλυτότητα τους. Ο βαιμίτης είναι λιγότερο διαλυτός από τον γιββσίτη, κάτι που διευκολύνει την καταβύθιση του.

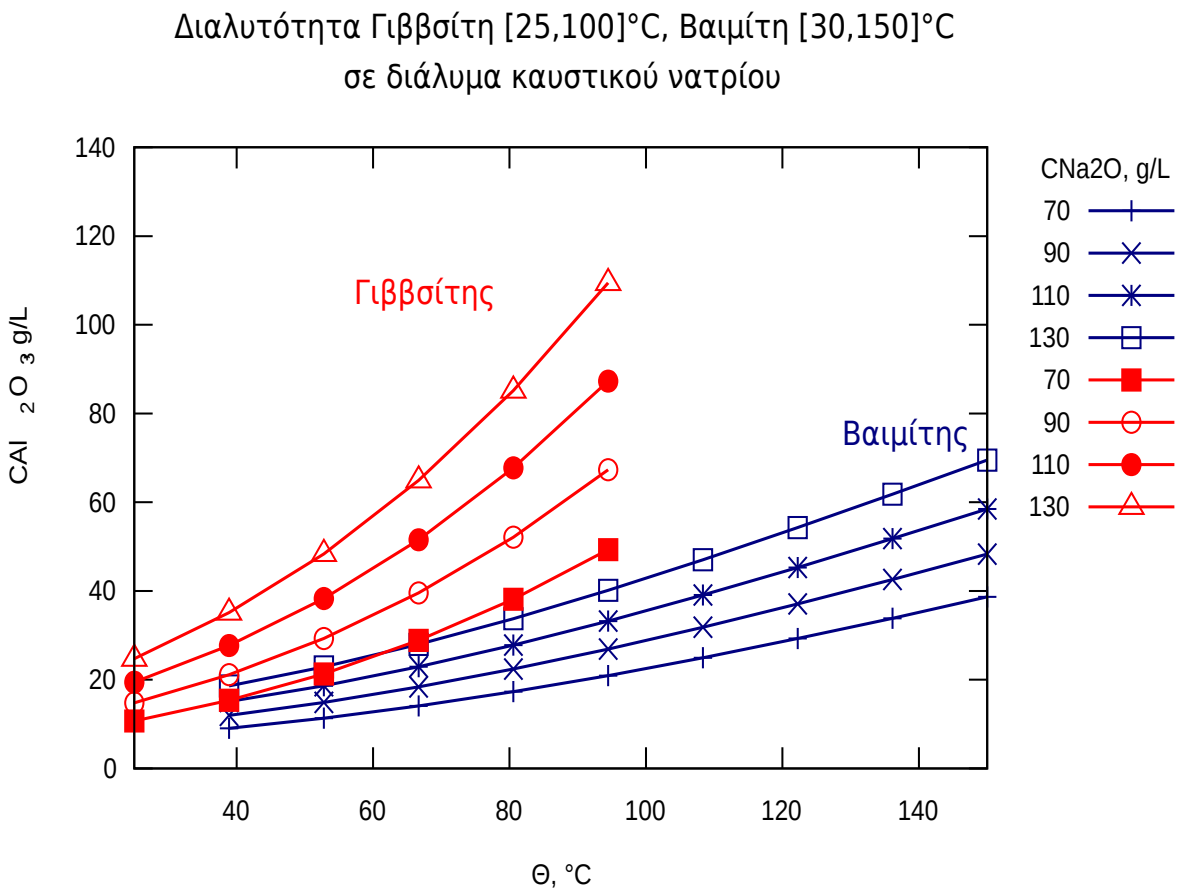

Figure 1.2

1.5 Παράγοντες στην Καταβύθιση Βαιμίτη

Θερμοκρασία (Temp)

Αρχικά μελετάται η επίδραση της θερμοκρασίας, οπότε μεταβάλλοντας τον παράγοντα της θερμοκρασίας οι υπόλοιποι παράγοντες παραμένουν σταθεροί, $CinAl_2O_3$ = 132 g/L, CNa_2O = 120 g/L, S.R. = 1.76 και η ταχύτητα ανάδευσης του πολφού (300min^{-1}).

Table 1.3: Επίδραση Θερμοκρασίας [2]

	Θερμοκρασία			
Χρόνος (h)	90°C	100°C	110°C	120°C
0	131.97	132.57	132.4	132.85
3	124	121.21	113.36	107.83
6	113.87	110.89	103.76	101.15
9	107.77	106.14	100.75	96.52
24	96.83	96.86	91.23	89.24
27				89.04
30		93.68	87.55	88.03
48	91.59	89.47	84.68	85.65
54			83.86	85.01
72	87.18	84.01	81.67	82.77
96	83.87	79.44	79.8	81.74
99			79.13	
Διαλυτότητα Βαιμίτη σε Al_2O_3 g/L	34.64	39.08	43.74	48.59
Αρχικός Βαθμός Υπερκορεσμού	97.33	93.49	88.66	84.26
Απόδοση 96h	49.4%	56.8%	59.3%	60.7%

Αρχική Συγκέντρωση Καυστικού Νατρίου ($C_{in}Na_2O$)

Ο δεύτερος παράγοντας που εξετάστηκε, ήταν η επίδραση του καυστικού νατρίου στην κινητική της καταβύθισης βαιμίτη. Οι υπόλοιποι παράγοντες παραμένουν σε σταθερές τιμές, θερμοκρασία=110°C, $CinAl_2O_3$ = 132 g/L, S.R.=1.76 και η ταχύτητα ανάδευσης του πολφού (300min^{-1}).

Table 1.4: Επίδραση της συγκέντρωσης καυστικού νατρίου [2]

	$C_{in}Na_2O$			
Χρόνος (h)	70 g/L	90 g/L	100 g/L	120 g/L
0	77.09	99.69	109.69	132.4
3	60.26	78.79	83.6	113.36
6	55.87	72.12	78.82	103.76
9	53.71	69.99	76.23	100.75
24	47.72	63.72	68.57	91.23
30	46.92	60.11	67.09	87.55

48	44.78	58.88	62.29	84.68
54				83.86
72	42.89	56.86	62.36	81.67
96	42.42	55.31	62.36	79.8
99				79.13
Διαλυτότητα Βαιμίτη σε Al_2O_3 g/L	25.45	32.44	36.06	43.74
Αρχικός Βαθμός Υπερκορεσμού	51.64	67.25	73.63	88.66
Απόδοση στις 96h	67.1%	66.0%	64.3%	59.3%

Λόγος Φύτρων Βαιμίτη (Seed Ratio (SR))

Τέλος μελετάται η επίδραση του λόγου φύτρων βαιμίτη με τους υπόλοιπους παράγοντες να παραμένουν σταθεροί, θερμοκρασία=110°C, $CinNa_2O$=100 g/L, $CinAl_2O_3$ = 110 g/L, η ταχύτητα ανάδευσης του πολφού (300min^{-1}).

Table 1.5: Επίδραση του λόγου προστιθέμενων φύτρων βαιμίτη [2]

	Λόγος Φύτρων Βαιμίτη			
Χρόνος (h)	0.4	0.8	1.76	2.5
0	110.20	110.72	109.69	110.72
3	104.62	96.79	83.60	86.68
6	99.88	91.20	78.82	82.49
9	97.33	89.82	76.23	78.17
24	91.26	80.30	68.57	69.50
30	90.60	81.17	67.09	69.30
48	85.69	77.13	62.29	64.99
72		74.72	62.81	63.06
96		73.01	62.36	
Διαλυτότητα Βαιμίτη σε Al_2O_3 g/L	36.06	36.06	36.06	36.06
Αρχικός Βαθμός Υπερκορεσμού	74.14	74.66	73.63	74.66
Απόδοση στις 48h	33.1%	45.0%	64.4%	61.2%

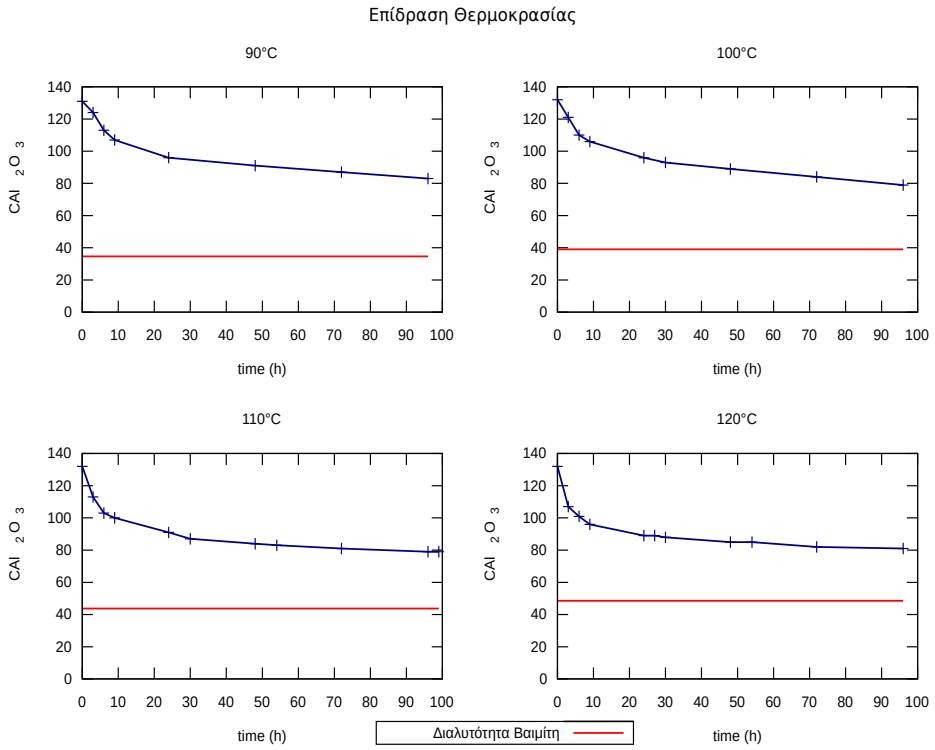

Figure 1.3 Επίδραση Θερμοκρασίας

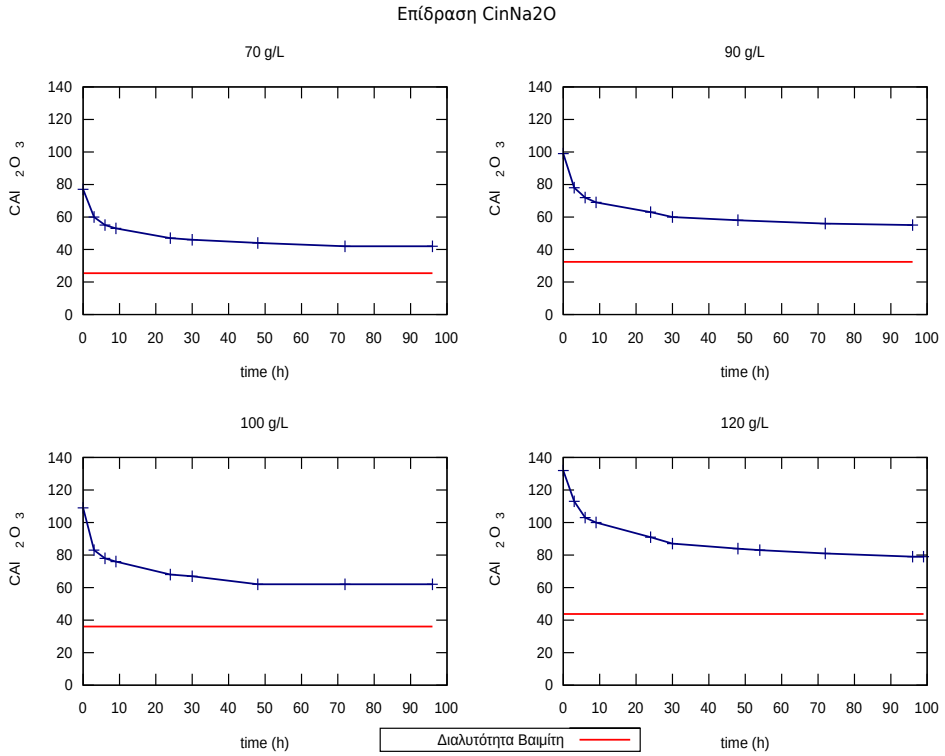

Figure 1.4 Επίδραση Αρχικής Συγκέντρωσης Καυστικού Νατρίου

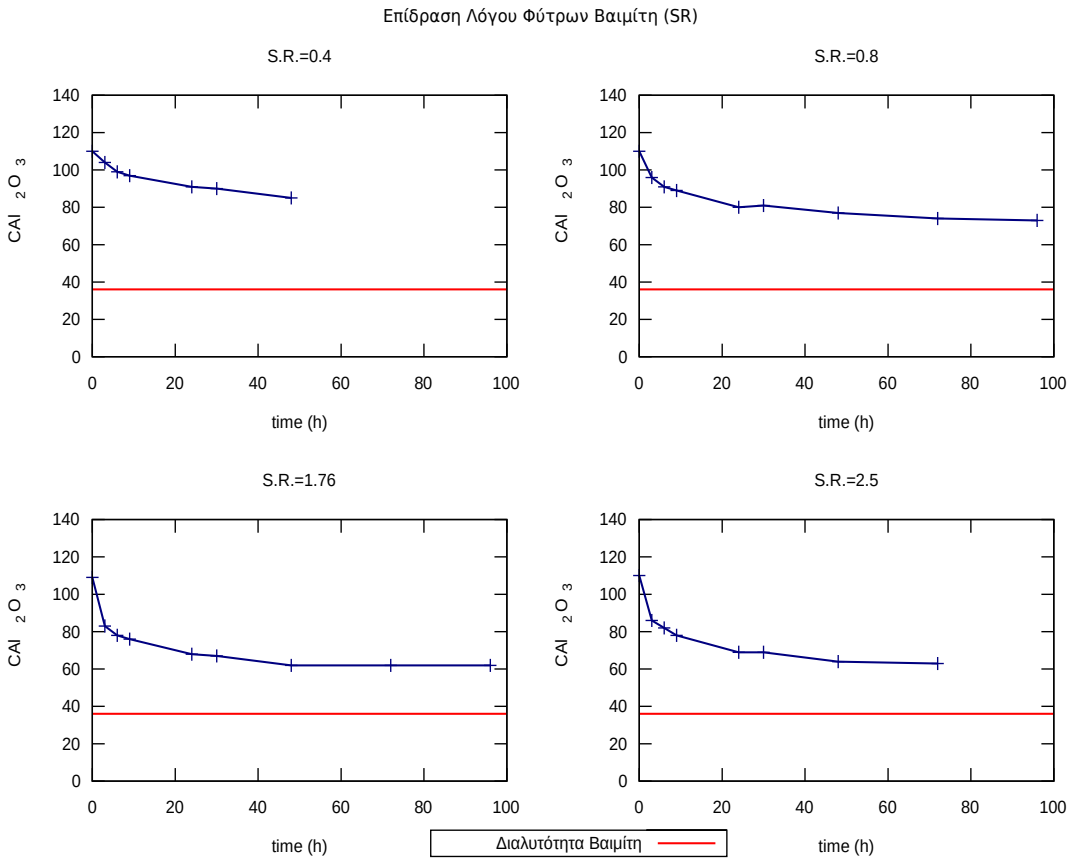

Figure 1.5: Επίδραση Φύτρων Βαιμίτη

1.6 Μοντέλο Κινητικής Καταβύθισης Βαιμίτη

Παρακάτω αναφέρεται το μοντέλο της κινητικής που σχεδιάστηκε από τα εργαστηριακά πειράματα καταβύθισης του βαιμίτη από υπέρκορο αργιλικό διάλυμα. Αναλυτική περιγραφή ολόκληρων των σταδίων σχεδιασμού του μοντέλου και ελέγχου ορθότητας του βρίσκεται στο κεφάλαιο 2.3.2 [2], [8].

Ως ωθούσα δύναμη στην καταβύθιση βαιμίτη στα υπέρκορα αργιλικά διαλύματα δρα ο βαθμός υπερκορεσμού (S), μαζί με την σταθερά (k) της αντίδρασης καταβύθισης. Ο βαθμός υπερκορεσμού του αργιλικού διαλύματος ορίζεται ως η διαφορά της πραγματικής συγκέντρωσης της αλούμινας και της φαινόμενης συγκέντρωσης ισορροπίας.

$$S = C - C_e^{app}$$

S: Βαθμός υπερκορεσμού
C: Συγκέντρωση του διαλύματος σε $g \quad Al_2O_3/L$
C_e^{app}: φαινόμενη συγκέντρωση ισορροπίας $g \quad Al_2O_3/L$ (1.9)

Στην περίπτωση της καταβύθισης βαιμίτη από υπέρκορα αργιλικά διαλύματα, το σύστημα πρακτικά φτάνει σε φαινόμενη ισορροπία, στην οποία η συγκέντρωση σε Al_2O_3 είναι υψηλότερη από την πραγματική συγκέντρωση ισορροπίας που ορίζει η διαλυτότητα.

Η καταβύθιση του βαιμίτη περιγράφεται από την παρακάτω σχέση.

$$\frac{dC_{pr}}{dt} = k \cdot \left(C - C_e^{app}\right)^n$$

$C_{pr} =$ Η ποσότητα του βαιμίτη που καταβυθίζεται ανά μονάδα όγκου διαλύματος εκφρασμένη ως $g \quad Al_2O_3/L$
C_e^{app}: η φαινόμενη συγκέντρωση ισορροπίας του διαλύματος εκφρασμένη ως $g \quad Al_2O_3$ (1.10)
C: Η συγκέντρωση σε Al_2O_3 τη χρονική στιγμή t, $g \quad Al_2O_3$
k: Σταθερά της ταχύτητας καταβύθισης
n: Η τάξη της ατνίδρασης ως προς τη συγκέντρωση
t: Ο χρόνος καταβύθισης

$$\frac{1}{C_{pr}} = \frac{1}{S_i^2 \cdot k} \cdot \left(\frac{1}{t}\right) + \frac{1}{S_i} \tag{1.11}$$

$$k = A^* \cdot e^{-\frac{E_\alpha}{R \cdot T}}$$

E_α: Η ενέργεια ενεργοποίησης
R: Η παγκόσμια σταθερά των αερίων, ίση με $8.3144 \; J/K \cdot mole$ (1.12)
T: Η απόλυτη θερμοκρασία
A^*: Προεκθετικός παράγοντας συχνότητας

$$k = A \cdot \left(C_{Na_2O}\right)^\alpha \cdot \left(SR\right)^b \cdot e^{-\frac{E_\alpha}{R \cdot T}} \tag{1.13}$$

Υπολογίζοντας κατά προσέγγιση τις τιμές των σταθερών, η τελική συνάρτηση που εκφράζει το κινητικό μοντέλο της καταβύθισης βαιμίτη σε υπέρκορο αργιλικό διάλυμα καυστικού νατρίου είναι η ακόλουθη.

$$\frac{dC_{pr}}{dt} = 2.30 \cdot 10^{13} \cdot \left(C_{Na_2O}\right)^{-1.8} \cdot \left(SR\right)^{0.54} \cdot e^{-\frac{10750}{T}} \cdot \left(C - C_e^{app}\right)^2$$

C_{pr}: Ποσότητα καταβυθισμένου βαιμίτη, $\left(g \quad Al_2O_3/L\right)$
C_e^{app}: Φαινόμενη συγκέντρωση ισορροπίας διαλύματος, $\left(g \quad Al_2O_3/L\right)$
C: Συγκέντρωση αλούμινας κατά τη χρονική στιγμή t, $\left(g \quad Al_2O_3/L\right)$ (1.14)
t: Διάρκεια καταβύθισης, h
C_{Na_2O}: Αρχική συγκέντρωση καυστικού νατρίου στο διάλυμα, $g \quad Na_2O/L$
SR: Λόγος προστιθέμενων φύτρων βαιμίτη,
g βαιμίτη ανά g οξειδίου αλουμινίου στο διάλμα
T: Απόλυτη θερμοκρασία, K

Στα επόμενα κεφάλαια, στα πλαίσια της αναφοράς στην Μηχανική Μάθηση, αλλά και στην Πολυπαραγοντική Βελτιστοποίηση, θα γίνεται συχνή αναφορά στο πείραμα είτε ως διάνυσμα ***X,*** είτε ως σημείο (του πολυδιάστατου χώρου που εξετάζεται), είτε ως *input*. Είναι πολύ σημαντικό να γίνει κατανοητός αυτός ο χαρακτηρισμός του πειράματος ως ένα διάνυσμα X των πέντε παραμέτρων:

i. Συγκέντρωση του διαλύματος σε οξείδιο του νατρίου
ii. Seed Ratio, δηλαδή τον αρχικό λόγο των προστιθέμενων φύτρων Βαιμίτη
iii. Βαθμός υπερκορεσμού του διαλύματος σε Al_2O_3. Πιο συγκεκριμένα ως Mass Ratio, δηλαδή ο λόγος g Al_2O_3 / g Na_2O
iv. Θερμοκρασία
v. Χρόνος αντίδρασης

$$\boldsymbol{X} = \left[C_{Na2O}, SR, MR, Temp, time\right] \quad (1.15)$$

ώστε στα επόμενα κεφάλαια, να μπορεί ο αναγνώστης να αντιληφθεί πως εφαρμόζονται αυτές οι τεχνικές στην περίπτωση της βελτιστοποίησης της Κινητικής Καταβύθισης Βαιμίτη από υπέρκορο αργιλικό διάλυμα.

Σε επόμενο κεφάλαιο που θα γίνει αναφορά στην υπολογιστική βιβλιοθήκη που σχεδιάστηκε για την παραγωγή “εικονικών” πειραματικών δεδομένων και την τροφοδότηση αυτών σε αλγορίθμους μηχανικής μάθησης, η υπολογιστική βιβλιοθήκη έχει ως βάση τις παραπάνω συναρτήσεις.

Αναλυτικότερες Πληροφορίες για τα εργαστηριακά πειράματα, τις διατάξεις που χρησιμοποιήθηκαν, τα κρυσταλλογραφικά αποτελέσματα του XRD, αλλά και όλες τις υπόλοιπες λεπτομέρειες, μπορούν να βρεθούν στις αντίστοιχες δημοσιεύσεις. [2], [9], [10], [11], [12], [13], [14].

1.7 Εφαρμογή Μηχανικής Μάθησης

Στο πείραμα είχε εφαρμοστεί η συμβατική προσέγγιση της σχεδίασης αναλυτικού μοντέλου βάσει των κλασικών συναρτήσεων της χημικής κινητικής. Εκ πρώτης όψεως πρόκειται για ένα πρόβλημα παραγοντικού σχεδιασμού που μελετάται η σχεδίαση κάποιου μοντέλου βάσει των παραγόντων $C_{in}Na_2O$, Seed Ratio (SR), θερμοκρασίας και χρόνου. Στο πλαίσιο αυτής της εργασίας εξετάζεται μία διαφορετική προσέγγιση, η χρήση της μηχανική μάθησης. Με την μηχανική μάθηση το μοντέλο που περιγράφει την κινητική της καταβύθισης βαιμίτη, δεν θα προκύψει επιλύοντας αναλυτικά συναρτήσεις της χημικής κινητικής, ούτε εφαρμόζοντας μεθοδολογίες παραγοντικού σχεδιασμού, αλλά εφαρμόζοντας αλγορίθμους μηχανικής μάθησης που δρουν πάνω σε δεδομένα και καταλήγουν σε κάποιο μοντέλο που εκφράζει τα πειραματικά δεδομένα με αυτοματοποιημένο τρόπο. Τα οφέλη στο συγκεκριμένο πρόβλημα μπορεί να μην είναι εμφανή, αλλά σε προβλήματα με περισσότερους παράγοντες και πιο σύνθετες αλληλεπιδράσεις μεταξύ αυτών, ο υπολογισμός του μοντέλου με τις συμβατικές μεθόδους ενδέχεται να είναι πολύ πιο περίπλοκος ή και αδύνατος.

2. Μηχανική Μάθηση

Μηχανική μάθηση ονομάζεται ο κλάδος της τεχνητής νοημοσύνης που εφαρμόζει αλγορίθμους για την ταυτοποίηση της συσχέτισης μεταξύ δεδομένων και πληροφορίας που εκφράζει το πρόβλημα. Με αυτό τον τρόπο, το πρόγραμμα μπορεί να κάνει προβλέψεις βάσει μελλοντικών δεδομένων (input), χωρίς να έχουν την ανάγκη να προγραμματιστούν αποκλειστικά για αυτό το σκοπό.[15]

Αλγόριθμος ονομάζεται τα μαθηματικά βήματα για την επίλυση ενός προβλήματος. Στην περίπτωση της μηχανικής μάθησης, πιο συγκεκριμένα, είναι τα στάδια για την παραγωγή ενός μοντέλου. Διαφορετικοί αλγόριθμοι παράγουν μοντέλα με διαφορετικά χαρακτηριστικά. Για κάθε πρόβλημα μηχανικής μάθησης (classification, regression, clustering, recommendation, forecasting, κλπ) υπάρχει και μία διαφορετικά σειρά αλγορίθμων. Η επιλογή του καταλληλότερου αλγορίθμου εξαρτάται από:

i. την φύση του προβλήματος

ii. τα χαρακτηριστικά των δεδομένων

iii. τους διαθέσιμους υπολογιστικούς πόρους

2.1 Βασικές Έννοιες της Μηχανικής Μάθησης

Σε αυτό το σημείο θα γίνει αναφορά σε γενικούς όρους που επικρατούν στην μηχανική μάθηση και είναι απαραίτητο να γνωρίζει κάποιος [16].

- Classifier

  Μία μέθοδος που δέχεται νέα, άγνωστα δεδομένα και τα κατατάσσει στην καταλληλότερη κατηγορία.

- Confusion matrix (error matrix)

  Ένας πίνακας που εκφράζει την ακρίβεια του αλγορίθμου της κατηγοριοποίησης (classification) μέσω των στοιχείων του, κάνοντας σύγκριση της προβλεπόμενης κατηγοριοποίησης των δεδομένων από τον αλγόριθμο με την πραγματική κατηγοριοποίηση τους.

- Ακρίβεια (error rate)

  Ο λόγος των σωστών προβλέψεων από τον μοντέλο σε ένα πειραματικό υποσύνολο του dataset., το οποίο δεν χρησιμοποιήθηκε για την εκπαίδευση του μοντέλου.

$$accuracy(AC) = \frac{TP + TN}{TP + TN + FN + FP}$$

TP = true positive
TN = true negative
FN = false negative
FP = false positive

(2.1)

$$precision(P) = \frac{TP}{TP + FP} \tag{2.2}$$

- Κόστος (cost)

  Το μέτρο της ακρίβειας του μοντέλου, δηλαδή το μέτρο της απόκλισης της προβλεπόμενης τιμής από την πραγματική. Οι συναρτήσεις βελτιστοποίησης κατά την εκπαίδευση των μοντέλων μηχανικής μάθησης σκοπεύουν στην επίλυση μονοπαραγοντικού προβλήματος βελτιστοποίησης, πιο συγκεκριμένα στην ελαχιστοποίηση της συνάρτησης κόστου.

- Cross-validation

  Μία τεχνική αξιολόγησης της ακρίβειας του μοντέλου, κατά την οποία πριν την έναρξη της διαδικασίας εκμάθησης, τα δεδομένα διαχωρίζονται σε υποσύνολα (συνήθως 80% και 20%), όπου το μικρότερο εξ αυτών δεν χρησιμοποιείται στους αλγορίθμους εκμάθησης, αλλά μόνο μετά το πέρας της εκμάθησης για τον υπολογισμό του κόστους του μοντέλου και την αξιολόγηση (evalutation) του.

- Dataset

  Τα δεδομένα που διατίθονται για την εκπαίδευση του μοντέλου από τους αλγορίθμους μηχανικής μάθησης. Κάθε στήλη συνήθως ορίζει ένα χαρακτηριστικό (feature) και κάθε γραμμή, ένα μέλος του dataset.

- Διάσταση (dimension)

  Μία συλλογή από χαρακτηριστικά που περιγράφουν μία ιδιότητα του προβλήματος.

- Σημείο (Instance)

  Ένα διάνυσμα από χαρακτηριστικά (feature vector), το οποίο πρακτικά είναι ένα αντικείμενο που χρησιμοποιείται είτε στην εκπαίδευση του μοντέλου, είτε στην πρόβλεψη αποτελεσμάτων.

- Μοντέλο

  Η δομή που προκύπτει σαν αποτέλεσμα μετά το πέρας της εκπαίδευσης του αλγορίθμου. Περιέχει όλα τα βήματα για τον μετασχηματισμό των δεδομένων, που παρέχονται σαν input, στην κατάλληλη μορφή και τα μαθηματικές σχέσεις που περιγράφουν την συσχέτιση μεταξύ των features του dataset.

- Σχήμα (Schema)

  Τα μακροσκοπικά χαρακτηριστικά των ιδιοτήτων του dataset. πρακτικά η περιγραφή της σύνθεσης των δεδομένων πάνω στα οποία εκπαιδεύεται το μοντέλο. Έχει κυρίως πρακτική χρήση από την πλευρά του λογισμικού (στον τρόπο που αποθηκεύεται το εκπαιδευμένο μοντέλο π.χ)

- Supervised Learning

Οι τεχνικές μηχανικής μάθησης που αναγνωρίζουν την συσχέτιση μεταξύ ανεξάρτητων μεταβλητών (**X**, input) και εξαρτημένων μεταβλητών (**Y**, label) από τα δεδομένου που τους παρέχονται ως dataset. Για την επαλήθευση της ακρίβειας του μοντέλου κατά το στάδιο της εκπαίδευσής υπολογίζεται το σφάλμα, δηλαδή η απόκλιση των προβλεπόμενων τιμών από τις πραγματικές τιμές του dataset. Μετά το πέρας της εκπαίδευσης το μοντέλο θα μπορεί για καινούρια δεδομένα **X** να κάνει προβλέψεις τις αντίστοιχες τιμές **Y** των εξαρτημένων μεταβλητών.

- Unsupervised learning

  Αλγόριθμοι που προσπαθούν να ταυτοποιήσουν την συσχέτιση σε δεδομένα, χωρίς να τους προσδιορίσθουν συγκεκριμένες εξαρτημένες μεταβλητές (**Y**, labels) σε αυτά. Μερικά παραδείγματα εφαρμογής αυτής της κατηγορία αλγορίθμων μηχανικής μάθησης είναι σε προβλήματα data compression, outlier detection, classification, natural language learning κτλ. Σε αυτή την κατηγορία ανήκουν και προβλήματα τύπου Dimensionality Reduction, για τα οποία θα γίνει αναφορά σε επόμενη παράγραφο.

- Feature Vector

  Ένα διάνυσμα n-διαστάσεων. Οι διαστάσεις ενός τέτοιου διανύσματος μπορούν να ελαττωθούν με τεχνικές, όπως Permutation Feature Importance που αναφέρεται παρακάτω, και άλλες όπως Principal Component Analysis (PCA), Multilinear Subspace Reduction, Isomaps, Latent Semantic Analysis (LSA). Ο χώρος στον οποίο ανήκουν αυτά τα διανύσματα ονομάζεται feature space.

2.2 Προετοιμασία Δεδομένων

Τις περισσότερες φορές, τα δεδομένα δεν είναι σε κατάλληλη μορφή, για να χρησιμοποιηθούν σαν features και χρειάζονται τους ανάλογους μετασχηματισμούς. Στο συγκεκριμένο πρόβλημα που μελετήθηκε, το *.csv* που περιέχει τα δεδομένα δεν χρειάστηκε κάποιο μετασχηματισμό, αλλά για λόγους πληρότητας θα γίνει αναφορά στους μετασχηματισμούς που εφαρμόζονται συνήθως σε προβλήματα διαφορετικών περιπτώσεων μηχανικής μάθησης, όπως classification, computer vision κτλ. Δεν θα δοθεί έμφαση, μόνο ονομαστική αναφορά σαν πρώτη επαφή του αναγνώστη για συνήθεις περιπτώσεις datasets που η κατανάλωση αυτών να είναι ακατάλληλη για τροφοδοσία των αλγορίθμων. Υπενθύμιση, οι αλγόριθμοι περιμένουν αριθμητικές τιμές, όχι κείμενο, ούτε εικόνες ή οποιασδήποτε άλλης μορφής αρχείο.

Table 2.1: Μετασχηματισμοί για την προετοιμασία των δεδομένων πριν την εκπαίδευση του μοντέλου[17]

Μετασχηματισμοί Δεδομένων	
Column mapping and grouping	Missing values
Normalization and scaling	Feature selection

Conversions between data types	Feature transformations
Text transformations	Explainability transformations
Image transformations	Calibration transformations
Categorical data transoformations	Deep learning tranformations
Time series data transformations	Custom transformations

2.3 Ελάττωση Διαστάσεων (Dimensionality Reduction)

Η μηχανική μάθηση ευνοείται από την παροχή μεγάλων datasets για την αποδοτικότερη εκπαίδευση μοντέλων. Αυτό συνεπάγεται μεγαλύτερο χρόνο για την εκπαίδευση, περισσότερο θόρυβο στα δεδομένα και ανάγκη για περισσότερους υπολογιστικούς πόρους. Σε ορισμένες περιπτώσεις μπορεί ο χρήστης να μην κατανοεί το πρόβλημα και να παρέχει δεδομένα στον αλγόριθμο με στατιστικά ασήμαντη επίδραση. Για να αποφευχθούν τέτοιες περιπτώσεις, έχουν αναπτυχθεί τεχνικές που ελαττώνουν τις διαστάσεις του προβλήματος, αφαιρώντας στήλες (features) αν κριθούν περιττά για την εκπαίδευση του μοντέλου. Παρακάτω γίνεται αναφορά σε αντίστοιχες μεθόδους και την σχετική βιβλιογραφία. Στην περίπτωση του προβλήματος που εξετάζεται δεν κρίθηκε απαραίτητο να εφαρμοστεί κάποια τέτοια τεχνική, εφόσον οι διαστάσεις ήταν μόλις πέντε και όλες είχαν υπολογίσιμη επίδραση.

- Permutation Features Importance (PFI)
- Singular Value Decomposition (SVD)
- Principal Component Analysis (PCA)
- Factor Analysis (FA)
- Linear Discriminant Analysis (LDA)
- Lanczos Algorithm
- Backward Elimination

Περισσότερες πληροφορίες στην σχετική βιβλιογραφία [16], [18], [19], [20] .

2.4 Βελτίωση της Ακρίβειας του Μοντέλου

Η βελτίωση της ακρίβειας του μοντέλου δεν έχει να κάνει αποκλειστικά “πειράζοντας” τις υπερπαραμέτρους του αλγορίθμου, αλλά ερμηνεύοντας και το ίδιο το πρόβλημα. Μπορεί να γίνει απλά ακολουθώντας τους παρακάτω απλούς κανόνες, αν μετά το πέρας της εκπαίδευσης του αλγορίθμου παρατηρηθεί ότι το μοντέλο δεν έχει αρκετά υψηλή ακρίβεια στις προβλέψεις του.

1. Αναθεώρηση του προβλήματος, εφαρμογή κάποιας τεχνικής ελάττωσης των διαστάσεων.
2. Εισαγωγή περισσότερων δεδομένων (μεγαλύτερο dataset).

3. Προσδιορισμός συσχέτισης μεταξύ των features, πρέπει τα features να είναι ανεξάρτητες μεταβλητές, όχι εξαρτημένες.
4. Εκπαίδευση του μοντέλου με τα features που έχουν ουσιαστική επίδραση σε αυτό και χρησιμοποιώντας δεδομένα χωρίς υπερβολικό θόρυβο ως dataset.
5. Cross-validation
6. Κατάλληλη ρύθμιση των υπερπαραμέτρων
7. Δοκιμή πληθώρας αλγορίθμων, ίσως ο αρχικός αλγόριθμος που επιλέχθηκε να ήταν ακατάλληλος για το συγκεκριμένο πρόβλημα και να αστοχεί σε κάποιο σημείο. Αν συμβαίνει αυτό θα γίνει φανερό από τις αρνητικές τιμές των metrics σε ορισμένες από τις δοκιμές.

2.5 Αλγόριθμοι Μηχανικής Μάθησης

Ανεξαρτήτως της βιβλιοθήκης που εφαρμόζεται για την επιλογή του αλγορίθμου και την εκπαίδευση του μοντέλου, το διάγραμμα ροής της διαδικασίας ισχύει γενικά στην μηχανική μάθηση και όπως φαίνεται και από το 18, τα στάδια που το αποτελούν είναι, η φόρτωση του dataset από τον δίσκο, ο μετασχηματισμός των δεδομένων αν αυτός είναι απαραίτητος, η εκπαίδευση του μοντέλου μέχρι την επίτευξη ικανοποιητικής ακρίβειας και τέλος η αποθήκευση του στον δίσκο για μελλοντική χρήση.

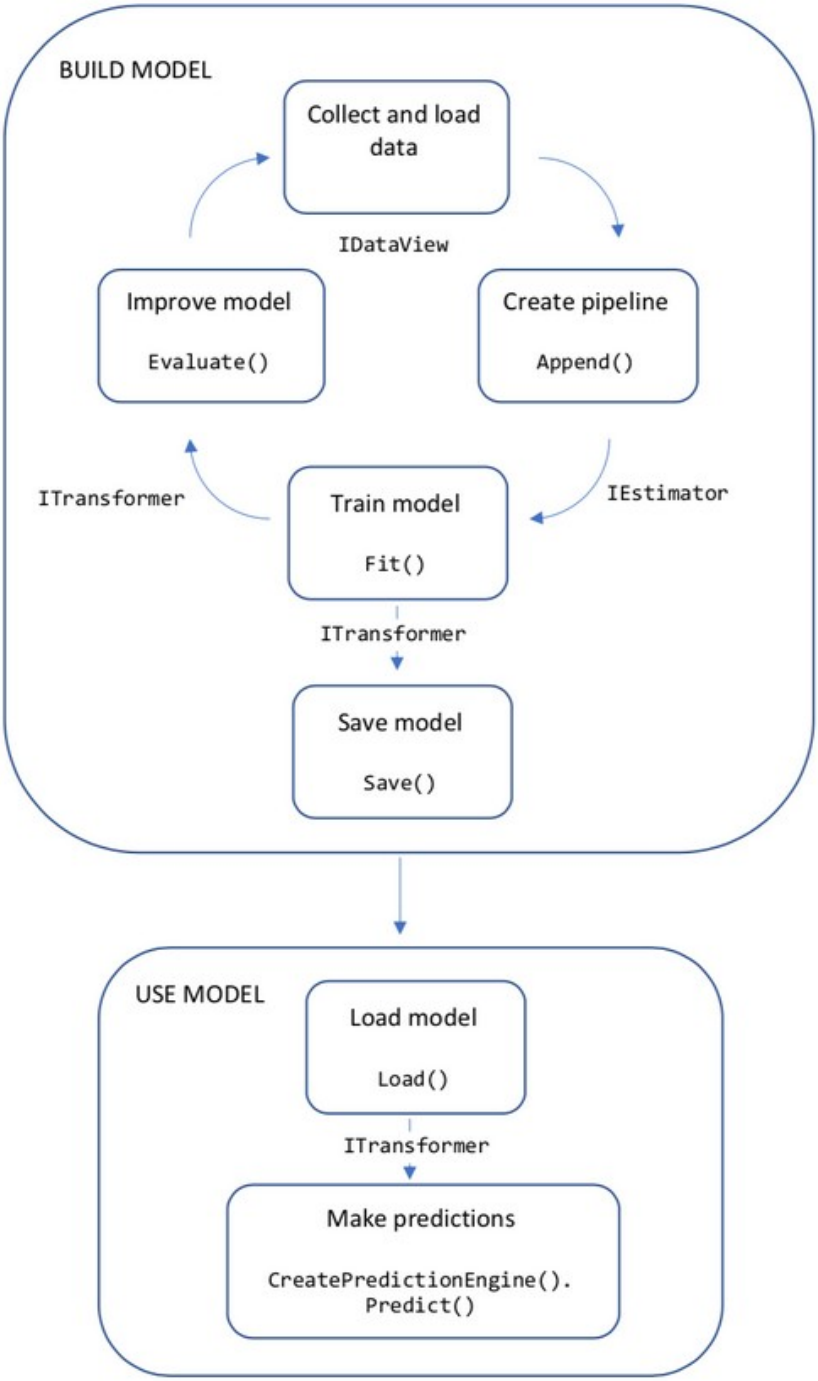

Figure 2.1: Διάγραμμα ροής ενός τυπικού αλγορίθμου μηχανικής μάθησης [15]

2.5.1 Support Vector Regression (SVR)

Το SVR είναι ένα πρόβλημα βελτιστοποίησης κατά το οποίο θεωρείται μία περιοχή *ε* γύρω από το εκπαιδευόμενο μοντέλο – συνάρτηση (σε αυτή την περίπτωση), όπου ονομάζεται (***ε***-tube), επειδή περικλείει την συνάρτηση σαν σωλήνας, όπως φαίνεται στην παρακάτω εικόνα. Βάσει αυτής της περιοχής ο αλγόριθμος προσπαθεί να προσδιορίσει την συνάρτηση του μοντέλου, κρατώντας περιορισμένη την πολυπλοκότητα και το σφάλμα. Πιο συγκεκριμένα, ορίζεται μία συνεχής συνάρτηση σφάλματος (που δεν λαμβάνει υπ' όψη τα σημεία εντός της περιοχής *ε* στον υπολογισμό του σφάλματος), με σκοπό να ελαχιστοποιηθεί κατά το στάδιο της βελτιστοποίησης και αυτός ο "σωλήνας" να περικλείει τα περισσότερα σημεία του dataset. Για την βελτιστοποίηση εφαρμόζονται οι κατάλληλες αριθμητικές μέθοδοι. Η υπερεπιφάνεια του μοντέλου εκφράζεται ως διανύσματα (Support Vectors) τα οποία είναι σημεία του dataset που βρίσκονται εκτός του χωρίου ***ε***.

Για παράδειγμα έστω ότι δίνεται η συνάρτηση

$$\begin{aligned} &y=f(x)=\langle w,x\rangle+b=\sum_{j=1}^{M} w_j x_j+b \\ &y,b\in\Re \\ &x,w\in\Re^M \end{aligned} \tag{2.3}$$

$$\begin{aligned} &f(x)=\left[\frac{w}{b}\right]^T\cdot\left[\frac{x}{1}\right]=w^T x+b \\ &x,w\in\Re^{M+1} \end{aligned} \tag{2.4}$$

το SVR θα προσπαθήσει να βρει το πιο στενό χωρίο ***ε*** γύρω από την επιφάνεια, ενώ παράλληλα ελαχιστοποιεί το σφάλμα, δηλαδή την διαφορά των προβλεπόμενων τιμών του μοντέλου από τις πραγματικές τιμές του προβλήματος.

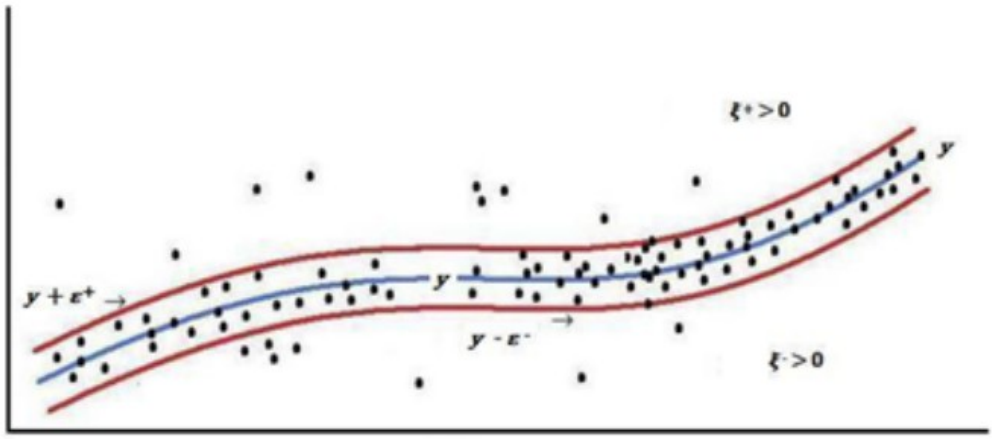

Figure 2.2: παράδειγμα ε-tube του SVR [16]

Για την παραπάνω περίπτωση προκύπτει η αντικειμενική συνάρτηση

$$min_w \frac{1}{2}||w||^2 \tag{2.5}$$

$||w||$: ο βαθμός του κανονικού διανύσματος (normal vector) στην επιφάνεια που εκφράζεται από το μοντέλο

Όπως αναφέρθηκε πριν η συνάρτηση σφάλματος είναι ένα σημαντικό τμήμα του αλγορίθμου, αφού εκεί βασίζεται η διαδικασία της ελαχιστοποίησης, συνήθως είναι συμμετρικές και συνεχής, ώστε να είναι βέβαιο ότι υπάρχει ρίζα που μπορεί να βρεθεί σε πεπερασμένο αριθμό επαναλήψεων, αν και μπορούν να χρησιμοποιηθούν και μη-συμμετρικές συναρτήσεις.

Παραθέτονται μερικές ενδεικτικές συναρτήσεις σφάλματος:

γραμμική (α): $$L_\varepsilon(y, f(x,w)) = \begin{cases} 0 & |y - f(x,w)| \le \varepsilon \\ |y - f(x,w)| - \varepsilon & |y - f(x,w)| > \varepsilon \end{cases} \tag{2.6}$$

τετραγωνική (β): $$L_\varepsilon(y, f(x,w)) \begin{cases} 0 & |y - f(x,w)| \le \varepsilon \\ (|y - f(x,w)| - \varepsilon)^2 & |y - f(x,w)| > \varepsilon \end{cases} \tag{2.7}$$

Huber (γ): $$L(y, f(x,w)) = \begin{cases} c|y - f(x,w)| - \frac{c^2}{2} & |y - f(x,w)| > c \\ \frac{1}{2}|y - f(x,w)|^2 & |y - f(x,w)| \le c \end{cases} \tag{2.8}$$

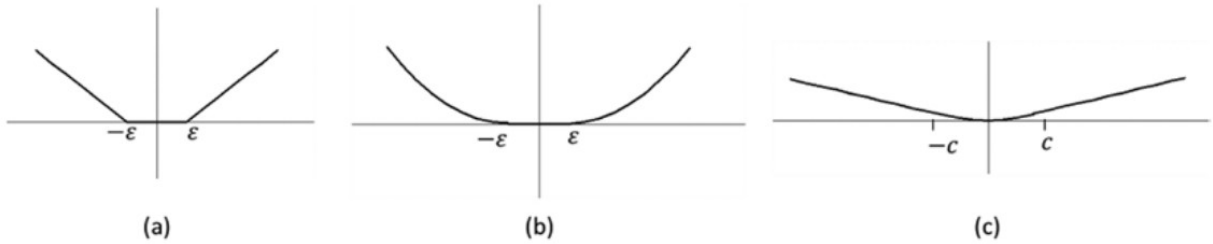

Figure 2.3: Συναρτήσεις Σφάλματος: (α) γραμμική, (β): τετραγωνική, (γ) Huber [16]

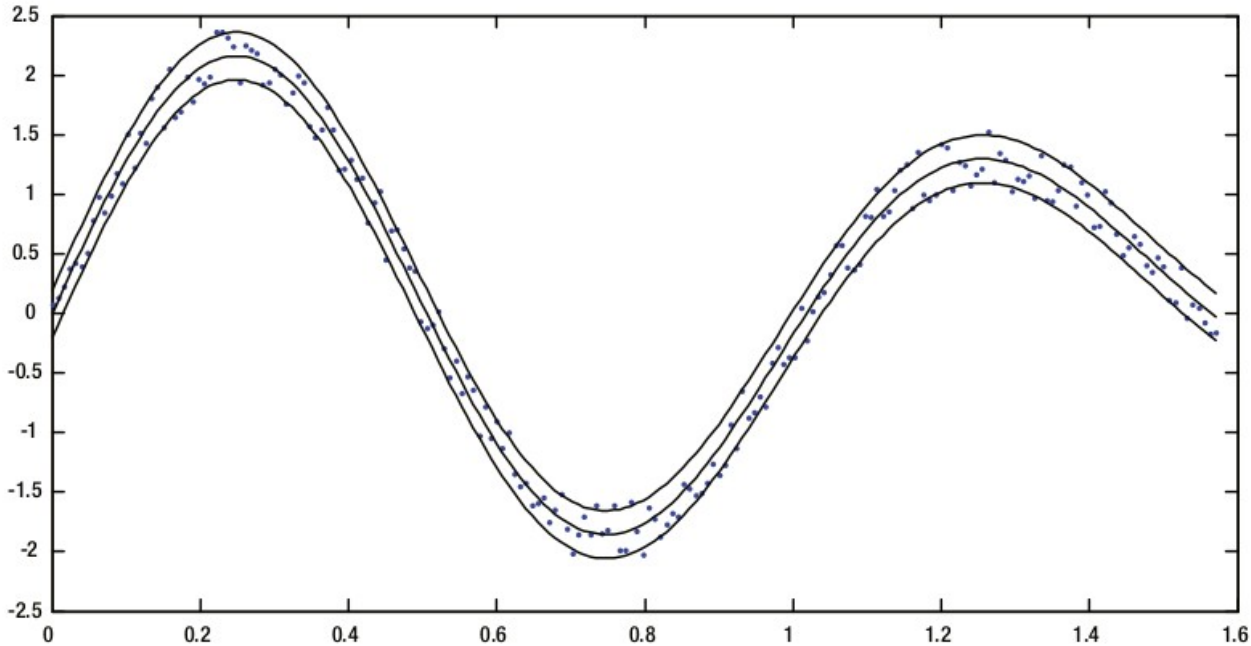

Figure 2.4: παράδειγμα SVR: μη-γραμμική παλινδρόμηση [16]

2.5.2 AdaBoost (Adaptive Boosting)

Μέθοδος που χρησιμοποιεί συνδυαστικά απλούς classifiers μικρής ακρίβειας, για την δημιουργία ενός αποδοτικότερου classifier. Αυτή η προσέγγιση είναι πολύ δημοφιλής καθώς βρίσκει εφαρμογή και σε άλλες σειρές αλγορίθμων με εξαιρετικά αποτελέσματα τόσο στην ακρίβεια των εκπαιδευμένων μοντέλων, όσο και στον χρόνο εκπαίδευσης, Ο ισχυρότερος classifier που παράγεται είναι στην πραγματικότητα ο γραμμικός συνδυασμός των αδύναμων classifiers.

$$H(x)=\sum_{t=1}^{T} b_t \cdot h_t(x)$$

$H(X)$ = ισχυρός classifier (2.9)

$h_t(x)$ = ασθενής classifier (feature)

T: επαναλήψεις

Ο αλγόριθμος λειτουργεί ως εξής:

$$D_1^i=\frac{1}{m} \tag{2.10}$$

D_1^i: ορισμός κατανομής βάρους

$$h_t=L(I, D_t)$$

h_t: εκπαίδευση ασθενούς learner (2.11)

I: αποτέλεσμα προηγούμενου learner

$$e_t = \sum_t D_t^i \left| h_t(x_i) - y_i \right| \tag{2.12}$$

e_t: υπολογισμός σφάλματος του h_t

$$\beta_t = \frac{1}{2} \ln \frac{1 - e_t}{e_t} \tag{2.13}$$

β_t: βάρος του h_t

$$D_{t+1}^i = \frac{D_t^i}{Z_t} \cdot e^{-\beta_t \cdot y_i \cdot h_t(x_i)} \tag{2.14}$$

D_{t+1}^i: επαναϋπολογισμός της κατανομής βάρους

Z_t: ο παράγοντας κανονικοποίησης

τελικό αποτέλεσμα του αλγορίθμου:

$$H(x) = sign\left(\sum_{t=1}^{T} \beta_t h_t(x) \right) \tag{2.15}$$

$H(x)$: ισχυρός classifier

Κατά την εκμάθηση οι συντελεστές βάρους μεταβάλλονται ανάλογα, με σκοπό να ελαττωθεί το σφάλμα των προηγούμενων επαναλήψεων. Τέλος το μοντέλο θα έχει μεγαλύτερη ακρίβεια στις προβλέψεις του από τους επί μέρους αδύναμους learners που το αποτελούν. Η απλότητα του AdaBoost το καθιστά από τους πιο δημοφιλείς αλγορίθμους, αφού είναι αρκετά απλό για να γραφθεί από την αρχή σε όποια εφαρμογή, και παράλληλα αρκετά γρήγορο και ευέλικτο.

2.5.3 Regression Trees

Μία ολόκληρη κατηγορία αλγορίθμων μηχανικής μάθησης είναι τα Classification and Regression Trees (CART), όπου πρόκειται για μη-παραμετρικά δένδρα αποφάσεων. Όπως θα δούμε σε επόμενο κεφάλαιο για την περίπτωση του προβλήματος που μελετήθηκε, αλγόριθμοι αυτής της κατηγορίας είχαν την μεγαλύτερη απόδοση σε όλες τις δοκιμές. Από τον αρχικό κόμβο (ρίζα) του δένδρου, τα επόμενα επίπεδα σχηματίζονται από διαδοχικούς διαμερισμούς και κάθε κόμβος συνδέεται με μέχρι δύο κόμβους του επόμενου επιπέδου και με έναν κόμβο του προηγούμενου επιπέδου. Η δομή μοιάζει με την κλασική δομή binary tree από την θεωρεία γραφημάτων. Μετά την πλήρη ανάπτυξη του δένδρου, ο αλγόριθμος εξετάζει ποιοι κλάδοι του δένδρου έχουν την μικρότερη συμβολή στην εκπαίδευση του μοντέλου και τους αφαιρεί. Η διαδικασία είναι επαναληπτική μέχρι το μοντέλο, στην προκειμένη περίπτωση το δένδρο, να αποκτήσει ικανοποιητική ακρίβεια στις προβλέψεις του.

Για περισσότερες πληροφορίες στις δομές της θεωρίας γραφημάτων παραθέτεται μία σειρά από σχετική βιβλιογραφία για το συγκεκριμένο πεδίο [21], [20], [22].

2.5.4 XGBoost (eXtreme Gradient Boosting)

Ο αλγόριθμος XGBoost είχε την μεγαλύτερη απόδοση από την πλευρά του sci-kit learn, οπότε και θα δοθεί η ανάλογη έμφαση σε αυτόν στις επόμενες παραγράφους. Είναι τύπου spanning-trees, οπότε ο τρόπος λειτουργίας του θυμίζει CART από την προηγούμενη παράγραφο.

Στόχος Εκπαίδευσης (training objective)

Έστω ότι η δομή αποτελείται από k δένδρα, οπότε το μοντέλο είναι ένα σύνολο από δένδρα αποφάσεων.

$$\sum_{k=1}^{K} f_k \tag{2.16}$$

f_k: η πρόβλεψη από κάθε δένδρο

Μετά την δημιουργία όλων τον δένδρων, η πρόβλεψη από το εκπαιδευμένο μοντέλο γίνεται ως:

$$\hat{y}_i^{(t)} = \sum_{k=1}^{t} f_k(x_i) \tag{2.17}$$

x_i: το διάνυσμα του σημείου i

$\hat{y}_i^{(t)}$: η πρόβλεψη από την επανάληψη t

Στην περίπτωση του Regression, για συνάρτηση κόστους χρησιμοποιείται η συνάρτηση των ελαχίστων τετραγώνων και η εκπαίδευση του μοντέλου βασίζεται στην ελαχιστοποίηση της.

$$L = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{2.18}$$

Ένας άλλος σημαντικός παράγοντας για το μοντέλο είναι το regularization. Δηλαδή ο έλεγχος για την διατήρηση της πολυπλοκότητας του μοντέλου σε χαμηλό επίπεδο και την αποτροπή του overfitting.

$$\Omega = \gamma \cdot T + \frac{1}{2} \lambda \sum_{j=1}^{T} w_j^2 \tag{2.19}$$

T: ο αριθμός των φύλλων (των κόμβων χωρίς σύνδεση σε κόμβους χαμηλότερου επιπέδου)

w_j^2: το score - ακρίβεια του φύλλου j

Συνδυάζοντας τις δύο εξισώσεις 23 και 23 προκύπτει η αντικειμενική συνάρτηση του μοντέλου:

$$Obj = L + \Omega \tag{2.20}$$

L: Loss function

Ω: regularization

Στον αλγόριθμο XGBoost η βελτιστοποίηση της αντικειμενικής συνάρτησης γίνεται μέσω του κλασικού αλγορίθμου Gradient Descent, όπου είναι ένας επαναληπτικός αλγόριθμος που υπολογίζει την παράγωγο της αντικειμενικής συνάρτησης

$$\partial_{\hat{y}} Obj(x, \hat{y}) \tag{2.21}$$

με σκοπό να την ελαχιστοποιεί σε κάθε επανάληψη. Εφόσον ο αλγόριθμος είναι επαναληπτικός, η αντικειμενική συνάρτηση μπορεί να γραφθεί ως:

$$Obj^{(t)} = \sum_{i=1}^{N} L(y_i, \hat{y}_i^{(t)}) + \sum_{i=1}^{t} \Omega(f_i) = \sum_{i=1}^{N} L(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \sum_{i=1}^{t} \Omega(f_i) \tag{2.22}$$

Η παράγωγος κάθε αντικειμενικής συνάρτησης είναι πιθανό να μην είναι διαθέσιμη. Σε αυτές τις περιπτώσεις υπολογίζεται κατά προσέγγιση η συνάρτηση Taylor δευτέρου βαθμού.

$$\begin{aligned} Obj^{(t)} &\simeq \sum_{i=1}^{N} [L(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \sum_{i=1}^{t} \Omega(f_i) \\ g_i &= \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)}) \\ h_i &= \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)}) \end{aligned} \tag{2.23}$$

η οποία ανάγεται σε

$$\Rightarrow Obj^{(t)} = \sum_{i=1}^{n} [g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t) \tag{2.24}$$

$Obj^{(t)}$: η αντικειμενική συνάρτηση για την επανάληψη t

Ανάπτυξη Δένδρου (Tree Building)

Ο αλγόριθμος του XGBoost προσπαθεί να “βρει” το δένδρο που κάνει την καλύτερη περιγραφή των δεδομένων για τον καθορισμό του μοντέλου. Γενικά ο αλγόριθμος XGBoost και οι υπόλοιποι αλγόριθμοι της κατηγορίας FastTree, προέρχονται από την θεωρία γραφημάτων, οπότε καλό είναι να γίνει η μελέτη στην σχετική βιβλιογραφία [21]. Όπως φαίνεται στο παρακάτω σχεδιάγραμμα το δένδρο αρχικά θα έχει μία δομή σαν την παρακάτω και με την πάροδο της εκπαίδευσης του μοντέλου, αυτή η δομή θα μεταλλάσσεται αναλόγως με σκοπό την βελτίωση στην ακρίβεια προβλέψεων.

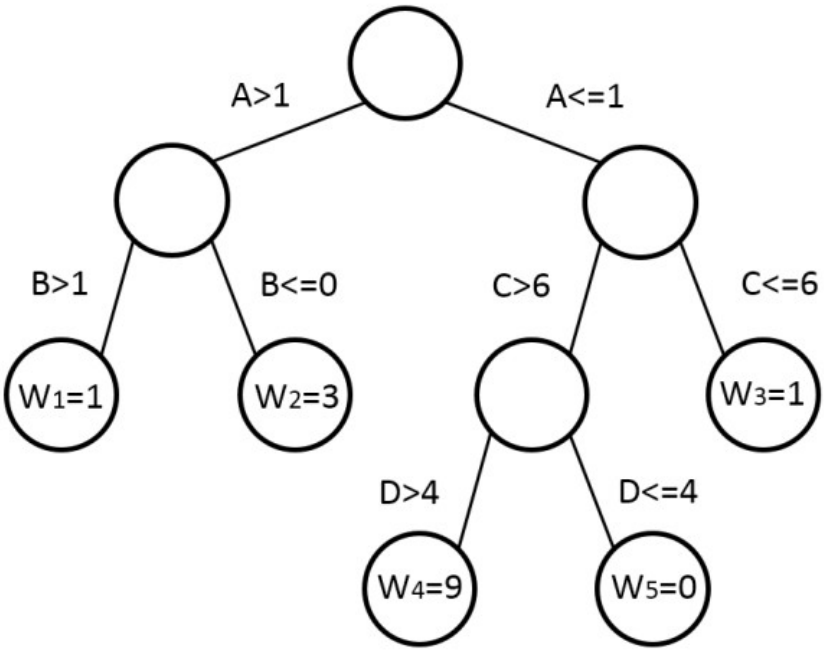

Figure 2.5: Δένδρο Αποφάσεων [0]

Για να το επιτύχει αυτό βασίζεται στις παραδοχές ότι οι εσωτερική κόμβοι του δένδρου διαχωρίζουν την ροή των δεδομένων για καθένα feature. Οι ακραίες συνθήκες ορίζουν τι δεδομένα περνάνε στον επόμενο κόμβο. Στα φύλλα του δένδρου ορίζεται ένας συντελεστής βάρους και αυτό το βάρους συμμετέχει σαν συντελεστής στην πρόβλεψη. Το δένδρο πρέπει να έχει μία “καλή” δομή, δηλαδή να πετυχαίνει στα φύλλα του, στους κόμβους στο τελευταίο επίπεδο, προβλέψεις υψηλής ακρίβειας.

Ο μαθηματικός ορισμός του δένδρου μπορεί να αποτυπωθεί ως:

$$f_t(x) = W_{q(x)}$$

$q(x)$: συνάρτηση που που ορίζει το κάθε σημείο (data point) σε ένα αντίστοιχο φύλλο του δένδρου

data point: σημείο από τα δεδομένα του dataset, δηλαδή μία γραμμή του .csv

$w_{q(x)}$: θέτει τα scores σε κάθε φύλλο του δένδρου

(2.25)

$$I_j = \{i | q(x_i) = j\}$$

I_j: index set, περιλαμβάει τα data points που αντιστοιχούν στο φύλλο j

(2.26)

Από τα παραπάνω, το score στα φύλλα του δένδρου υπολογίζεται από την συνάρτηση:

$$w_j = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda} \tag{2.27}$$

λ: παράμετρος για το regularization

$w_j \propto g, h$

Δηλαδή το w_j είναι ανάλογο των συναρτήσεων g και h 24. Οπότε για την καλύτερη ανάπτυξη του δένδρου πρέπει για κάθε feature, όταν ένα φύλλο του δένδρου διαχωρίζεται σε δύο νέους κόμβους να υπολογίζεται η συνάρτηση *gain* και με διαδοχικές επαναλήψεις να βρίσκεται ο καταλληλότερος κόμβος για διαμερισμό (split) κάθε φορά, μέχρι να διαμορφωθεί το δένδρο με την καλύτερη δομή. Οι κόμβοι με αρνητική τιμή gain αφαιρούνται από το δένδρο. Η τιμή της συνάρτησης gain υπολογίζεται για τους δύο κλάδους που αντιστοιχούν στον κόμβο j και η πορεία του datapoint κατευθύνεται προς στην μεριά με την υψηλότερη τιμή gain.

$$gain = \frac{1}{2}\left[\frac{\left(\sum_{i \in I_L} g_i\right)^2}{\sum_{i \in I_L} h_i + \lambda} + \frac{\left(\sum_{i \in I_R} g_i\right)^2}{\sum_{i \in I_R} h_i + \lambda} - \frac{\left(\sum_{i \in I} g_i\right)^2}{\sum_{i \in I} h_i + \lambda}\right] - \gamma \tag{2.28}$$

I: indices (δείκτες) των data points που αντιστοιχούν στον αρχικό κόμβο

I_L, I_R: indices (δείκτες) των data points που αντιστοιχούν στους δύο νέους κόμβους

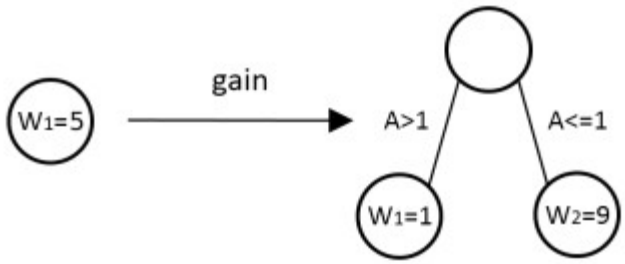

Figure 2.6: Node-split [0]

Συνοψίζοντας, ο αλγόριθμος XGBoost λειτουργεί με τα παρακάτω βήματα:

1. Εκτέλεση N-επαναλήψεων.
2. Ανάπτυξη δένδρου.
   i. Εύρεση του καταλληλότερου κόμβου για διαμερισμό (split).
   ii. Ορισμός συντελεστών βάρους στους δύο νέους κόμβους μετά το split.
3. Όσοι κόμβοι έχουν αρνητική τιμή gain αφαιρούνται από το δένδρο.

2.5.5 Artificial Neural Networks (ANNs)

Η ιδέα των νευρωνικών δικτύων προέρχεται ήδη από το 1949 όταν ο όρος Hebbian Learning αναφέρθηκε από τον ψυχολόγο Donald Hebb. Στηρίχθηκε στην ιδέα ότι όταν οι νευρώνες μιας σύναψης διεγείρονται ταυτόχρονα και αυτό επαναλαμβάνεται, η ισχύς τις σύναψης αυξάνεται ανάλογα.

$$w_{ij}(t+1)=w_{ij}(t)+\eta_{ij}\, x_i(t)\cdot x_j(t)$$

t: ο αριθμός της επανάλληψης
w_{ij}: το βάρος της σύναψης μεταξύ των νευρώνων i και j (2.29)
x_i: το αποτέλεσμα του i νευρώνα
η_{ij}: ο ρυθμός εκμάθησης της σύναψης μεταξύ των νευρώνων i και j

Με τον όρο νευρώνας εννοείται ο κάθε κόμβος του δικτύου, το όνομα προέρχεται από την βιολογία, λόγω ότι από είναι από εκεί είναι εμπνευσμένοι όπως και πολύ άλλοι αλγόριθμοι. Ο κανόνας του Hebb είναι μία τεχνική για το unsupervised-leaning που ανανεώνει τους συντελεστές βάρους στο δίκτυο. Η εκμάθηση κάθε σύναψης εξαρτάται μόνο από τα βάρη των νευρώνων που αντιστοιχούν σε αυτή. Η υλοποίηση αυτού του κανόνα σε αλγόριθμο είχε ήδη συμβεί από το 1954 και από αυτόν προέρχονται πολλές διαφορετικές εκδοχές νευρωνικών δικτύων. Την ίδια δεκαετία (1958) έγινε η εμφάνιση του αλγορίθμου perceptron, ενός απλού δικτύου δύο επιπέδων βασισμένο στο οπτικό σύστημα. Βασίζεται στην ιδέα ότι, ένας νευρώνας λαμβάνει σήμα (που προέρχεται από διαφορετικές περιοχές) από τους γειτονικούς του νευρώνες, αυτό το σήμα μπορεί να διεγείρει ή να καταστείλει τον νευρώνα. Αν η διέγερση είναι μεγαλύτερη από ένα προκαθορισμένο όριο ο νευρώνας αντιδρά.

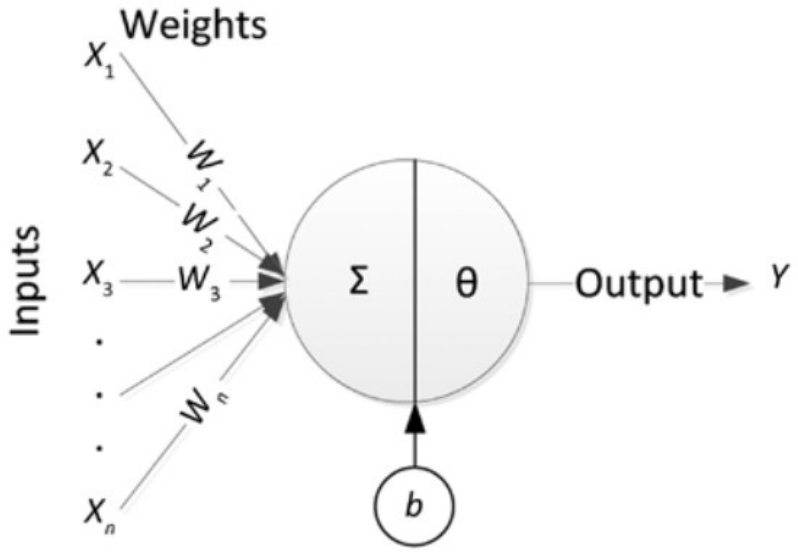

Figure 2.7: Διάταξη Νευρώνα [16]

Ένας νευρώνας απεικονίζεται με την παρακάτω διάταξη.

$$\begin{aligned} &w_i: \text{ συντελεστής βάρους} \\ &b: \text{ bias} \\ &\theta: \text{ συνάρτηση ενεργοποίησης} \end{aligned} \tag{2.30}$$

Μαθηματικά διατυπώνεται ως:

$$Y=\theta\left(\sum_{i=1}^{n} W_i X_i + b\right) \tag{2.31}$$

Διαφορετικά μπορεί να αποτυπωθεί ως πινάκας:

$$Y=\theta\left(\boldsymbol{W}\cdot\boldsymbol{X}+b\right) \tag{2.32}$$

$$\boldsymbol{W}=\begin{bmatrix} W_1 & W_2 & \cdots & W_n \end{bmatrix},\ \boldsymbol{X}=\begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{bmatrix} \tag{2.33}$$

Μερικές από τις συνήθεις συναρτήσεις ενεργοποίησης:

$$\text{Hard limiter:}\quad \theta(\alpha)=\begin{cases} 0 & a<0 \\ 1 & a>0 \end{cases} \tag{2.34}$$

$$\text{συνάρτηση saturating linear:}\quad \theta(\alpha)=\begin{cases} 0 & a<0 \\ \alpha & 0\leq a<1 \\ 1 & a>1 \end{cases} \tag{2.35}$$

$$\text{συνάρτηση log-sigmoid:}\quad \theta(\alpha)=\frac{1}{1+e^{-a}} \tag{2.36}$$

$$\text{υπερβολικής εφαπτομένης σιγμοειδούς συνάρτηση:}\quad \theta(\alpha)=\frac{e^{\alpha}-e^{-\alpha}}{e^{\alpha}+e^{-\alpha}} \tag{2.37}$$

Σε κάθε επίπεδο νευρώνων μπορούν να εφαρμοστούν διαφορετικές συναρτήσεις ενεργοποίησης, και κάθε επίπεδο του δικτύου μπορεί να αποτελείται από διαφορετικό αριθμό νευρώνων. Ο αριθμός των ενδιάμεσων “κρυφών” επιπέδων, εξαρτάται από το ίδιο το πρόβλημα. Μεγαλύτερος αριθμός επιπέδων μπορεί να προκαλέσει overfitting στο μοντέλο, ενώ μικρότερος underfitting. Το αποτέλεσμα από κάθε νευρώνα είναι η συνάρτηση 28, (*output*), πρακτικά αυτό σημαίνει ότι είναι ένας μη-γραμμικός μετασχηματισμός μιας υπερπιφάνειας.

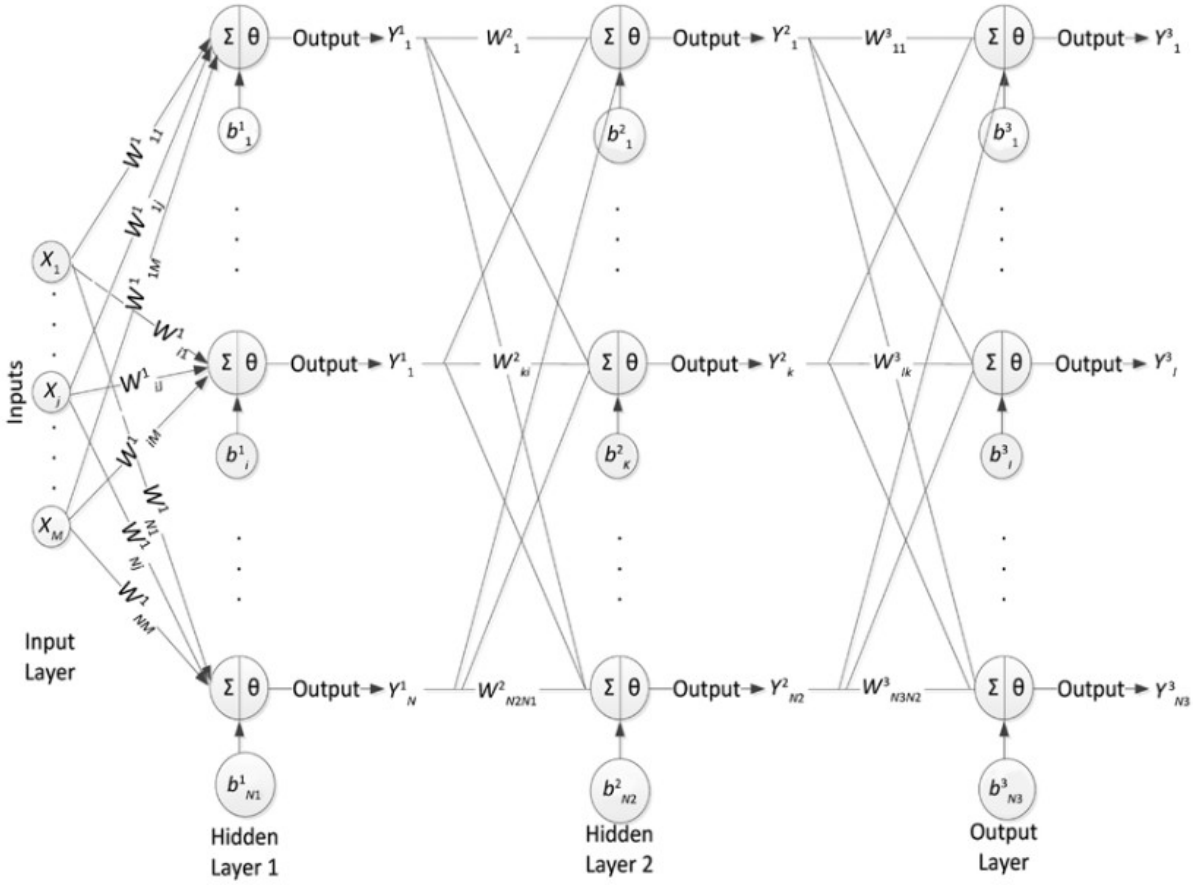

Figure 2.8: Παράδειγμα: Νευρωνικό Δίκτυο τριών(3) Επιπέδων [16]

2.5.6 Αλγόριθμος Backpropagation

Ο αλγόριθμος backpropagation είναι πάρα πολύ συνηθισμένος και εφαρμόζεται σε πάρα πολλές περιπτώσεις δικτύων. Στο στάδιο της εκμάθησης ο αλγόριθμος προσπαθεί να ελαχιστοποιήσει μία συνάρτηση κόστους βρίσκοντας τους βέλτιστους συντελεστές βάρους των νευρώνων.

$$E=\frac{1}{2}\sum_{i=1}^{P}\|Y_i-T_i^2\|$$

E: συνάρτηση κόστους
X: διάνυσμα M-διαστάσεων, input - πρώτο επίπεδο του δικτύου (2.38)
T: διάνυσμα N-διαστάσεων, output - τελευταίο επίπεδο του δικτύου
Y: ενδιάμεσο επίπεδο του δικτύου
i: το κάθε αντικείμενο i του dataset

Η συνάρτηση E μπορεί να φαίνεται γνώριμη, επειδή είναι είναι η συνάρτηση ελαχίστων τετραγώνων. Εναλλακτικά σαν συνάρτηση κόστους μπορεί να χρησιμοποιηθεί και η εντροπία. Η ανανέωση των συντελεστών βάρους δεν πραγματοποιείται σε κάθε επανάληψη του αλγορίθμου, αλλά μετά το πέρας μερικών επαναλήψεων, σύμφωνα με τον κανόνα του Hebb. Παρακάτω αναφέρονται τα στάδια του αλγορίθμου.

1. Initialization:

   Ορίζονται με τυχαίο τρόπο συντελεστές βάρους W_i στους κόμβους του δικτύου. Κάτι αντίστοιχο συμβαίνει και σε τελείως διαφορετικές οικογένειας αλγορίθμων σαν πρώτο στάδιο, όπως θα δούμε σε επόμενα κεφάλαια.

2. Feedforward:

   Στο πρώτο επίπεδο του δικτύου τροφοδοτούνται διανύσματα X_i σαν input και καταλήγουν στο τελευταίο επίπεδο σαν Y_i output, ύστερα υπολογίζεται η απόκλιση (σφάλμα) του output από τις πραγματικές τιμές.

3. Feedback:

   Ανανέωση των συντελεστών βάρους και προκατάληψης (bias).

$$W_{ij}^k(t+1)=W_{ij}^k(t)-a\frac{\partial E}{\partial W_{ij}^k} \tag{2.39}$$

$$b_i^k(t+1)=b_i^k(t)-a\frac{\partial E}{\partial b_i^k} \tag{2.40}$$

α: ρυθμός εκμάθησης $\alpha>0$ (2.41)

Η τιμή του α επηρεάζει πόσο γρήγορα συγκλίνει ο αλγόριθμος του backpropagation. Πολύ υψηλές τιμές μπορούν να τον αποτρέψουν από την σύγκλιση, ενώ χαμηλές τιμές θα συνεπάγονται αργή σύγκλιση οπότε και πολύ χρόνο για την ολοκλήρωση του. Όπως παρατηρούμε χρησιμοποιεί την παράγωγο της συνάρτησης E, αυτό σημαίνει ότι η συνάρτηση κόστους πρέπει να είναι παραγωγίσιμη, σε αντίθετη περίπτωση που κάτι τέτοιο δεν ισχύει, θα πρέπει να εφαρμοστεί κάποια τελείως

διαφορετική μέθοδος από το backpropagation. Όταν η τιμή της συνάρτησης κόστους γίνει μικρότερη από κάποιο προκαθορισμένο όριο (threshold), τότε ολοκληρώνεται η εκπαίδευση του μοντέλου.

Deep Learning

Η περίπτωση του deep learning απαιτεί χρήση πολλαπλών ενδιάμεσων επιπέδων νευρώνων για την εκπαίδευση ενός μοντέλου με ικανοποιητική ακρίβεια, δεν ήταν αναγκαία στην περίπτωση του προβλήματος που μελετά η εργασία, οπότε δεν εφαρμόστηκε κάποιο Deep Neural δίκτυο. Για λόγους πληρότητας παρατίθεται βιβλιογραφία [23], λόγω του ότι το deep learning βρίσκει ευρέα χρήση στις εφαρμογές μηχανικής μάθησης και ενδεχομένως σε διαφορετικά πιο σύνθετα προβλήματα μεταλλουργίας να ήταν χρήσιμη η εφαρμογή του.

2.5.7 Multilayer Perceptron

Multilayer Perceptron (MLP) ονομάζεται ένα το feedforward δίκτυο απλών νευρώνων, το οποίο αντιστοιχεί ένα σύνολο δεδομένων (input) σε ένα σύνολο αποτελεσμάτων (outputs). Ένα MLP αποτελείται από πολλαπλά επίπεδα κόμβων συνδεμένων ως γράφημα με προσανατολισμό προς μία κατεύθυνση, δηλαδή σαν δένδρο, όχι network και στο οποίο οι κόμβοι των ενδιάμεσων επιπέδων είναι νευρώνες με μη-γραμμικές συναρτήσεις ενεργοποίησης. Αυτό σημαίνει ότι συνδέονται μεταξύ τους μόνο κόμβοι-νευρώνες γειτονικών επιπέδων με μονόδρομες συνδέσεις που χαρακτηρίζονται από συντελεστές βάρους. Το MLP έχει το λιγότερο τρία επίπεδα , δηλαδή τείνει να είναι πολύ μικρό συγκριτικά με τα deep learning δίκτυα.

Ο αριθμός των κόμβων του πρώτου επιπέδου (input layer) είναι ανάλογος του αριθμού των features του dataset, ενώ ο αριθμός των κόμβων των ενδιάμεσων επιπέδων εξαρτάται από την φύση του ίδιου του προβλήματος, γίνονται προσαρμογές στον αριθμών των ενδιάμεσων επιπέδων και τον αριθμό των κόμβων αυτών μέχρι το μοντέλο να αποκτήσει αποδεχτή ακρίβεια, προσέχοντας πάντα την χρήση υπέρμετρου αριθμού επιπέδων και κόμβων που οδηγεί σε overfiting.

Η εκπαίδευση του μοντέλου τύπου supervised-learning μέσω ενός αλγορίθμου backprogation, κατά τον οποίο ο κάθε κόμβος υπολογίζει ένα αποτέλεσμα **y** από τον υπολογισμό ενός γραμμικού συνδυασμού του input με τον συντελεστή βάρους κάθε κόμβου και της μη-γραμμικής συνάρτησης ενεργοποίησης του κόμβου, όπως φαίνεται στην παρακάτω συνάρτηση.

$$y=\varphi\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{2.42}$$

w_i: συντελεστής βάρους
x_i: input του κόμβου
b: bias
φ: συνάρτητη ενεργοποίησης

Συνήθως η συνάρτηση ενεργοποίησης φ θα είναι:

$$\varphi = \frac{1}{1+e^{-x}} \quad \text{είτε} \quad \varphi = \tanh(x) \tag{2.43}$$

Το πλεονέκτημα αυτών των συναρτήσεων είναι ότι είναι γραμμικές κοντά στο *O(0,0)*, αλλά αυτό παύει να ισχύει όσο απομακρύνονται από αυτό.

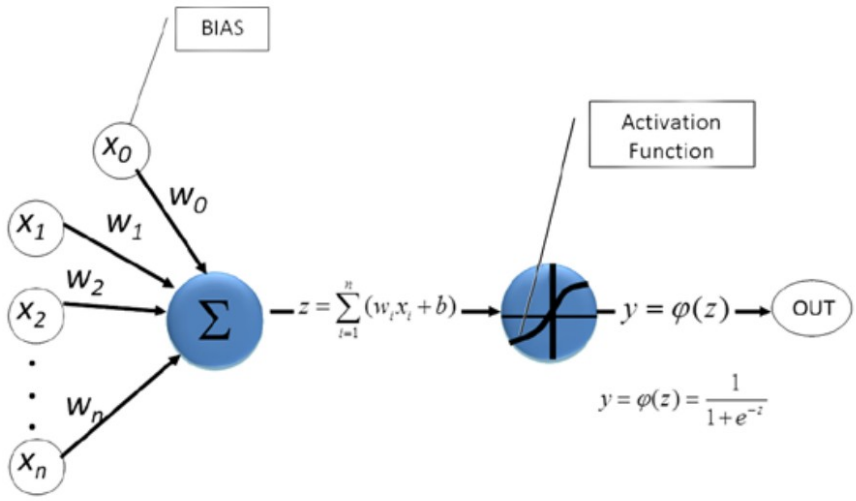

Figure 2.9: Multilayer Perceptron (MLP) [16]

Κατά την εκπαίδευση ο αλγόριθμος προσαρμόζει τους συντελεστές βάρους των κόμβων με σκοπό την ελάττωση της συνάρτησης κόστους. Αφού θέσει στην αρχή τυχαίους συντελεστές βάρους, στη συνέχεια τροφοδοτεί τους κόμβους με τα δεδομένα του input και επιστρέφει το σφάλμα προς τα πίσω (προς την αντίθετη φορά) ξεκινώντας από το τελευταίο επίπεδο, δηλαδή το output, με την ακόλουθη συνάρτηση σφάλματος:

$$\begin{aligned} & E(O_n(t)) = T_n - O_n(t) \\ & O_n(t): \text{ πραγματικές τιμές} \\ & T_n: \text{ επιθυμητές τιμές} \\ & E: \text{ σφάλμα} \end{aligned} \tag{2.44}$$

Τα βήματα του αλγορίθμου είναι τα ακόλουθα:

1. Ορίζονται τυχαίοι συντελεστές βάρους στους κόμβους στο διάστημα [-1, 1].
2. Στέλνεται το input από το πρώτο επίπεδο κόμβων στα εσωτερικά επίπεδα.

3. Υπολογίζεται το output (**Y**) του δικτύου.
4. Για κάθε κόμβο *n* στο τελευταίο επίπεδο (output layer):
   i. Υπολογίζεται το σφάλμα από την συνάρτηση σφάλματος.
   ii. Προστίθεται το σφάλμα σε όλους τους συντελεστές βάρους που συνδέονται με τον κόμβο *n*
5. Επιστροφή στο βήμα 2.

Για την επιτάχυνση του αλγορίθμου, χρησιμοποιείται μία παράμετρος εκμάθησης *η(<1)* κατά την οποία ελαττώνεται ο απαιτούμενος αριθμός επαναλήψεων για την σύγκλιση του αλγορίθμου, μειώνοντας το πλήθος των αριθμητικών πράξεων για τον επαναϋπολογισμό των συντελεστών βάρους.

$$w_{ij}(t+1)-w_{ij}(t)=\eta\cdot E(O_j(t)) \tag{2.45}$$

όπου το βάρος του κόμβου i που συνδέεται στο output j

3. Πολυπαραγοντική Βελτιστοποίηση

3.1 Βελτιστοποίηση

Βελτιστοποίηση ορίζεται η διαδικασία εύρεσης των τιμών που ελαχιστοποιούν ή μεγιστοποιούν την αντικειμενική συνάρτηση που χαρακτηρίζει το πρόβλημα και συνήθως χαρακτηρίζεται από ορισμένους περιορισμούς. Τα περισσότερα προβλήματα βελτιστοποίησης διαθέτουν το λιγότερο μία αντικειμενική συνάρτηση ή περισσότερες, όπως συμβαίνει στα προβλήματα πολυπαραγοντικής βελτιστοποίησης, αλλά υπάρχουν και ορισμένες εξαιρέσεις:

- Υπάρχουν περιπτώσεις που δεν υπάρχει καμία αντικειμενική συνάρτηση, όπως σε προβλήματα που ο στόχος είναι να βρεθούν οι τιμές για τις οποίες ικανοποιούνται όλοι οι περιορισμοί του προβλήματος και δεν είναι απαραίτητο να διατυπωθεί κάποια αντικειμενική συνάρτηση. Τέτοιου είδους προβλήματα ονομάζονται *feasibility problems*.
- Στα πολυπαραγοντικά προβλήματα πολλές φορές οι αντικειμενικές συναρτήσεις αντικρούονται μεταξύ τους και η βελτιστοποίηση κάποιας από αυτές μπορεί να απαιτεί τιμές που απομακρύνονται από τα βέλτιστα των υπολοίπων. Ορισμένες φορές τέτοια προβλήματα αναδιατυπώνονται σε προβλήματα απλής παραγοντικής βελτιστοποίησης, όπου εξετάζεται μόνο μία αντικειμενική συνάρτηση ανάγοντας τις διάφορες αντικειμενικές συναρτήσεις σε μία, χρησιμοποιώντας συντελεστές βάρους ή διαφορετικά αντικαθιστώντας μερικές από τις αντικειμενικές συναρτήσεις με συναρτήσεις περιορισμών.

Τα προβλήματα βελτιστοποίησης διαχωρίζονται σε τρεις κύριες κατηγορίες [24]:

Προβλήματα συνεχούς βελτιστοποίησης, στα οποία όλοι οι παράγοντες μπορούν να λάβουν πραγματικές τιμές στα αντίστοιχα πεδία ορισμού τους και είτε να διαθέτουν περιορισμούς (constraints), είτε όχι. Στην περίπτωση προβλημάτων βελτιστοποίησης χωρίς συναρτήσεις περιορισμού (unconstrained optimization), το ζητούμενο είναι απλά να βρεθούν οι τιμές για τα ακρότατα της συνάρτησης, είτε ελάχιστα, είτε μέγιστα. Ενώ στην δεύτερη περίπτωση που υπάρχουν περιορισμοί (constrained optimization) συναντώνται οι εξής περιπτώσεις:

i. Μη-γραμμική βελτιστοποίηση με περιορισμούς, όπου το ζητούμενο είναι η εύρεση του ελαχίστου μιας μη-γραμμικής συνάρτησης με μη-γραμμικούς περιορισμούς.

ii. Bound-contrained optimization, οι παράμετροι του προβλήματος εκφράζουν φυσικά μεγέθη και δεν είναι εφικτό να λάβουν τιμές εκτός αυτών των ορίων.

iii. Quadratic programming (τετραγωνικός προγραμματισμός), το πρόβλημα περιλαμβάνει την ελαχιστοποίηση κάποιας τετραγωνικής συνάρτησης που υπόκειται σε γραμμικούς περιορισμούς.

iv. Γραμμικός προγραμματισμός, ο στόχος είναι στην ελαχιστοποίηση μιας γραμμικής παραγοντικής συνάρτησης με μεταβλητές που ανήκουν στο σύνολο των πραγματικών αριθμών και χαρακτηρίζεται από γραμμικές συναρτήσεις περιορισμών.

v. (semidefinite programming), μια εκδοχή γραμμικού προγραμματισμού, όπου οι περιορισμοί δεν έχουν απόλυτο ορισμό, αλλά διαθέτουν έναν βαθμό αοριστίας.

vi. Στοχαστικός προγραμματισμός, για την επίλυση του προβλήματος γίνεται εκτεταμένη χρήση τυχαίων τιμών και σύγκριση των αποτελεσμάτων αυτών των τυχαίον τιμών μεταξύ τους. Χρησιμοποιούνται συναρτήσεις περιορισμών και σε αυτή την σειρά μεθόδων. Μερικά παραδείγματα από μεθόδους της κατηγορίας είναι οι Gradient Descent, Random Walk, Simulated Annealing, Monte Carlo κ.α.

vii. Network programming, γίνεται χρήση δικτύων (όπως είναι γνωστά από την θεωρία γραφημάτων) και μπορούν να είναι γραμμικά ή μη-γραμμικά προβλήματα.

Η επόμενη κατηγορία προβλημάτων βελτιστοποίησης είναι προβλήματα που οι παράμετροι λαμβάνουν διακριτές τιμές και όχι συνεχείς σε αντίθεση με την πρώτη περίπτωση. Σε αυτή την περίπτωση προβλημάτων βελτιστοποίησης συναντώνται δύο κατηγορίες.

i. integer programming, περιπτώσεις που έχουν νόημα μόνο αν ορισμένες από τις παραμέτρους του προβλήματος μπορούν να λάβουν μόνο ακέραιες τιμές. Ένα τέτοια παράδειγμα είναι ο κλασικός αλγόριθμος του “Πλανόδιου Πωλητή”.

ii. Στοχαστικός προγραμματισμός, συνήθως θεωρείται ότι τα δεδομένα του προβλήματος είναι γνωστά με μεγάλη ακρίβεια. Σε ορισμένες περιπτώσεις αυτό δεν συμβαίνει για διάφορους λόγους, όπως το σφάλμα μετρήσεων, ή δεδομένα που προέρχονται από μεθόδους πρόβλεψης και υπάρχει απόκλιση από τις πραγματικές τιμές, δηλαδή παρουσιάζουν κάποιο τυχαίο σφάλμα. Η λογική των στοχαστικών αλγορίθμων είναι να διαθέτουν μηχανισμούς αυτο-διόρθωσης που παρατηρώντας τις μεταβολές στην απόκλιση του αποτελέσματος, σταδιακά συγκλίνουν προς την ρίζα του προβλήματος.

Η τελευταία περίπτωση των μεθόδων βελτιστοποίησης είναι τα προβλήματα πολυπαραγοντικής βελτιστοποίησης (multi-objective optimization). Σε αυτή την κατηγορία το πρόβλημα χαρακτηρίζεται από πολλές αντικειμενικές συναρτήσεις, σε αντίθεση με την μοναδική αντικειμενική συνάρτηση που διαθέτουν τα απλά προβλήματα βελτιστοποίησης. Πιο συγκεκριμένα στα προβλήματα πολυπαραγοντικής βελτιστοποίησης οι αντικειμενικές συναρτήσεις συνθέτουν έναν πολυδιάστατο χώρο πέρα από τον απλό χώρο των μεταβλητών απόφασης (*decision variable space*), ο οποίος χαρακτηρίζεται από διανύσματα που ικανοποιούν τους περιορισμούς του προβλήματος. [25]

$$
\begin{aligned}
&f_m(\boldsymbol{x}), \quad m=1,2,\ldots,M \\
&g_j(\boldsymbol{x})\geq 0, \quad j=1,2,\ldots,J \\
&h_k(\boldsymbol{x})=0, \quad k=1,2,\ldots,K \\
&x_i^{(L)}\leq x_i\leq x_i^{(U)}, \quad i=1,2,\ldots,n
\end{aligned}
\tag{3.1}
$$

f_m: αντικειμενική συνάρτηση
g_j, h_k, x_i: περιορισμοί
$\boldsymbol{x}\in\Re^n \rightarrow \boldsymbol{x}=(x_1,x_2,\ldots,x_n)^T$
x: decision variables

3.2 Μέθοδος Gradient Descent

Η μέθοδος Gradient Descent χρησιμοποιείται κατά το στάδιο της βελτιστοποίησης σε αλγορίθμους, όπως το XGBoost στο οποίο έγινε αναφορά στο προηγούμενο κεφάλαιο, αλλά και σε αλγορίθμους όπως το FastTree, στο οποίο θα γίνει αναφορά στο επόμενο κεφάλαιο. Για αυτό το λόγο κρίνεται απαραίτητο να γίνει μία σύντομη αναφορά στον εν λόγω αλγόριθμο.

Ο αλγόριθμος Gradient Descent ή αλλιώς Steepest Descent είναι μία μέθοδος βελτιστοποίησης κατά την οποία εντοπίζεται το τοπικό ελάχιστο μιας πολυδιάστατης συνάρτησης

$$f(x_1,x_2,\ldots,x_n) \tag{3.2}$$

Προϋποθέτει ότι η συνάρτηση είναι παραγωγίσιμη και η παράγωγος μπορεί να υπολογιστεί, αν και αυτό δεν είναι εύκολο σε όλες τις περιπτώσεις.

Για την εύρεση του τοπικού ελαχίστου ορίζεται από τον χρήστη ένα αρχικό σημείο από το οποίο ο αλγόριθμος αρχίζει να ψάχνει. Γίνεται δηλαδή μία υπόθεση για την περιοχή κοντά στην οποία βρίσκεται το τοπικό ελάχιστο. Είναι επαναληπτική μέθοδος και σε κάθε επανάληψη υπολογίζεται η παράγωγος μέχρι η τιμή της να γίνει μικρότερη από ένα προκαθορισμένο σφάλμα. Στην αντίθετη περίπτωση που η μέθοδος προσπαθεί να εντοπίσει το τοπικό μέγιστο, ονομάζεται Gradient Ascent. Βασίζεται στην παραδοχή ότι αν μία πραγματική συνάρτηση G(x) είναι ορισμένη και παραγωγίσιμη γύρω από ένα σημείο x_a τότε ο ρυθμός που μειώνεται η G(x) είναι μεγαλύτερος προς την κατεύθυνση που η παράγωγος είναι αρνητική.

$$
\begin{aligned}
&\text{Αν } \boldsymbol{x_b}=\boldsymbol{x_a}-a\nabla G(\boldsymbol{x_a}) \\
&\forall a>0 \rightarrow G(\boldsymbol{x_a})\geq G(\boldsymbol{x_b})
\end{aligned}
\tag{3.3}
$$

οπότε μετά την υπόθεση του αρχικού σημείου, τα επακόλουθα σημεία μπορούν να γραφούν ως:

$$\boldsymbol{x_{n+1}}=\boldsymbol{x_n}-a_n\nabla G(\boldsymbol{x_n}), \quad n\geq 0 \tag{3.4}$$

οπότε συνεπάγεται ότι

$$G(\boldsymbol{x_0}) \geq G(\boldsymbol{x_1}) \geq G(\boldsymbol{x_2}) \geq \ldots \geq G(\boldsymbol{x_n}) \tag{3.5}$$

και η σειρά σταδιακά συγκλίνει στο τοπικό ελάχιστο. Το βήμα ***a*** της μεταβολής μπορεί να προσαρμοστεί σε κάθε επανάληψη ανάλογα με το πόσο απότομη είναι η μεταβολή από την προηγούμενη επανάληψη.

3.3 Πολυπαραγοντική Βελτιστοποίηση (Multiobjective Optimization)

Ένα πρόβλημα πολυπαραγοντικής βελτιστοποίησης αποτελείται έναν πεπερασμένο αριθμό αντικειμενικών συναρτήσεων. Έστω ότι όλες οι αντικειμενικές συναρτήσεις έχουν την ίδια σημαντικότητα *n* και πρέπει να ελαχιστοποιηθούν ή να μεγιστοποιηθούν βάσει κάποιου κριτηρίου απόδοσης. Μαθηματικά μπορεί να διατυπωθεί ως

$$F(x) = min[f_1(x), f_2(x), f_3(x), \ldots, f_m(x) \quad \forall x \in \boldsymbol{X}] \tag{3.6}$$

$$\boldsymbol{X} = [x_1, x_2, x_3, \ldots, x_n]^T$$
$\boldsymbol{X}$: σύνολο από n decision vectors, (decision space) (3.7)

$$x_i^{low} \leq x_i \leq x_i^{high} \qquad i = 1,2,3,\ldots,n \tag{3.8}$$

Για την ορθή αξιολόγηση της σημαντικότητας κάθε παράγοντα πρέπει να υπάρχει πλήρης κατανόηση του προβλήματος. Εφόσον το F(x) είναι διάνυσμα οι αντικρουόμενες αντικειμενικές συναρτήσεις, δηλαδή εκείνες οι συναρτήσεις που η βελτιστοποίηση της μίας απομακρύνει το σημείο από την βελτιστοποίηση της άλλης, δεν είναι εφικτό να υπολογίσουν κάποιο συγκεκριμένο σημείο ως το βέλτιστο του συστήματος, αλλά περιοχές βέλτιστων ή επικρατέστερων σημείων, όπως ονομάζονται.

$$f(x) = \boldsymbol{Y} = [y_1, y_2, y_3, \ldots, y_m]^T$$
$\boldsymbol{X}$: χώρος των λύσεων του συστήματος
x: μία από τις λύσεις του συστήματος, $x \in \boldsymbol{X}$ (3.9)
$\boldsymbol{Y}$: ένα σημείο στον χώρο του συστήματος

Από το παραπάνω προκύπτει ότι το F(x) = **x** είναι μία από τις λύσεις της βελτιστοποίησης, ενώ το f(x) = **y** είναι ένα οποιοδήποτε σημείο στο πολυδιάστατο χώρο που ορίζει το πρόβλημα.

3.3.1 Pareto Optimality

Ο όρος Pareto Optimality έχει καθιερωθεί για τα προβλήματα πολυπαραγοντικής βελτιστοποίησης που περιλαμβάνουν την βελτιστοποίηση διανυσματικών παραγόντων, με τους παράγοντες να αντικρούονται μεταξύ τους και η βελτιστοποίηση ενός παράγοντα να συνεπάγεται την επιβάρυνση άλλων παραγόντων, με αποτέλεσμα να πρέπει να γίνουν συμβιβασμοί που να ικανοποιούν όλους τους παράγοντες, αλλά σε μικρότερο βαθμό. Κάθε σημείο του αντικειμενικού χώρου **Y** αντιστοιχεί σε ένα μοναδικό σύνολο μεταβλητών του μοντέλου με τον τρόπο που τα κατηγοριοποιεί η Καταλληλότητα

Pareto (Pareto Optimality). Σε μια ιδανική περίπτωση οι βέλτιστες λύσεις ενός πολυπαραγοντικού προβλήματος ονομάζονται Σύνολο Pareto και παριστάνονται ως:

$$\boldsymbol{X}' \subseteq \boldsymbol{X} \tag{3.10}$$

Ενώ το αντικειμενικό διάνυσμα (objective vector) ως:

$$\boldsymbol{Y}' + f(\boldsymbol{X}') \subseteq \boldsymbol{Y} \tag{3.11}$$

Φυσικά στην πραγματικότητα δεν υπάρχει το "βέλτιστο" σημείο ως μοναδική καλύτερη λύση, αλλά ένα σύνολο από βέλτιστες ρίζες που προέρχονται από συμβιβασμούς κατά την βελτιστοποίηση των επιμέρους παραγόντων. Αυτό εκφράζουν και οι βέλτιστε ρίζες στο σύνολο Pareto, ότι δεν μπορεί να βελτιστοποιηθεί κάποιος από τους παράγοντες περαιτέρω χωρίς να ζημιωθεί κάποιος άλλος. Με αυτή την αρχή σαν βάση πρέπει να γίνει ξεκάθαρο ότι από αυτούς τους αλγορίθμους ψάχνουμε να βρούμε τον καλύτερο συμβιβασμό και όχι την "καλύτερη" λύση.

3.3.2 Σχέση Επικράτειας (Dominance Relationship)

Αφού δεν υπάρχει απόλυτο κριτήριο για την ξεκάθαρη ανωτερότητα κάποιων ριζών από τις υπόλοιπες, προκύπτει το ερώτημα πως θα συγκρίνουμε τις ρίζες μεταξύ τους, αφού όλη η λογική αυτών τον αλγορίθμων βασίζεται στην σύγκριση.

Για την σύγκριση των ριζών χρησιμοποιείται από την βιβλιογραφία ο όρος επικράτεια (dominance) και λειτουργεί κάνοντας μία σειρά από απλές συγκρίσεις [26].

1. Έστω ότι η ρίζα x_1 είναι καλύτερη από την ρίζα x_2

$$f_i(x_1) \leq f_i(x_2) \quad \forall i \in 1,2,3,\ldots,m \tag{3.12}$$

2. Έστω για τουλάχιστον έναν παράγοντα η ρίζα x_1 είναι με διαφορά καλύτερα από την ρίζα x_2

$$f_j(x_1) < f_j(x_2) \quad \exists j \in 1,2,3,\ldots,m \tag{3.13}$$

Αν δεν ισχύει καμία από τις δύο περιπτώσεις τότε η ρίζα x_1 δεν μπορεί να θεωρηθεί επικρατέστερη της ρίζας x_2 και αντίστροφα η x_2 δεν μπορεί να θεωρηθεί επικρατέστερη της x_1.

$$\text{Αν} \quad x_1 \nprec x_2 \quad \neq \Rightarrow x_2 \nprec x_1 \tag{3.14}$$

Ισχύει όμως η περίπτωση αν η ρίζα x_1 είναι επικρατέστερη της ρίζας x_2 και η ρίζα x_2 είναι επικρατέστερη της ρίζας x_3, τότε η ρίζα x_1 είναι επικρατέστερη της ρίζας x_3.

$$\begin{aligned} &\text{Αν} \quad x_1 \preccurlyeq x_2 \quad \text{και} \quad x_2 \preccurlyeq x_3 \\ &\text{τότε} \quad x_1 \preccurlyeq x_3 \end{aligned} \tag{3.15}$$

Αυτή η ιδιότητα είναι πολύ σημαντική επειδή επιτρέπει να αποφανθούμε ποιες ρίζες (X’) δεν είναι υπολειπόμενες στις ρίζες του συνόλου (X). Οι μη-υπολειπόμενες ρίζες (X’) ολόκληρου του χώρου **Y** ονομάζονται ολικές βέλτιστες ρίζες Pareto (globally Pareto-optimal sets). Είναι κατανοητό φυσικά ότι για να υπολογισθούν όλα τα **y** του αντικειμενικού χώρου **Y**, είναι πολύ χρονοβόρο και ειδικά για προβλήματα με πάρα πολλού παράγοντες γίνεται πρακτικά αδύνατο. Οπότε η σχεδίαση εξυπνότερων και κατ’ επέκταση ταχύτερων αλγορίθμων είναι αναγκαία. Στον κλάδο τον εξελικτικών αλγορίθμων έχει γίνει πολλή έρευνα σε αυτό το κομμάτι και έχουν αναπτυχθεί μέθοδοι που μπορούν να κάνουν πολύ καλή προσέγγιση του συνόλου Pareto-optimal χωρίς να υπολογίσουν ολόκληρο τον αντικειμενικό χώρο **Y**.

3.3.3 Διαχείριση Περιορισμών (Constraint Handling)

Σε προηγούμενη παράγραφο έγινε αναφορά στα προβλήματα που οι παράγοντες τους λαμβάνουν πραγματικές τιμές και είναι συνεχείς μεταβλητές, αλλά πολλοί περιορισμοί που χαρακτηρίζουν το ίδιο το πρόβλημα καθιστούν αδύνατο οι παράγοντες να λάβουν τιμές σε ορισμένα διαστήματα. Στην περίπτωση όμως που οι αλγόριθμοι είναι στοχαστικοί και λειτουργούν με τυχαίους αριθμός, θα θέσουν στα **y** ακατάλληλες τιμές που δεν επιτρέπεται να λάβουν. Για να αποφευχθούν τέτοια περιστατικά, σαν μηχανισμός επαλήθευσης εφαρμόζονται κάποιες συνήθεις τεχνικές. [27]

i. Απόρριψη αδύνατων σημείων **y**.

ii. μείωση της τιμής του fitness των αδύνατων σημείων μέσω μιας συνάρτησης ποινής.

iii. σχεδιασμός “γενετικών” τελεστών που να δίνουν σαν αποτέλεσμα μόνο εφικτά σημεία.

iv. Μετασχηματισμός των αδύνατων σημείων σε εφικτά, με “διόρθωση” (repair).

Οι περιπτώσεις i, iii, iv εξαρτώνται από το ίδιο το πρόβλημα και θα πρέπει ο χρήστης να τις παρέχει σε αυτό. Ενώ για την περίπτωση ii, το πρόβλημα είναι ότι ενώ στα απλά προβλήματα βελτιστοποίησης είναι πολύ συνηθισμένη η εφαρμογή της, στα πολυπαραγοντικά προβλήματα κάποιες φορές δεν είναι εύκολος ο προσδιορισμός της συνάρτησης fitness, οπότε αντίστοιχα γίνεται δύσκολη και η χρήση συνάρτησης ποινής.

3.3.4 Έλεγχος Απόδοσης (Performance Measure)

Για την αξιολόγηση των χρωμοσωμάτων – ριζών του πληθυσμού συνήθως εφαρμόζονται τα παρακάτω τρία κριτήρια [26].

1. Σύγκλιση (*Convergence*) (γ):

   Εκτιμά την απόσταση των υποψήφιων υπολειπόμενων (Pareto) ριζών από την περιοχή των γνωστών βέλτιστων ριζών. Για κάθε νέα ρίζα που υπολογίζεται από τον αλγόριθμο που έχει επιλεγεί, μετριέται η (Ευκλείδεια) απόσταση μεταξύ της νέας ρίζας με τα βέλτιστα της περιοχής Pareto και η μέση απόσταση χρησιμοποιείται σαν μέτρο σύγκλισης. Όσο μικρότερη η τιμή του *γ*, τόσο μεγαλύτερη η σύγκλιση.

2. Διασπορά (Diversity) (Δ):

   Εφόσον ο αλγόριθμος ψάχνει τις επικρατέστερες ρίζες σε ολόκληρο τον χώρο των Pareto επικρατέστερων, είναι σημαντικό να είναι γνωστή η διασπορά των σημείων, επειδή θα παίξει ρόλο στην επαλήθευση της ακρίβειας του αλγορίθμου και την επιλογή της τελικής ρίζας για την επίλυση του προβλήματος.

3. Displacement (D):

   Ενδεχομένως να έχει βρεθεί μόνο ένα ποσοστό των Pareto βέλτιστων τιμών από το πραγματικό τους σύνολο, αν θεωρηθεί ότι ο αλγόριθμος δεν θα έχει τον χρόνο να ψάξει ολόκληρο το **Y**. Η μέθοδος του displacement λύνει αυτό το πρόβλημα, καθώς μετράει την σχετική εμβέλεια ενός πιθανού συνόλου ριζών από μία γνωστή περιοχή βέλτιστων Pareto ριζών, όσο μικρότερη είναι η τιμή τόσο πιο κοντά βρίσκεται στην τελική λύση, και ακολουθεί την μαθηματική διατύπωση:

$$D=\frac{1}{|P'|}\cdot\sum_{i=1}^{|P'|} min_{j=1}^{|Q|}[d(i,j)] \tag{3.16}$$

P': ομοιόμορφα κατανεμημένες ρίζες απο την πραγματική περιοχή των επικρατέστερων Pareto ριζών
Q: τελική ρίζα του προβλήματος
$d(I,j)$: απόσταση μεταξύ της i ρίζας του P' και της j ρίζας του Q

Ο κάθε αλγόριθμος μπορεί να θέσει διαφορετικά κριτήρια για τον έλεγχο της απόδοσης του ανάλογα με τον τρόπο που λειτουργεί. Σε γενικές γραμμές όμως, εξαρτάται από την σύγκριση των ριζών με τις επικρατέστερες, όπως ορίζονται από τον Pareto.

3.4 Γενετικοί Αλγόριθμοι

Στο προηγούμενο κεφάλαιο στην παράγραφο των Νευρωνικών Δικτύων είδαμε ότι υπάρχουν περιπτώσεις που για την ανάπτυξη νέων αλγορίθμων και μοντέλων οι ερευνητές του κλάδου, είχαν επιρροές από άλλα επιστημονικά πεδία και πιο συγκεκριμένα της βιολογίας. Αντίστοιχη περίπτωση είναι και ο κλάδος της πολυπαραγοντικής βελτιστοποίησης που οι ερευνητές δανείζονται επιρροές πάλι από την βιολογία, για να αναπτύξουν μία σειρά αλγορίθμων που ονομάζεται Εξελικτικοί Αλγόριθμοι. Οι Εξελικτική αλγόριθμοι είναι πολύ αποδοτικοί σε τέτοια προβλήματα. Μιμούμενοι την θεωρία της εξέλιξης και εφαρμόζουν δοκιμασμένες τεχνικές που από την φύση τους είναι πολυπαραγωντικής φύσης.

Υποκατηγορία των Εξελικτικών Αλγορίθμων είναι οι Γενετικοί Αλγόριθμοι στους οποίους θα δοθεί έμφαση, λόγω του ότι αποτελεί το τελευταίο σκέλος της εφαρμογής. Περισσότερα για αυτό στο επόμενο κεφάλαιο. Σαν σύντομη εισαγωγή στους γενετικούς αλγορίθμους δίνονται ορισμένα πλεονεκτήματα και μειονεκτήματα που τους χαρακτηρίζουν [28]. Για τις επόμενες παραγράφους

επισημαίνεται ότι στην περίπτωση των γενετικών αλγορίθμων ο όρος “χρωμόσωμα” και “ρίζα” είναι έννοιες ταυτόσημες.

Πλεονεκτήματα των γενετικών αλγορίθμων:

- Δεν απαιτεί την παράγωγο για να τρέξει όπως το Gradient Ascent, η οποία πολλές φορές δεν θα είναι διαθέσιμη σε πολλά προβλήματα.
- Είναι γρηγορότεροι και πιο αποδοτικοί από άλλους παραδοσιακούς αλγορίθμους.
- Είναι σχετικά εύκολο να τρέξουν παράλληλα (multi-thread).
- Μπορούν να χρησιμοποιηθούν σε προβλήματα με παράγοντες που έχουν συνεχείς αλλά και διακριτές τιμές.
- Παρέχουν σύνολα με βέλτιστες ρίζες (dominant solutions) και όχι μοναδικές ρίζες (unique solutions) και πάντα θα παρέχουν λύση στο πρόβλημα, η οποία θα βελτιώνεται σε κάθε γενιά του πληθυσμού. Δεν κινδυνεύει να εγκλωβιστεί σε κάποιο τοπικό ακρότατο.
- Πολύ εύχρηστοι σε προβλήματα με πολλούς παράγοντες και με μεγάλα πεδία ορισμού.

Μειονεκτήματα των Γενετικών Αλγορίθμων:

- Οι Γενετικοί αλγόριθμοι δεν προτείνονται για απλούστερα προβλήματα, ειδικά σε εκείνα που η παράγωγος μπορεί να υπολογιστεί εύκολα. Ο Γενετικός αλγόριθμος θα επιβάλλει άσκοπη τυχαιότητα σε εκείνες τις περιπτώσεις.
- Η συνάρτηση *Fitness* υπολογίζεται πολλές φορές και για αυτό το λόγο πρέπει να είναι υπολογιστικά φθηνή, ώστε να μην αποτελέσει το bottleneck. Σε μερικά προβλήματα αυτό μπορεί να είναι δύσκολο έως αδύνατο.
- Λόγω της στοχαστικής φύσης τους δεν μπορεί να είναι βέβαιη η ποιότητα του αποτελέσματος, δηλαδή ότι δεν θα περιοριστεί σε κάποια περιοχή βέλτιστων αποτελεσμάτων Pareto, αγνοώντας τις υπόλοιπες εν δυνάμει επικρατούσες περιοχές. Αυτό εξαρτάται και από το ίδιο το πρόβλημα.
- Εξαρτώνται σε τεράστιο βαθμό από τον ορισμό της συνάρτησης *Fitness* που ενδέχεται να μην είναι πάντα εύκολο να ορισθεί.
- Σε περίπτωση που δεν σχεδιαστούν σωστά, ίσως να μην συγκλίνουν.

3.4.1 Τυπική Δομή Γενετικού Αλγορίθμου

Η δομή ενός τυπικού γενετικού αλγορίθμου μπορεί να χαρακτηριστεί από τα ακόλουθα βήματα [27].

1. Ορισμός του αρχικού τυχαίου πληθυσμού.
2. Υπολογισμός της τιμής fitness από την αντίστοιχη συνάρτηση που έχει ορισθεί για το πρόβλημα για κάθε χρωμόσωμα του πληθυσμού.

3. Διασταύρωση (Crossover), γίνεται διασταύρωση μεταξύ ζευγών χρωμοσωμάτων που έχουν επιλεχθεί από κάποια συνάρτηση επιλογής, (υπάρχουν πολλές εναλλακτικές για μία τέτοια συνάρτηση) και προσωρινή αποθήκευση των απογόνων (offsprings) στην μνήμη σαν πληθυσμός Q_t.
4. Μετάλλαξη (Mutation), βάσει ενός τυχαίου συντελεστή μετάλλαξης (συνήθως <0.2) μπορεί να συμβεί κάποια μετάλλαξη σε κάθε χρωμόσωμα και προκαλεί αλλαγές σε αυτό το χρωμόσωμα του πληθυσμού. Λειτουργεί σαν μηχανισμός αποτροπής του πρόωρου τερματισμού του αλγορίθμου και διατήρησης της ποικιλομορφίας στον πληθυσμό. Επιπλέον υπάρχει κάποιος μηχανισμός (Constraint Handling) που ελέγχει την μετάλλαξη να μην προκαλέσει τιμές εκτός του πεδίου ορισμού του προβλήματος.
5. Υπολογισμός fitness για κάθε χρωμόσωμα του νέου πληθυσμού βάσει της τιμής της αντικειμενικής συνάρτησης.
6. Φυσική Επιλογή, επιλέγονται N λύσεις από τον πληθυσμό των απογόνων Q_t σύμφωνα με τις τιμές fitness και αντιγράφονται στον πληθυσμό P(t+1), όπου είναι η επόμενη γενιά του πληθυσμού.
7. Λήξη της εκτέλεσης του αλγορίθμου αν εκπληρώνεται κάποιο από τα κριτήρια λήξης.
   - Ο πληθυσμός έχει συγκλίνει στην λύση και το fitness δεν έχει βελτιωθεί στις τελευταίες γενιές.
   - Έχει επέλθει η μέγιστη προκαθορισμένη γενιά, δηλαδή έχει εκτελέσει τον προκαθορισμένο αριθμό μέγιστο επαναλήψεων.
   - Σε διαφορετική περίπτωση συνεχίζει την εκτέλεση του από το βήμα 3.

3.4.2 Συναρτήσεις Fitness

Υπάρχουν πολλοί διαφορετικοί τρόποι για να τον ορισμό του Fitness, αναλυτική περιγραφή των οποίων ξεφεύγει από τον σκοπό της εργασίας, ο αναγνώστης μπορεί να βρει περισσότερες πληροφορίες στην αντίστοιχη βιβλιογραφία [27] στην παράγραφο *5.1*. Ονομαστικά αναφέρονται ορισμένοι μέθοδοι.

- Weighted Sum Approach
- Altering Objective Functions
- Pareto Ranking Approaches

Στην υλοποίηση της βιβλιοθήκης του γενετικού αλγορίθμου χρησιμοποιήθηκε η πρώτη περίπτωση, οπότε σε επόμενο κεφάλαιο θα γίνει εκτενέστερη περιγραφή για αυτή. Η επιλογή μίας κατάλληλης συνάρτησης *Fitness*, εξαρτάται από το ίδιο το πρόβλημα και επομένως είναι απαραίτητα η καλή κατανόηση αυτού. Επίσης υπενθυμίζουμε ότι ο γενετικός αλγόριθμος είναι επαναληπτική μέθοδος και

η συνάρτηση *fitness* πρόκειται να εκτελεστεί πάρα πολλές φορές. Η ταχύτητα εκτέλεσης της είναι κρίσιμη, διότι μπορεί να αυξήσει σημαντικά την ταχύτητα ολοκλήρωσης του αλγορίθμου.

3.4.3 Fitness Sharing

Το fitness sharing είναι μία μέθοδος που προωθεί το ψάξιμο σε ανεξερεύνητες περιοχές των επικρατέστερων λύσεων, μειώνοντας τεχνητά τις τιμές του fitness σε πολύ πυκνές περιοχές λύσεων – ριζών. Είναι πιθανό ο αλγόριθμος να εγκλωβιστεί σε κάποια περιοχή οδηγώντας πρόωρα στον τερματισμό του και αγνοώντας άλλες περιοχές με εν δυνάμει επικρατέστερες ρίζες, όπως παρόμοια συμβαίνει και σε άλλους στοχαστικούς αλγορίθμους, πχ simulated annealing. Ο τρόπος λειτουργίας του είναι απλός. Ορίζεται ένας συντελεστή ποινής στις ρίζες εκείνων των περιοχών που είναι πολύ πυκνές και επισκιάζουν τις υπόλοιπες ανεξερεύνητες περιοχές. Τα βήματα εφαρμογής του fitness sharing είναι τα παρακάτω [27]:

1. Υπολογισμός της Ευκλείδειας απόστασης μεταξύ κάθε ζεύγους σημείων **x** και **y** στο κανονικοποιημένο παραγοντικό χώρο στο διάστημα [0, 1].

$$dz(x,y)=\sqrt{\sum_{k=1}^{K}\left(\frac{z_k(\boldsymbol{x})-z_k(\boldsymbol{y})}{z_k^{max}-z_k^{min}}\right)^2} \tag{3.17}$$

z_k^{max}, z_k^{min}: η μέγιστη και ελάχιστη τιμής της αντικειμενικής συνάρτησης z_k από όσα σημεία έχουν ερευνηθεί μέχρι στιγμής

2. Πάνω σε αυτές τις αποστάσεις υπολογίζεται ένας συντελεστής σπανιότητας (niche count) για κάθε ρίζα.

$$\forall \quad \boldsymbol{x}\in P \quad nc(\boldsymbol{x},t)=\sum_{y\in P, r(\boldsymbol{y},t)=r(\boldsymbol{x},t)} max\left|\frac{\sigma_{share}-dz(\boldsymbol{x},\boldsymbol{y})}{\sigma_{share}},0\right| \tag{3.18}$$

P: πλυθυσμός
$nc(\boldsymbol{x},t)$: niche count, συντελεστής σπανιότητας
σ_{share}: niche size

3. Μετά τον υπολογισμό των niche counts, η τιμή fitness κάθε χρωμοσώματος προσαρμόζεται ως εξής:

$$f'(\boldsymbol{x},t)=\frac{f(\boldsymbol{x},t)}{nc(\boldsymbol{x},t)} \tag{3.19}$$

Το σ_{share} ορίζει μία περιοχή χρωμοσωμάτων στο **Y** . Ένα χρωμόσωμα – ρίζα σε μία πυκνή περιοχή θα έχει υψηλότερο niche count από κάποια άλλη ρίζα σε αραιή περιοχή. Με την μέθοδο fitness sharing, ελαττώνεται η πιθανότητα επιλογής χρωμοσωμάτων σε κάποιο σημείο του **Y** για το στάδιο του Crossover και προτρέπεται η περαιτέρω αύξηση του πληθυσμού σε εκείνες τις ήδη πυκνές περιοχές.

4. Εφαρμογή

4.1 Σκοπός της Εργασίας

Ο σκοπός της εργασίας ήταν η διερεύνηση της βελτιστοποίησης της διεργασίας της καταβύθισης βαιμίτη από υπέρκορα αργιλικά διαλύματα μέσω της Μηχανικής Μάθησης για την παραγωγή του μοντέλου που εκφράζει την κινητική της καταβύθισης με επαρκή ακρίβεια, ως υποκατάστατο της συμβατικής αναλυτικής μεθόδου παλινδρόμησης. Το επόμενο βήμα ήταν η βελτιστοποίηση των αρχικών παραμέτρων του πειράματος, ώστε να επιτευχθεί η μεγιστοποίηση της απόδοσης της αντίδρασης, με παράλληλη ελαχιστοποίηση της θερμοκρασίας και του χρόνου αντίδρασης. Δηλαδή η συλλογή δεδομένων και αποθήκευση τους στον δίσκο, είτε ως απλό .csv, είτε ως βάση δεδομένων (SQLite, Redis), η σύγκριση Αναλυτικής Μεθόδου με την Μηχανική Μάθηση, και η βελτιστοποίηση των αρχικών παραμέτρων του πειράματος. Για τα παραπάνω σχεδιάστηκε αυτοσχέδιο πρόγραμμα τύπου CLI(command-line interface), ως βοηθητικό εργαλείο για την παραγωγή των δεδομένων, την γραφική απεικόνιση τους και την βελτιστοποίηση του πειράματος, δηλαδή όλες τις λειτουργίες που ήταν απαραίτητες για τα ακόλουθα βήματα του πρακτικού μέρους της εργασίας, όπως αναλύονται στις παραγράφους που ακολουθούν.

Επιγραμματικά τα τρία στάδια του πρακτικού μέρους της εργασίας μπορούν να αποτυπωθούν ως:

1. Συλλογή Δεδομένων
   - Αποθήκευση Δεδομένων
2. Σχεδιασμός μοντέλου
   - Σύγκριση κλασικής αναλυτικής μεθόδου –- με μέθοδο μηχανικής μάθησης
3. Βελτιστοποίηση του πειράματος
   - Πολυπαραγοντική Βελτιστοποίηση

4.2 Υπολογιστική Βιβλιοθήκη

Η υλοποίηση πειραμάτων είναι μία χρονοβόρα και κοστοβόρα διαδικασία, καθώς για την μοντελοποίηση του προβλήματος απαιτείται μεγάλος αριθμός πειραμάτων. Το πείραμα είχε διεξαχθεί το 2003 και τα πειραματικά αποτελέσματα δεν ήταν αρκετά σε αριθμό για να καλύψουν την εκπαίδευση ενός μοντέλου μηχανικής μάθησης, ήταν όμως επαρκή για την σχεδίαση ενός αναλυτικού μοντέλου. Στο πρώτο κεφάλαιο διατυπώθηκε το μοντέλο της κινητικής καταβύθισης βαιμίτη σε διάλυμα καυστικού ασβεστίου.

$$Cin_{Al_2O_3} = Cin_{Na_2O} \cdot MR \tag{4.1}$$

$$C_{eq}=A_1\cdot 10^{-6}\cdot T^3+A_2\cdot 10^{-3}\cdot T^2+A_3\cdot 10^{-2}\cdot T+A_4$$

$T\in[30,150]\,°C$ και $C_{inNa2O}\in[60-140]$ g/L
C_{eq}: διαλυτότητα του βαιμίτη σε g/L
T: θερμοκρασία, °C
C_{Na_2O}: αρχική συγκέντρωση καυστικού νατρίου, g/L Na_2O (4.2)

$$A_1=-0.0618925\cdot C_{Na2O}+1.36953$$
$$A_2=0.02301\cdot C_{Na2O}+0.1707$$
$$A_3=2.498\cdot 10^{-6}\cdot C_{Na2O}^3-3.106\cdot 10^{-4}\cdot C_{Na2O}^2+5.483\cdot 10^{-2}\cdot C_{Na2O}-1.332$$
$$A_4=3.236\cdot 10^{-6}\cdot C_{Na2O}^3-7.887\cdot 10^{-4}\cdot C_{Na2O}^2+1.584\cdot 10^{-1}\cdot C_{Na2O}-2.518 \quad (4.3)$$

$$\frac{dC_{pr}}{dt}=2.30\cdot 10^{13}\cdot\left(C_{Na_2O}\right)^{-1.8}\cdot\left(SR\right)^{0.54}\cdot e^{-\frac{10750}{T}}\cdot\left(C-C_e^{app}\right)^2 \quad (4.4)$$

Από τις παραπάνω συναρτήσεις προκύπτουν τα τρία αναλυτικά μοντέλα στα οποία βασίστηκε η παραγωγή των δεδομένων, αλλά και η οπτική απεικόνιση όπως εξηγείται παρακάτω.

$$C_{Al_2O_3}(T,t)=\left[\frac{dC_{pr}}{dt}\cdot t+\frac{1}{Cin_{Al2O3}-1.9248\cdot SR^{-0.136}\cdot Ce}\right]^{-1}+1.9248\cdot SR^{-0.136}\cdot C_{eq} \quad (4.5)$$

$$Efficiency\left(C_{Al_2O_3}\right)=\frac{Cin_{Al_2O_3}-Ct_{Al_2O_3}}{Cin_{Al_2O_3}-C_{eq}}\cdot 100\,\% \quad (4.6)$$

$$LocalEfficiency\left(C_{n,Al_2O_3},C_{n+1,Al_2O_3}\right)=\frac{C_{n,Al_2O_3}-C_{n+1,Al_2O_3}}{Cin_{Al_2O_3}-C_{eq}}\cdot 100\,\% \quad (4.7)$$

MR: mass ratio
SR: seed ratio
T: θερμοκρασία
t: χρόνος
$Cin_{Al_2O_3}$: αρχική συγκέντρωση Al_2O_3
Cin_{Na_2O}: αρχική συγκέντρωση Na_2O
Ce: συγκέντρωση ισορροπίας
$Efficiency$: η απόδοση του πειράματος
$LocalEfficiency$: η απόδοση του πειράματος σε χρόνο dt (4.8)

Με την χρήση του αναλυτικού μοντέλου είναι πλέον εφικτό να αντιμετωπιστεί η έλλειψη πειραμάτων, καθώς μπορούν να υπολογιστούν “εικονικά” πειραματικά δεδομένα για την εκπαίδευση των μοντέλων μηχανική μάθησης.

4.3 Παραγωγή Δεδομένων

Μετά το πέρας του υπολογισμού των δεδομένων και τους απαραίτητους μαθηματικούς μετασχηματισμούς ακολουθεί η αποθήκευση αυτών των δεδομένων στον δίσκο (.csv). Υπενθύμιση, αυτά τα δεδομένα είναι το υποκατάστατο πειραματικών μετρήσεων που θα είχαν διεξαχθεί σε ένα εργαστήριο υπό κανονικές συνθήκες. Παρατηρώντας τα αποτελέσματα από τους υπολογισμούς του Η/Υ γίνεται εμφανές, ότι σε αντίθεση με κανονικές εργαστηριακές μετρήσεις δεν υπάρχει θόρυβος από τα όργανα μέτρησης ή άλλους εξωτερικούς παράγοντες, υπάρχει όμως θόρυβος στα δεδομένα από τον τρόπο που λειτουργεί το hardware. Το συγκεκριμένο πρόβλημα είναι γνωστό στον κλάδο της υπολογιστικής μηχανικής και έχουν αναπτυχθεί διάφορες τεχνικές για την αντιμετώπισή του, οι οποίες ξεφεύγουν από τον σκοπό της συγκεκριμένης εργασίας. Περισσότερες πληροφορίες στο αντίστοιχο κεφάλαιο(3) του [29]

4.3.1 Φίλτρο

Κατά την παραγωγή των δεδομένων παρατηρήθηκε ότι προέκυπταν τεράστια αρχεία (6-12GB ανάλογα την επιλογή αποθήκευσης δεδομένων που είχε επιλεχθεί, ~78.500.000 γραμμές) και προκαλεί τα ακόλουθα προβλήματα:

1. Σπατάλη αποθηκευτικού χώρου στον δίσκο
2. Τεράστιοι χρόνοι εκμάθησης
3. Κίνδυνος για overfitting
4. Θόρυβος στα δεδομένα
5. Επιπλοκές σε ορισμένους αλγορίθμους που δεν είναι σχεδιασμένοι για τέτοιου μεγέθους datasets

Για την αντιμετώπιση του προβλήματος, έγινε χρήση φίλτρου. Το φίλτρο λειτουργεί με έναν πολύ απλό τρόπο, κρατάει στην μνήμη του τις τελευταίες τιμές των πειραμάτων που αποθηκεύτηκαν στο .csv και συγκρίνει κάθε νέο πείραμα με αυτές, αν η διαφορά στην απόδοση του νέου πειράματος συγκριτικά είναι αισθητή, τότε αποθηκεύεται το νέο πείραμα στο (.csv) και στην μνήμη του φίλτρου και αφαιρείται από το φίλτρο το παλαιότερο από (cached) πειράματα. Δηλαδή πρόκειται για ένα τυπικό queue [30]. Οι παράμετροι του φίλτρου μπορεί να καθοριστούν από τον χρήστη στο αρχείο *config.ini* του προγράμματος. Περισσότερα στο τελευταίο κεφάλαιο. Το αποτέλεσμα ήταν η μείωση του dataset σε μόλις (14-21MB, ~84.000 γραμμές), αρκετά μικρό για την πολύ γρήγορη εκπαίδευση των αλγορίθμων, αφού μπορεί να φορτωθεί ολόκληρο στην μνήμη του υπολογιστή, κάτι που διευκολύνει πολλούς αλγορίθμους. Το σημαντικότερο είναι πως δεν είχε καμία επίπτωση στην ακρίβεια των μοντέλων, όπως φαίνεται σε επόμενη παράγραφο. Ο λόγος για αυτό είναι ότι κατά την διάρκεια που εκτελούνται οι υπολογισμοί για τις τιμές των “εικονικών” πειραμάτων τα περισσότερα από αυτά είναι περιττά και αυτό γίνεται ξεκάθαρο στις εικόνες της επιφάνειας απόδοσης της αντίδρασης με την απεικόνιση των αποθηκευόμενων πειραμάτων (πράσινα στίγματα) με την χρήση φίλτρου και χωρίς. [Παράρτημα]

4.4 Οπτική Απεικόνιση

Μετά την παραγωγή των δεδομένων υπάρχουν στον δίσκο πέντε αρχεία *.csv* και ~84.000 γραμμές με την κάθε γραμμή να εκφράζει ένα “εικονικό” πείραμα. Για την αξιολόγηση των πειραματικών δεδομένων θα χρειαστεί να εφαρμοστεί οπτική απεικόνιση, Data Exploration, όπως ονομάζεται αυτός ο όρος στην βιβλιογραφία. Δεύτερο σκέλος του project, μετά την υπολογιστική βιβλιοθήκη είναι η υλοποίηση τρισδιάστατης γραφικής απεικόνισης σε πραγματικό χρόνο. Η συγκεκριμένη υλοποίηση έγινε με χρήση της βιβλιοθήκης Avalonia για το UI και της Skia για την χάραξη των γραφικών, η οποία με την σειρά της χρησιμοποιεί OpenGL σαν renderer. Τεχνικά η ορθότερη επιλογή θα ήταν μάλλον η απευθείας χρήση κάποιου API γραφικών όπως OpenGL ή Vulkan σε συνδυασμό με το ImGUI για το UI, καθώς με αυτό τον τρόπο θα είχαν αποφευχθεί ορισμένες αναγκαστικές κακές επιλογές λόγω περιορισμών, αλλά προσθέτοντας πολύ μεγαλύτερη περιπλοκότητα στο project. Σε γενικές γραμμές η γνώση ενός σύγχρονου API γραφικών όπως το Vulkan μπορεί να είναι ένα πολύτιμο εφόδιο στην μηχανική, εφόσον προσφέρει άμεση πρόσβαση στην GPU και μπορεί να εφαρμοστεί είτε για αποδοτικότερη απεικόνιση γραφικών σε πραγματικό χρόνο, είτε για την εκτέλεση αριθμητικών υπολογισμών μέσω Compute Shaders.

Είναι πρωτεύοντος σημασίας να υπάρχει κατανόηση του προβλήματος και της επίδρασης των παραμέτρων στο πείραμα. Με την αλληλεπίδραση που προσφέρει το πρόγραμμα αυτή η κατανόηση γίνεται πολύ ευκολότερη, καθώς παρέχει στον χρήστη τις δυνατότητες:

i. Ελέγχου αποτελεσματικότητας του φίλτρου και επιλογής πόσο εκτενές θα είναι το dataset που θα χρησιμοποιηθεί για την εκπαίδευση. Μικρότερο dataset συνεπάγεται διευκόλυνση στην εκπαίδευση του μοντέλου μηχανικής μάθησης και μείωση του κινδύνου overfitting.

ii. Καλύτερο προσδιορισμό των παραγόντων σε τιμές που φέρουν ικανοποιητικότερη απόδοση του πειράματος. Κάτι που εξυπηρετεί και την καλύτερη ερμηνεία του προβλήματος, όπως έχει αναφερθεί στο Κεφάλαιο 2, στην παράγραφο για την βελτίωση της ακρίβειας του μοντέλου.

iii. Ευκολότερο προσδιορισμό της θεωρητικά μέγιστης απόδοσης του πειράματος, σύγκριση με τις λιγότερο υψηλές αποδόσεις, δηλαδή σαν βοηθητικό εργαλείο για την πολυπαραγοντική βελτιστοποίηση. Με την επιλογή “ClipToLocal” απενεργοποιημένη γίνεται περισσότερο κατανοητή η διαφορά στην απόδοση των διαφορετικών πειραμάτων που εξετάζονται.

Αν γίνει η σύγκριση του βοηθητικού προγράμματος με την χρήση του excel™ πχ, η διαφορά είναι χαοτική, καθώς το πρώτο παρέχει άμεση απόκριση σε όλες τις αλλαγές και παραμετροποιήσεις που σκοπεύει να κάνει ο χρήστης για την εξερεύνηση του συστήματος.

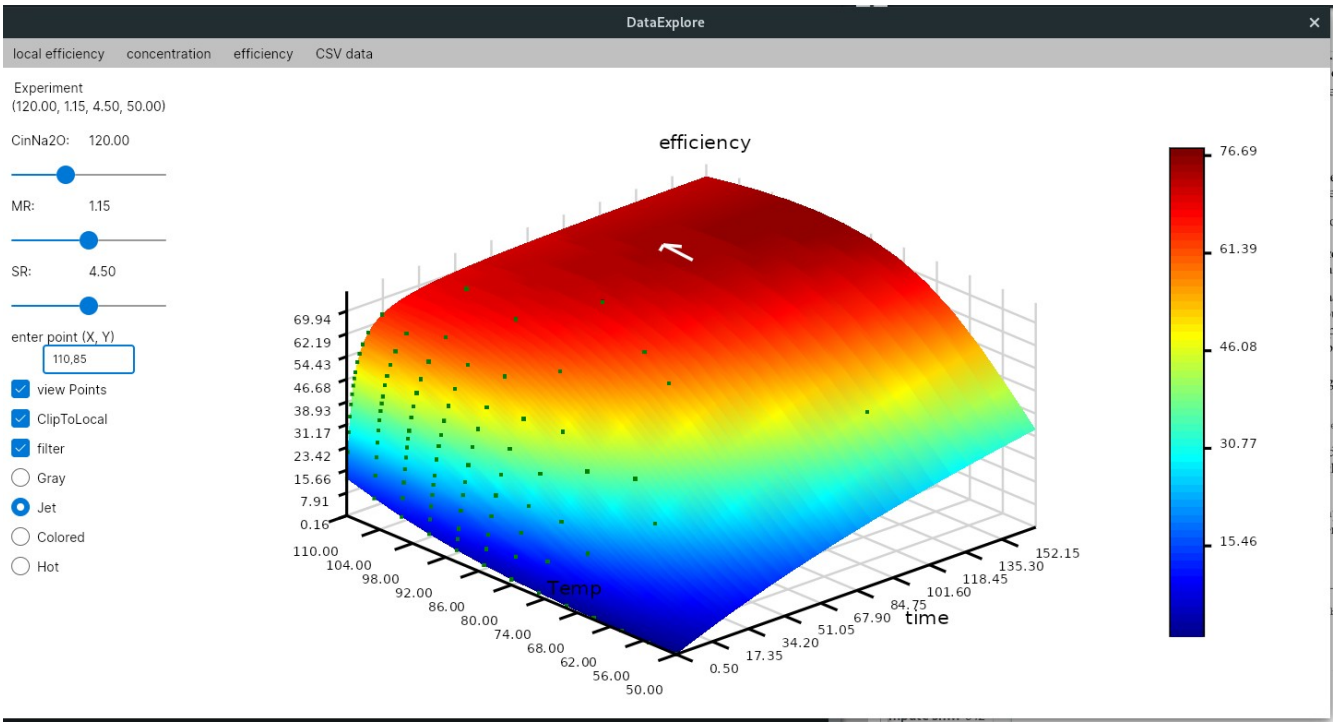

Figure 4.1: γραφική απεικόνιση της επιφάνειας απόδοσης του πειράματος

4.5 Βιβλιοθήκες Μηχανικής Μάθησης

Ο τομέας της μηχανικής μάθησης βρίσκεται στο επίκεντρο της εξέλιξης τα τελευταία χρόνια, βρίσκει εφαρμογή σε πολλά προβλήματα με αποτέλεσμα να έχουν αναπτυχθεί σε διαφορετικές γλώσσες βιβλιοθήκες με την κυριότερη να είναι η C++, και φυσικά η python σαν wrapper, χτίζοντας ένα higher level API πιο φιλικό στο μέσο χρήστη. Εφόσον το συγκεκριμένο πρόβλημα ήταν μία περίπτωση παλινδρόμησης/Regression έγινε χρήση του scikit-learn και XGBoost από την πλευρά της python και του ML.NET από την πλευρά της C#.

4.5.1 Εκπαίδευση Μοντέλων

Οι διαθέσιμοι αλγόριθμοι για την εκπαίδευση του μοντέλου ήταν συνολικά δεκατρείς(13), οπότε και το καταλληλότερο μοντέλο προέκυψε από:

<u>ML.NET</u>

- Sdca (Stochastic dual coordinated ascent),
- LbfgsPoisson,
- OnlineGradientDescent,
- LightGbm,

- FastTree,
- FastTreeTweedie,
- FastForest,
- OlsTrainer (Ordinary Least Squares)

Stochastic Dual Coordinated Ascent

Ο συγκεκριμένος αλγόριθμος ήταν από τους λιγότερο αποδοτικούς αλγορίθμους τόσο στην ακρίβεια, όσο και στον χρόνο εκτέλεσης και για αυτό τον λόγο δεν εφαρμόστηκε παρά μόνο στον πρώτο γύρω του ML.NET-cli [31]. Υπάρχει όμως μία πολύ εκτενής δημοσίευση για αυτόν, οπότε θα δοθεί ένας σύνδεσμος για όποιον αναγνώστη θέλει να μία πλήρη ανάλυση των χαρακτηριστικών του και των ιδιοτήτων του [32].

FastTreeTweedie

Ο αλγόριθμος του *FastTreeTweedie* ήταν από τους τρεις ακριβέστερους, αλλά και ταχύτερους σε όλες τις δοκιμές. Δυστυχώς δεν υπάρχει κάποια δημοσίευση που να αναλύει τις λεπτομέρειες της υλοποίησης του και τα χαρακτηριστικά του. Για λόγους πληρότητας όμως θα γίνει παράθεση ο σύνδεσμος του κώδικα για όποιον αναγνώστη θελήσει να μάθει περισσότερα [33].

LightGbm (Light Gradient Boosting Machine)

Άλλος ένας από τους τρεις πιο γρήγορους και ακριβείς αλγορίθμους σε όλες τις δοκιμές που έγιναν στα datasets. Σε αυτή την περίπτωση είναι διαθέσιμη μία δημοσίευση με λεπτομέρειες, οπότε παρέχεται σύνδεσμος για τα όποιον αναγνώστη θέλει περισσότερες λεπτομέρειες [34].

FastTree

Ο αλγόριθμος FastTree είναι ο τρίτος από τους πιο γρήγορους και αποδοτικούς αλγορίθμους του ML.NET που εφαρμόστηκαν στο dataset σε όλες τις δοκιμές. Είναι μία αποδοτική υλοποίηση του αλγορίθμου MART (Multiple Additive Regression Trees). Για περισσότερες πληροφορίες παραθέτεται η αντίστοιχη δημοσίευση [35].

<u>Sci-kit learn</u>

- SVR (Support Vector Regressors)
- XGBoost (eXtreme Gradient Boost)
- MLPerceptrons (Multilayer Perceptron)
- Keras NN (Neural Networks)
- Pytorch NN (Neural Networks)

Από την πλευρά του sci-kit learn ο αλγόριθμος XGBoost, ήταν αυτός που είχε σε όλες τις δοκιμές την μεγαλύτερη ακρίβεια στον μικρότερο χρόνο εκτέλεσης. Για το XGBoost έχει γίνει ήδη αναλυτική περιγραφή στην αντίστοιχη παράγραφο του κεφαλαίου της Μηχανική Μάθησης. Για το δεδομένο dataset παρατηρούμε ότι οι αλγόριθμοι κατηγορίας FastTree - FastForest πέτυχαν τις καλύτερες επιδόσεις σε όλες τις δοκιμές. Αυτό ίσως να οφείλεται στο γεγονός ότι το dataset είναι προϊόν αναλυτικής συνάρτησης και η εύκολα υπολογιζόμενη παράγωγος της αντικειμενικής συνάρτησης που χρησιμοποιούν αυτοί οι αλγόριθμοι να τους ευνοεί.

4.5.2 Επιλογή Καταλληλότερου Μοντέλου

Μετά το πέρας της εκπαίδευσης ενός μοντέλου μηχανικής μάθησης για την περίπτωση του Regression, η αξιολόγηση του εκπαιδευμένου μοντέλου γίνεται με τους ακόλουθους δείκτες, Metrics.

Table 4.1: Metrics μοντέλων Regression [36]

Metric	περιγραφή
R-Squared(R2)	Εκφράζει την ακρίβεια πρόβλεψης του μοντέλου με μία τιμή σε [-∞, 1.00], δηλαδή την απόκλιση των προβλεπόμενων τιμών από τις πραγματικές. Με το 1.00 να σημαίνει απόλυτη ακρίβεια, το 0.00 να σημαίνει ότι το μοντέλο δίνει σχετικά τυχαίες τιμές, και οι αρνητικές τιμές του R2 ότι το fit δεν ακολουθεί την τάση των δεδομένων και οι τιμές που δίνει είναι ακόμα χειρότερες από τυχαίες (έντονο bias). Αρνητικές τιμές μπορεί να λάβει μόνο σε μη-γραμμικά μοντέλα ή constrained linear regression.
Absolute-loss(MAE)	Mean-Absolute error, μετράει πόσο κοντά είναι οι προβλέψεις στις πραγματικές τιμές. Είναι η μέση τιμή όλων των σφαλμάτων του μοντέλου. Ως σφάλμα του μοντέλου ορίζεται το μέτρο της απόστασης της προβλεπόμενης τιμής από την πραγματική τιμή.
Squared-loss(MSE)	Mean-Squared error, δείχνει πόσο κοντά βρίσκεται η καμπύλη της παλινδρόμησης από ένα σύνολο σημείων του training dataset, υψώνοντας τα σφάλματα στο τετράγωνο, για να δοθεί μεγαλύτερη έμφαση στα μεγαλύτερα σφάλματα.
RMS-loss(RMSE)	Root Mean Squared error, μετράει την διαφορά των προβλεπόμενων τιμών από τις πραγματικές. Είναι η τετραγωνική ρίζα του MSE και έχει την ίδια μονάδα με το **Y** (label) των δεδομένων.
Duration	Η διάρκεια σε δευτερόλεπτα για την εκπαίδευση του μοντέλου.

Οπότε είναι πολύ εύκολο από το πλήθος των εκπαιδευμένων μοντέλων να γίνει η επιλογή του καταλληλότερου βάσει των τιμών του, Metrics. Κάθε αλγόριθμος εκπαιδεύεται σε κάθε επανάληψη με διαφορετικές αρχικές υπερπαραμέτρους και μετά το πέρας της εκπαίδευσης ελέγχεται η αξιοπιστία του σύμφωνα με τα μεγέθη του παραπάνω πίνακα. Από τα μεγέθη του πίνακα το R2 χρησιμοποιείται στο στάδιο του evaluation εξ ορισμού, όσο εκτελείται το mlnet-cli. Η αλλαγή του όμως σε κάποιο διαφορετικό μέγεθος είναι εφικτή [31].

Το μοντέλο με την μεγαλύτερη ακρίβεια, και εν δυνάμει καταλληλότερο για τον χαρακτηρισμό του συστήματος, αποθηκεύεται στον δίσκο σαν εκπαιδευμένο μοντέλο (serialization). Την επόμενη φορά

που ο χρήστης θελήσει να το χρησιμοποιήσει το φορτώνει από τον δίσκο (deserialization) και με την συνάρτηση *.predict()* το εκπαιδευμένο μοντέλο για τις τιμές **X** (features/input) που του δίνονται μπορεί να υπολογίσει το προβλεπόμενο αποτέλεσμα **Y** (label/output) με την ακρίβεια που είχε υπολογιστεί από τα metrics κατά το στάδιο της αξιολόγησης (evaluation).

Serialization: ονομάζεται η διαδικασία κωδικοποίησης σε επιθυμητή μορφή ενός αντικειμένου (ή μιας κατάστασης (που πρακτικά και αυτή αντικείμενο είναι) και η αποθήκευσή του στον δίσκο του Η/Υ σαν κάποιο αρχείο, δηλαδή μιας σειράς από bytes.

Deserialization: ονομάζεται η αντίστροφη διαδικασία, δηλαδή η διαδικασία φόρτωσης ενός αρχείου στην μνήμη του Η/Υ από τον δίσκο (δηλαδή μίας σειράς από bytes) και η ερμηνεία του από το τρέχον πρόγραμμα σαν αξιοποιήσιμα δεδομένα.

Στην πραγματικότητα όταν το λειτουργικό (OS) διαβάζει από τον δίσκο, απλά αντιγράφει από τον δίσκο στην μνήμη μία σειρά από bytes. Όταν αυτή η σειρά από bytes είναι αξιοποιήσιμη μόνο από το πρόγραμμα που την δημιούργησε (binary) όπως συμβαίνει και στην περίπτωση τόσο του ML.NET όσο και του sci-kit, το αποθηκευμένο αρχείο (εκπαιδευμένο μοντέλο) είναι άχρηστο εκτός εκείνης της εφαρμογής. Υπάρχει η εναλλακτική όμως να χρησιμοποιηθεί κάποια πρότυπη μορφή, όπως το ONNX στο πεδίο της μηχανική μάθησης. Εφόσον το πρότυπο είναι κοινό και υποστηρίζεται από τις διαφορετικές βιβλιοθήκες Μηχανικής Μάθησης, τότε ένα εκπαιδευμένο μοντέλο από την βιβλιοθήκη Α μπορεί να αξιοποιηθεί από την βιβλιοθήκη Β. Κάτι τέτοιο προσφέρει τεράστια ευελιξία, αφού ο χρήστης δεν περιορίζεται μόνο σε ένα οικοσύστημα, αλλά μπορεί να εντάξει στην ροή εργασιών του πληθώρα οικοσυστημάτων και εργαλείων για την κάθε περίσταση.

Όλοι οι αλγόριθμοι που εξετάστηκαν υποστηρίζουν το ONNX σαν format, οπότε το εκπαιδευμένο μοντέλο δεν μένει περιορισμένο αποκλειστικά σε κάποια binary μορφή, αλλά μπορεί να φορτωθεί από διαφορετικές βιβλιοθήκες στο μέλλον σε περίπτωση που κριθεί αναγκαίο.

4.5.3 Αποτελέσματα Εκπαιδευμένων Μοντέλων

Κατά την εκπαίδευση των μοντέλων από τους αλγορίθμους καταγράφονται ορισμένοι δείκτες, όπως η ταχύτητα ολοκλήρωσης της διαδικασίας εκπαίδευσης και η ακρίβεια του μοντέλου. Με αυτό τον τρόπο είναι εύκολη η σύγκριση μεταξύ των διαφορετικών αλγορίθμων και των υπερπαραμέτρων τους. Στους δύο πίνακες που θα δείξουμε στην συνέχεια φαίνονται τα αποτελέσματα των εκπαιδευμένων μοντέλων σε αυτές τις 13 περιπτώσεις. Στους παρακάτω πίνακες φαίνονται τα αποτελέσματα των μοντέλων που εκπαιδεύτηκαν από τους δεκατρείς διαθέσιμους αλγορίθμους.

Start Training

	Trainer	RSquared	Absolute-loss	Squared-loss	RMS-loss	Duration	#Iteration
1	SdcaRegression	0,9698	2,48	11,54	3,40	0,9	1
2	LightGbmRegression	0,9977	0,69	0,88	0,94	0,5	2
3	FastTreeRegression	0,9977	0,70	0,90	0,95	0,9	3
4	FastTreeTweedieRegression	0,9976	0,67	0,93	0,96	0,9	4
5	FastForestRegression	0,9193	4,51	30,80	5,55	0,9	5
6	LbfgsPoissonRegression	0,8553	5,40	55,23	7,43	0,5	6
7	OnlineGradientDescentRegression	0,7697	7,81	87,93	9,38	0,3	7
8	OlsRegression	0,9699	2,48	11,48	3,39	0,3	8
9	LightGbmRegression	0,9500	3,43	19,07	4,37	0,3	9
10	FastTreeRegression	-1,6027	25,88	993,69	31,52	0,4	10
11	FastTreeTweedieRegression	0,9982	0,58	0,71	0,84	1,1	11
12	LightGbmRegression	0,9835	1,93	6,32	2,51	0,2	12
13	FastTreeRegression	0,6639	8,41	128,31	11,33	1,9	13
14	FastTreeTweedieRegression	0,9994	0,32	0,22	0,47	3,2	14
15	LightGbmRegression	0,9983	0,59	0,63	0,79	0,5	15
16	FastTreeRegression	-1,5353	25,51	967,93	31,11	0,3	16
17	FastTreeTweedieRegression	0,9988	0,48	0,47	0,69	2,4	17
18	LightGbmRegression	0,9487	3,36	19,59	4,43	0,2	18
19	FastTreeRegression	0,9734	2,38	10,14	3,18	1,3	19
20	FastTreeTweedieRegression	-1,9386	27,38	1121,90	33,49	0,3	20
21	LightGbmRegression	0,6941	9,17	116,80	10,81	0,2	21
22	FastTreeRegression	0,9286	3,94	27,25	5,22	0,5	22
23	FastTreeTweedieRegression	0,9153	4,23	32,32	5,69	0,4	23
24	LightGbmRegression	0,9851	1,82	5,68	2,38	0,3	24
25	FastTreeRegression	-1,6832	26,31	1024,42	32,01	0,7	25
26	FastTreeTweedieRegression	-1,4802	24,84	946,92	30,77	2,0	26
27	LightGbmRegression	0,9992	0,39	0,32	0,57	0,8	27
28	FastTreeRegression	0,9942	1,11	2,23	1,49	0,4	28
29	FastTreeTweedieRegression	-1,8044	26,66	1070,67	32,72	0,3	29
30	LightGbmRegression	0,8377	6,26	61,98	7,87	0,2	30
31	FastTreeRegression	0,9341	3,66	25,17	5,02	0,4	31
32	FastTreeTweedieRegression	0,9987	0,47	0,49	0,70	1,2	32
33	LightGbmRegression	0,9555	3,35	16,99	4,12	0,2	33
34	FastTreeRegression	0,9765	2,25	8,96	2,99	0,5	34
35	FastTreeTweedieRegression	0,9872	1,60	4,87	2,21	0,4	35

[1]

1 Αποτελέσματα αλγορίθμων της βιβλιοθήκης MLNET

Start Training

	Trainer	RSquared	Absolute-loss	Squared-loss	RMS-loss	Duration	#Iteration
1	XGBRegressor	0.9864	1.64	1.64	N/A	0.4	1
2	MLPerceptron	-0.0002	16.68	16.68	N/A	3.0	2
3	Keras.NN	0.9152	4.81	4.81	N/A	21.1	3
4	XGBRegressor	0.9958	0.91	0.91	N/A	1.1	4
5	MLPerceptron	0.9836	1.94	1.94	N/A	17.1	5
6	Keras.NN	0.9389	3.70	3.70	N/A	23.4	6
7	XGBRegressor	0.9965	0.84	0.84	N/A	1.4	7
8	MLPerceptron	-0.0002	16.68	16.68	N/A	3.1	8
9	Keras.NN	0.7694	7.39	7.39	N/A	19.3	9
10	XGBRegressor	0.9971	0.77	0.77	N/A	1.9	10
11	MLPerceptron	-0.0028	16.72	16.72	N/A	21.0	11
12	XGBRegressor	0.9974	0.73	0.73	N/A	2.8	12
13	MLPerceptron	-0.0000	16.67	16.67	N/A	6.5	13
14	Keras.NN	-0.0025	16.66	16.66	N/A	21.1	14
15	XGBRegressor	0.9976	0.70	0.70	N/A	8.6	15
16	MLPerceptron	-0.0000	16.67	16.67	N/A	12.9	16
17	Keras.NN	0.9641	2.81	2.81	N/A	25.7	17
18	XGBRegressor	0.9978	0.68	0.68	N/A	9.5	18
19	MLPerceptron	-0.0000	16.67	16.67	N/A	17.1	19
20	Keras.NN	0.9047	4.73	4.73	N/A	28.2	20
21	XGBRegressor	0.9979	0.65	0.65	N/A	3.2	21
22	MLPerceptron	-0.0126	16.69	16.69	N/A	21.0	22
23	Keras.NN	-0.0007	16.66	16.66	N/A	40.4	23
24	XGBRegressor	0.9981	0.62	0.62	N/A	17.7	24
25	MLPerceptron	0.9785	2.38	2.38	N/A	57.8	25
26	Keras.NN	0.9895	1.51	1.51	N/A	84.3	26

[2]

2 Αποτελέσματα αλγορίθμων της βιβλιοθήκες sci-kit learn και XGBoost.

Κάνοντας σύγκριση των δύο παραπάνω πινάκων, βλέπουμε ότι οι αλγόριθμοι της οικογένειας FastTree (FastTree, LightGbm, XGBoost) είχαν τόσο την μεγαλύτερη ακρίβεια όσο και εξαιρετικά ταχύ χρόνο εκπαίδευσης, οπότε συμπεραίνουμε ότι αυτή η κατηγορία αλγορίθμων εκφράζει με επιτυχία το πρόβλημα. Το καλύτερο μοντέλο που εκπαιδεύτηκε ήταν από το τον αλγόριθμο FastTree, και πλέον αποθηκευμένο στον δίσκο, μπορεί να χρησιμοποιηθεί σε μελλοντική εφαρμογή.

```
mlnet regression

--dataset <path> (REQUIRED)

--label-col <col> (REQUIRED)

--cache <option>

--has-header (Default: true)

--ignore-cols <cols>

--log-file-path <path>

--name <name>

-o, --output <path>

--test-dataset <path>

--train-time <time> (Default: 30 minutes, in seconds)

--validation-dataset <path>

-v, --verbosity <v>

-?, -h, --help
```

Drawing 1: mlnet (cli) παράμετροι. Παράδειγμα χρήσης του mlnet για την αυτόματη εκπαίδευση με τους οχτώ διαθέσιμους αλγορίθμους για την περίπτωση του Regression και την ταυτοποίηση των αλγορίθμων με την μεγαλύτερη ακρίβεια

4.6 Γενετικός Αλγόριθμος

Το επόμενο σκέλος του προβλήματος, μετά το πέρας της εκπαίδευσης του μοντέλου Μηχανικής Μάθησης είναι η βελτιστοποίηση στην επιλογή των αρχικών παραμέτρων με σκοπό την “καλύτερη” απόδοση του πειράματος. Όπως είδαμε στο κεφάλαιο της Πολυπαραγοντικής Βελτιστοποίησης, δεν υπάρχει η μοναδική “καλύτερη” λύση, αλλά περιοχές βέλτιστων ριζών. Η τελική επιλογή φέρει την υποκειμενική κρίση του ερευνητή.

Ο κώδικας για τον γενετικό αλγόριθμο γράφτηκε σαν βιβλιοθήκη γενικής χρήσης, ώστε να μπορεί να χρησιμοποιηθεί πολύ εύκολα και από διαφορετικά μελλοντικά projects.

Ο γενετικός αλγόριθμος λειτουργεί εκτελώντας τα ακόλουθα βήματα:

i. Ορισμός ενός πειράματος (χρωμόσωμα) που περιέχει σαν γονίδια τους παράγοντες του προβλήματος. Στην συγκεκριμένη περίπτωση, φέρει πέντε γονίδια (Temp, MR, SR, CNa_2O, Time).

ii. Ορισμός αρχικού πληθυσμού.

iii. Υπολογισμός της τιμής "συμβατότητας" (fitness) για κάθε χρωμόσωμα του πληθυσμού.

iv. Διασταύρωση των χρωμοσωμάτων του πληθυσμού για την δημιουργία της επόμενης γενιάς.

v. Μετάλλαξη για την διατήρηση της ποικιλομορφίας στον πληθυσμού και την αποφυγή του πρόωρου τερματισμού του αλγορίθμου.

vi. Επιλογή των καταλληλότερων χρωμοσωμάτων για την στελέχωση του πληθυσμού. Στην συγκεκριμένη περίπτωση, από την διασταύρωση προκύπτουν δύο νέοι απόγονοι που αντικαθιστούν του δύο γονείς, οπότε ο πληθυσμός παραμένει σταθερός.

vii. Όταν ολοκληρωθεί ο προκαθορισμένος αριθμός των γενεών ο αλγόριθμος τερματίζει και προτείνει στον χρήστη τα δέκα καλύτερα χρωμοσώματα - πειράματα, δηλαδή δέκα προτάσεις για επιλογή αρχικών παραμέτρων με τις υψηλότερες αποδόσεις της αντίδρασης.

Υπενθυμίζουμε, εκτός από τον όρο "χρωμόσωμα" που προέρχεται από την βιολογία από την οποία είναι εμπνευσμένη αυτή η σειρά αλγορίθμων, μπορεί να χρησιμοποιηθεί και ο όρος ρίζα ή πείραμα, εφόσον κάθε μέλος του πληθυσμού *P* εκφράζει ένα "εικονικό" εργαστηριακό πείραμα.

Το πρώτο και το τελευταίο βήμα εκτελείται μία φορά φορά, ενώ τα ενδιάμεσα είναι μία επαναληπτική διαδικασία. Για την εφαρμογή του αλγορίθμου, ο χρήστης μπορεί να επιλέξει από τις έτοιμες συναρτήσεις που περιέχει η βιβλιοθήκη ή να ορίσει δίκες του καινούργιες αν το κρίνει απαραίτητο για το πρόβλημα που εξετάζει. Η υλοποίηση του αλγορίθμου είναι πολύ απλή, μία σειρά από βήματα (IStep<T> interfaces) που εκτελούνται στην σειρά μέχρι να συγκλίνει ο αλγόριθμος και να τερματίσει. Ο πληθυσμός *P* είναι ένα buffer (συνεχής χώρος στην μνήμη) που περιέχει τα πειράματα - χρωμοσώματα, κάτι που σαν αποτέλεσμα προσθέτει μεγάλη ευελιξία στον χρήστη και τις συναρτήσεις που ενδεχομένως να θελήσει να προσθέσει.

Στο documentation του project, στο *Computational.Optimazation namespace*, βλέπουμε ότι υπάρχει μία σειρά από interfaces, τα οποία προέρχονται όλα από το IStep<T>.

- Iinitialization<T>
- IFitness<T>
- ICrossover<T>
- IMutation<T>
- ISelection<T>

- ITtermination<T>

Καθένα αντιστοιχεί και σε ένα βήμα, όπως είδαμε προηγουμένως. Ο χρήστης είναι υποχρεωμένος να ορίσει μόνο το χρωμόσωμα και την συνάρτηση υπολογισμού του fitness. Για τα υπόλοιπα στάδια μπορεί να χρησιμοποιήσει της υπόλοιπες, έτοιμες, γενικές συναρτήσεις. Όλα τα παρακάτω υπάρχουν αναλυτικότερα στο documentation.

Table 4.2: Πληθυσμός (Population)

Μέθοδος	Περιγραφή
Random Initialization	Για τις τιμές των πειραμάτων που συνθέτουν τον πληθυσμό, οι τιμές των παραμέτρων τους είναι τελείως τυχαίες από το πεδίο ορισμού τους. $CNa_2O \in [100, 160]$ $MR \in [1.0, 1.3]$ $SR \in [2.0, 7.0]$ $T \in [50, 100]$, Θερμοκρασία $t \in [0, 168]$, χρόνος
Heuristic Initialization	Οι τιμές δεν είναι τυχαίες, αλλά έχουν ορισμένη βαρύτητα σε κάποιο χαρακτηριστικό του προβλήματος. (Δεν εφαρμόζεται στην βιβλιοθήκη, αλλά αναφέρεται αποκλειστικά για λόγους πληρότητας).

Fitness

- Για το βήμα του fitness, επειδή εξαρτάται από το χρωμόσωμα του κάθε προβλήματος, δεν δίνεται κάποια έτοιμη συνάρτηση. Στην συγκεκριμένη περίπτωση του προβλήματος εφαρμόζεται ο αλγόριθμος *WSGAFitness (Weighted Sum Approach) [37]*:

i. Παραγωγή τυχαίου αριθμού $u_k \in [0, 1]$ για κάθε παράγοντα k, $k = 1, \ldots, K$

ii. Υπολογισμός τυχαίου βάρους για κάθε παράγοντα k, $w_k = (1/u_k) \sum_{i=1}^{K} u_i$ (4.9)

iii. υπολογισμός του fitness για κάθε χρωμόσωμα $\boldsymbol{x}$, $f(\boldsymbol{x}) = \sum_{k=1}^{K} w_k z_k(\boldsymbol{x})$

Table 4.3: Επιλογή Γονέων (Parent Selection)

Μέθοδος	Περιγραφή
Fitness Proportionate Selection	Είναι ο πιο απλός και συνηθισμένος τρόπος για την επιλογή των “πειραμάτων” που θα επιλεγούν για την διαδικασία του Crossover. Σαρώνεται ο πληθυσμός και η επιλογή γίνεται με πιθανότητα την τιμή fitness του κάθε πειράματος.

Roulette Wheel Selection	Μιμείται την κλασική ρουλέτα με την διαφορά ότι η επιφάνεια που καταλαμβάνει το κάθε “πείραμα” είναι ανάλογη της τιμής fitness του. Μοιάζει αρκετά με το Proportionate Selection. Διαφέρει μόνο στον μηχανισμό επιλογής, καθώς αθροίζει τις τιμές fitness του πληθυσμού και ύστερα από ένα τυχαίο αριθμό [0, Sum] επιλέγει το “πείραμα” για μεταφορά στο matingPool για την διαδικασία του crossover.
Stochastic Universal Sampling (SUS)	Λειτουργεί όπως το Roulette Wheel Selection, με την μόνη διαφορά ότι οι γονείς επιλέγονται ταυτόχρονα.
Tournament Selection	Μιμείται την μέθοδο λειτουργίας ενός τουρνουά. Από ένα τυχαίο υποσύνολο του πληθυσμού, γίνεται διαδοχική σύγκριση των τιμών fitness των πειραμάτων ανά δύο, μέχρι να απομείνουν τα δύο τελευταία που θα συμμετέχουν στο Crossover.
Rank Selection	Δεν χρησιμοποιεί απευθείας την τιμή fitness για να επιλέξει τα “καταλληλότερα” πειράματα, αλλά τα κατατάσσει βάσει αυτής και επιλέγει από την κατάταξη τα ζεύγη. Αυτή η τεχνική έχει νόημα όταν οι διαφορές στις τιμές fitness είναι μικρές και μπορεί να οδηγήσουν την συνάρτηση σε επιλογή πειραμάτων που απλά συνάντησε πρώτα κατά το iteration πριν προφτάσει να φτάσει σε εκείνα με την υψηλότερη τιμή fitness.
Random Selection	Η επιλογή γίνεται τελείως τυχαία από τον πληθυσμό, δεν εφαρμόζεται κάποιο κριτήριο.

Table 4.4: Διασταύρωση (Crossover)

Μέθοδος	Περιγραφή
One Point Crossover	Το νέο πείραμα θα διαθέτει το πρώτο σκέλος παραμέτρων του από το Γονέα Α και το δεύτερο σκέλος από τον Γονέα Β.
Multi Point Crossover	Η πληροφορία των γονέων διαχωρίζεται σε περισσότερα από δύο σκέλη και ο απόγονος (offspring) λαμβάνει τα αντίστοιχα τμήματα για να συμπληρώσει το γονιδίωμα του. Έστω ότι το πείραμα καταλαμβάνει 40bytes (σαν το γονιδίωμα του), αυτή η αλληλουχία διασπάτε σε τμήματα και αντιγράφεται στους απογόνους αντίστοιχα.
Uniform Crossover	γίνεται ομοιόμορφη αντιγραφή στο γονιδίωμα των απογόνων (offsprings) από τα αρχικά πειράματα “γονείς” Α και Β.

Table 4.5: Μετάλλαξη (Mutation)

Μέθοδος	Περιγραφή
Bit Flip Mutation	Αλλάζει με τυχαίο τρόπο, ένα μέρος των bits του πειράματος. Η απεικόνιση των bits δεν φαίνεται στο σχήμα, αλλά είναι κάθε τετράγωνο των bytes διαιρεμένο σε 8 τμήματα. Τέτοιου είδους μετατροπές είναι πολύ συνηθισμένες στον προγραμματισμό και ονομάζονται bit-operations. Όπως θα δούμε στην τελευταία παράγραφο του κεφαλαίου, τέτοιες μετατροπές είναι πανεύκολες χρησιμοποιώντας pointers[3].
Random Resetting	Σε αυτή την περίπτωση ορίζεται μία νέα τυχαία τιμή σε κάποιες από τις παραμέτρους του πειράματος.
Swap Mutation	Επιλέγονται δύο τυχαία bytes (σειρές των 8bits) εντός του πειράματος και ανταλλάζονται οι τιμές μεταξύ τους, δηλαδή τα 8bits του πρώτου A, λαμβάνουν τις τιμές του B και αντίστοιχα τα 8bits του B αποκτούν τις αρχικές τιμές του A.
Scramble Mutation	Μια αλληλουχία των 4bytes (32bits) εντός του πειράματος αποκτάει τελείως τυχαία τιμή.

Όπως μπορεί να φανταστεί ο αναγνώστης, όλες αυτές οι τυχαίες μεταβολές μπορούν να προκαλέσουν τιμές στις παραμέτρους του πειράματος που ξεφεύγουν εκτός του πεδίου ορισμού τους, όπως αναφέρεται παραπάνω ή και τιμές που είναι παντελώς παράλογες π.χ -60.000°C. Κάτι τέτοιο είναι απόλυτα φυσιολογικό από την πλευρά του υπολογιστή, επειδή αυτό που διαβάζει είναι αριθμοί των 64bits, δηλαδή αριθμοί που ανήκουν στο:

$$d \in [\pm 5.0 \times 10^{-324},\ \pm 1.7 \times 10^{308}] \tag{4.10}$$

Για να αποφευχθούν αυτά τα προβλήματα πάντα μετά την μετάλλαξη (mutation) ενός πειράματος, εφαρμόζεται ένας έλεγχος (constraints handling) που διορθώνει τυχόν αστοχίες και τις επαναφέρει στο αντίστοιχο πεδίο ορισμού της παραμέτρου του πειράματος. Το κλασικό

```
clamp(value: f64, min: f64, max: f64)
```

δηλαδή αν η τιμή *value* είναι εντός του πεδίου ορισμού του παράγοντα παραμένει ίδια, αν είναι μικρότερη λαμβάνει την ελάχιστη τιμή, ενώ αν είναι μεγαλύτερη τότε λαμβάνει τη μέγιστη.

Table 4.6: Τερματισμός (Termination)

Μέθοδος	Περιγραφή
Termination	τερματισμός μετά από έναν πεπερασμένο αριθμό γενεών. (Αυτό είναι το μοναδικό κριτήριο

3 Pointer ονομάζεται ένας αριθμός που δείχνει σε μία θέση μνήμης. Μέσω των pointers δίνεται περισσότερη ελευθερία στο τρόπο που επέμβουμε στην μνήμης εκείνης της θέσης ή και σε γειτονικές θέσεις. Με αυτό τον τρόπο γίνεται να ασκηθούν τεχνικές που υπό διαφορετικές συνθήκες δεν θα ήταν εφικτές.

	τερματισμού του αλγορίθμου που παρέχεται έτοιμο στην βιβλιοθήκη)

Με το πέρας της ολοκλήρωσης του γενετικού αλγορίθμου, ο χρήστης έχει στην διάθεσή του μία σειρά από τις καλύτερες αρχικές τιμές των παραγόντων CNa_2O, MR, SR, Temp, Time, ώστε να μπορεί να εξετάσει πειραματικά αποφεύγοντας να ξοδέψει χρόνο και πόρους από το εργαστήριο για τυχαία πειράματα που ήταν καταδικασμένα να επιφέρουν χαμηλή απόδοση εξ αρχής.

Στο σχεδιάγραμμα παρακάτω δίνεται η βασική σχεδίαση του αλγορίθμου. Είναι εμφανές, ότι ο γενετικός αλγόριθμος ταιριάζει στην μέθοδο του Data-Oriented Design [38] και όλα τα οφέλη που προσφέρει για βελτιωμένη απόδοση στην ταχύτητα εκτέλεσης, αφού πρακτικά είναι μονοδιάστατοι πίνακες, “buffers”, όσο αφορά το layout στην μνήμη του υπολογιστή. Είναι σημαντικό, επειδή όπως έχει ήδη ειπωθεί οι Γενετικοί Αλγόριθμοι είναι επαναληπτικές μέθοδοι και αυτό σημαίνει ότι εκτελούνται πολλές αριθμητικές πράξεις συνέχεια. Με την παρούσα μορφή ο Η/Υ μπορεί πολύ εύκολα να τις εκτελέσει παράλληλα αυξάνοντας ραγδαία την ταχύτητα ολοκλήρωσης του αλγορίθμου. Το δεύτερο πλεονέκτημα αυτής της μορφής είναι ότι απλοποιεί σε μεγάλο βαθμό την εφαρμογή ορισμένων συναρτήσεων, όπως BitFlipMutation και ScrambleMutation, καθώς επίσης και την αντιγραφή του νέου πληθυσμού από το matingPool στο population buffer.

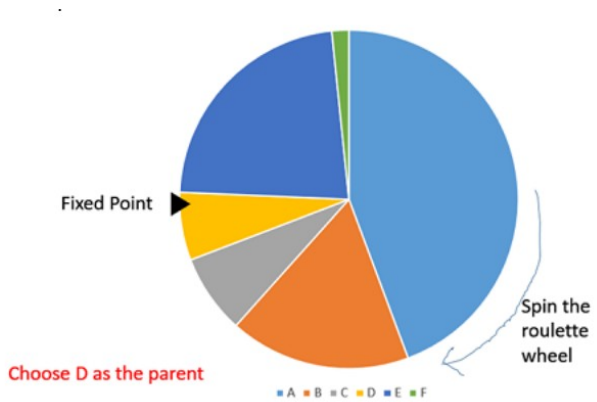

Figure 4.3: Roulette Wheel Selection [0]

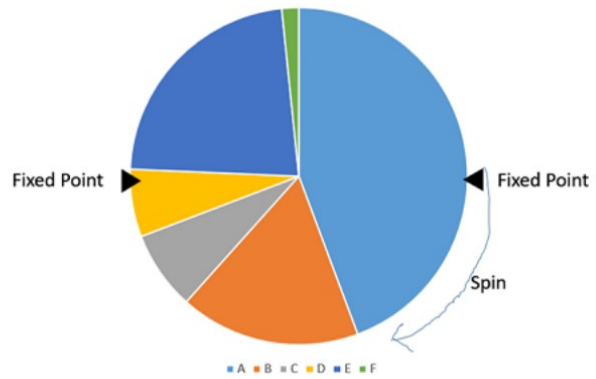

Figure 4.2: Stochastic Universal Sampling (SUS) [0]

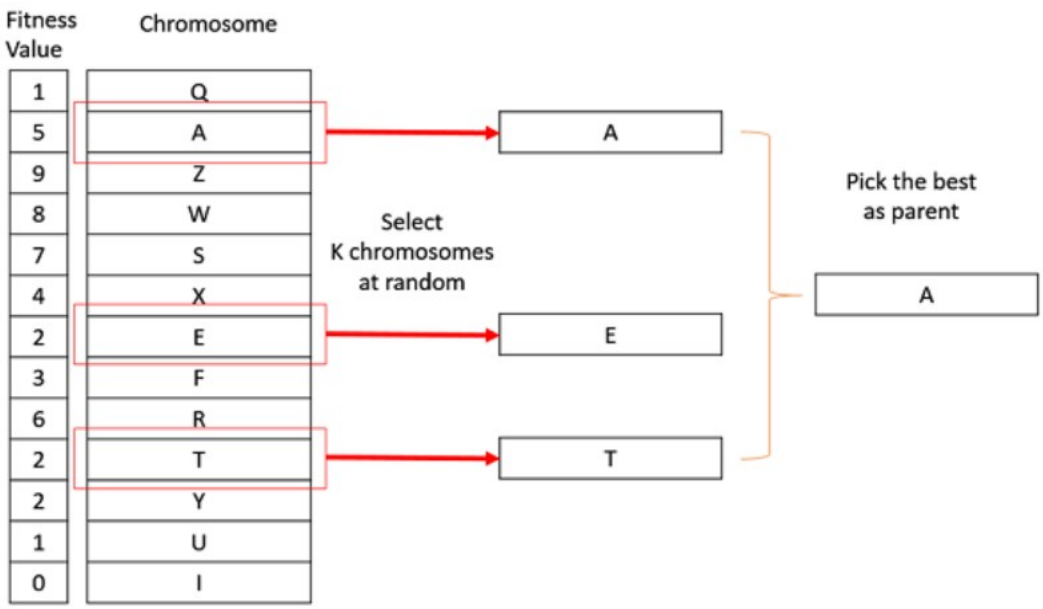

Figure 4.4: Tournament Selection [0]

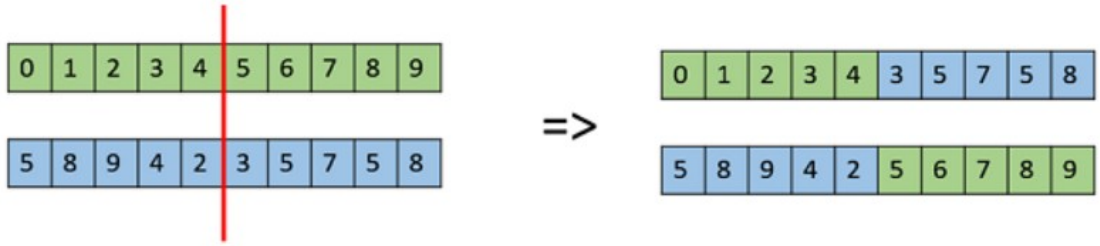

Figure 4.5: One-point Crossover[39]

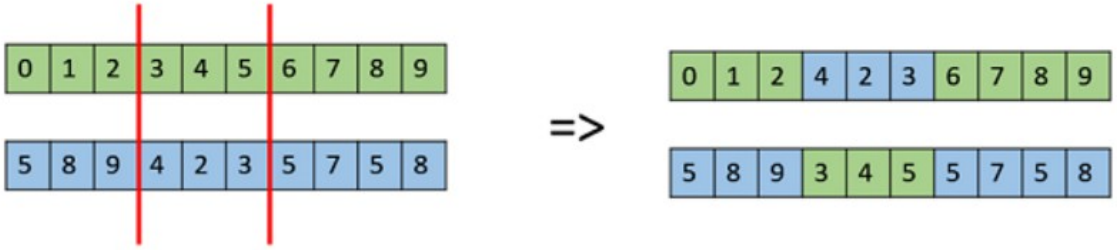

Figure 4.6: Multi-point Crossover [39]

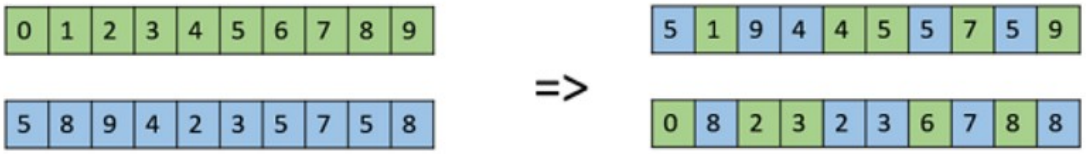

Figure 4.7: Uniform Crossover [39]

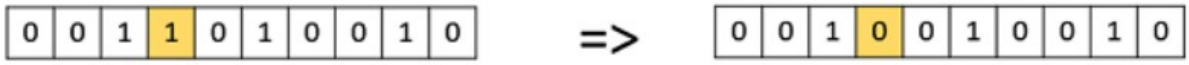

Figure 4.8: Bit-flip Mutation [40]

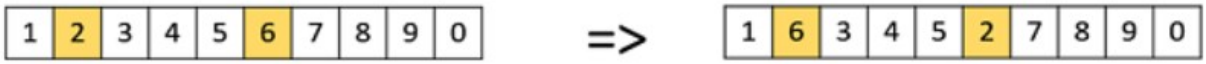

Figure 4.9: Swap Mutation [40]

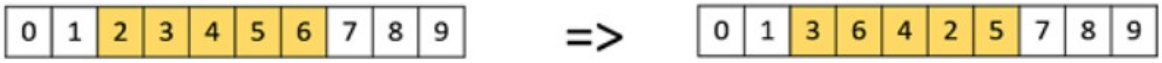

Figure 4.10: Scramble Mutation [40]

```
Chromosome = struct { gene1, gene2, ... geneN };
// blittable-συνεχής σειρά από bytes στην  μνήμη

// οπότε στη συγκεκριμένη περίπτωση θα ισχύει
Experiment = struct { genes = double[5] };
/* Experiment.genes[0] = CNa2O
   Experiment.genes[1] = MR
   Experiment.genes[2] = SR
   Experiment.genes[3] = Temp
   Experiment.genes[4] = Time */

GAimplementation
{
     population: Chromosome[N];
     matingPool: Chromosome[N];
     fitness:    double[N];
     weights:    double[N];   // κάποιοι αλγόριθμοι το χρησιμοποιούν

     command_queue: IStep[]; // οι εντολές που εκτελούνται σε σειρά
}


interface IStep {}
interface IInitialization<T> : IStep{}
interface IFitness<T> : Istep{}
interface ISelection<T> : IStep{}
interface ICrossover<T> : Istep{}
interface IMutation<T> : Istep{}
interface ITermination<T> : IStep{}
```

Drawing 2: υλοποίηση γενετικού αλγορίθμου

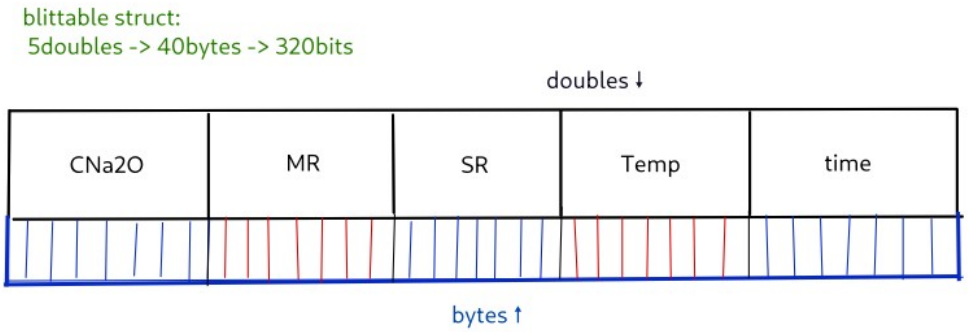

Figure 4.11: Απεικόνιση του πειράματος σαν χρωμόσωμα, σειρά από bytes στην μνήμη του υπολογιστή

5. Εναλλακτικές Επιλογές - Εργαλεία

Κατά την διαδικασία επιλογής εργαλείων βρέθηκαν αρκετές επιλογές που δεν χρησιμοποιήθηκαν, χωρίς να σημαίνει ότι δεν παρουσιάζουν κάποιο γενικότερο ενδιαφέρον. Θα γίνει μία συνοπτική αναφορά σε αυτά τα εργαλεία και στις περιπτώσεις που θα μπορούσαν να εφαρμοστούν.

Κατά κανόνα οι περισσότερες βιβλιοθήκες σχεδιασμένες για αριθμητικούς υπολογισμούς υψηλής απόδοσης (HPC – high performance computing) είναι γραμμένες σε Fortran και C++, και διαθέτουν κάποιο C interface ώστε να μπορούν να χρησιμοποιηθούν και από άλλες γλώσσες προγραμματισμού. Στην περίπτωση της python αυτό ισχύει για όλες τις “διάσημες” βιβλιοθήκες που την έχουν καθιερώσει ως κανόνα στην Μηχανική Μάθηση. Πιο συγκεκριμένα οι βιβλιοθήκες της python είναι γραμμένες σαν ένα high level API πάνω στις native βιβλιοθήκες της C++. Προκύπτει ο εξής προβληματισμός, αποτελεί “μονόδρομο” η χρήση της python ως αναγκαίο κακό για την χρήση όλων αυτών των βιβλιοθηκών;;

Έστω ότι είναι απαραίτητο να σχεδιαστεί κάποιο πρόγραμμα που απαιτεί περισσότερο έλεγχο στους υπολογιστικούς πόρους ή στα επιμέρους τμήματα (core modules), είτε για εσωτερική χρήση, είτε για εμπορική. Σε αυτή την περίπτωση η διανομή του interpreter της python, μαζί με όλα τα .py scripts, δεν αποτελεί την καλύτερη λύση[4]. Επιπλέον ο προηγούμενος προβληματισμός παραμένει, αν η python λειτουργεί μόνο σαν περιτύλιγμα για τις native βιβλιοθήκες, γιατί να μην χρησιμοποιηθούν απευθείας οι native βιβλιοθήκες της C++ στην κυριολεξία από οποιαδήποτε άλλη γλώσσα;

5.1 C2CS

Το C2CS είναι ένα εργαλείο που διαβάζει τους headers (.h) της C, πρακτικά το interface της βιβλιοθήκης και παράγει αυτόματα τα bindings για την C#., δηλαδή μεταφράζει το interface της C για την βιβλιοθήκη της C++ σε ένα interface για την C#. Εφαρμογές σχεδιασμένες στην C# μπορούν ύστερα να καλέσουν τις native βιβλιοθήκες, όπου τις χρειάζονται, χωρίς την ανάγκη να γραφούν τα bindings για την native βιβλιοθήκη χειροκίνητα, αλλά παράγονται αυτόματα από το C2CS.

5.2 TensorFlow.NET & TorchSharp

Είναι οι wrappers αντίστοιχα των TensorFlow και LibTorch για το οικοσύστημα του .NET. Δεν διαθέτουν κάποιο πλεονέκτημα από τις υλοποιήσεις για την python. Αντιθέτως η ομάδα που το συντηρεί έλαβε την ασυνήθιστη απόφαση να παραμείνει πιστή στο API της pyhton και να μην σχεδιάσει ένα API που να συμβαδίζει με την γραφή της C#, ώστε τα παραδείγματα γραμμένα για την python να μπορούν να χρησιμοποιηθούν αυτούσια (σχεδόν copy+paste) από την πληθώρα των ελεύθερων παραδειγμάτων. Η διαφορά σε αυτή την προσέγγιση έχει να κάνει ξεκάθαρα με την επιλογή του οικοσυστήματος που ωφελεί το project (π.χ. AvaloniaUI, ASP Core, Blazor) και την κατεύθυνση που σκοπεύει να επιλέξει ο χρήστης για αυτό. Ένα δεύτερο χαρακτηριστικό είναι ότι τα

4 Cython: είναι μία λύση σε αυτές τις περιπτώσεις, αλλά πάντα είναι προτιμότερα να χρησιμοποιούνται εργαλεία που σχεδιάστηκαν από την αρχή με ορισμένες προδιαγραφές.

TensorFlow.NET και TorchSharp, αποφεύγουν να χρησιμοποιήσουν την python και καλούν τις native βιβλιοθήκες απευθείας, οπότε το project μπορεί να γίνει compiled natively αποφεύγοντας επιπρόσθετα dependencies.

5.3 DiffSharp

Η DiffSharp είναι μια βιβλιοθήκη τανυστών γραμμένη στην F#. Ακολουθεί το παράδειγμα του συναρτησιακού προγραμματισμού (functional programming) και έχει σχεδιαστεί για χρήση σε προβλήματα Μηχανικής Μάθησης, Probabilistic Programming, Βελτιστοποίηση κ.α. Τέλος χρησιμοποιεί ως βάση την LibTorch που της προσδίδει τα ανάλογα οφέλη και χαρακτηριστικά, όπως gpgpu, κ.α.

5.4 F# & TypeProviders

Η F# είναι μία γλώσσα συναρτησιακού προγραμματισμού που ανήκει στο .NET, όποτε μπορεί να συνυπάρχει σε projects μαζί με την C#. TypeProviders είναι ένα χαρακτηριστικό της F#, όπου ο compiler (μεταγλωττιστής) μπορεί από κάποια εξωτερική πηγή να παράγει τύπους και συναρτήσεις για χρήση της F#. Διατίθενται TypeProviders τόσο για την R, όσο και για την python. Η F# είναι μία compiled γλώσσα, αλλά μπορεί να τρέξει και απευθείας σαν script (.fsix) στην γραμμή εντολών. Τα παραπάνω χαρακτηριστικά την καθιστούν μία εν δυνάμει εναλλακτική της python σε συγκεκριμένες περιπτώσεις.

5.5 Gnuplot

Το Gnuplot είναι ένα κλασικό εργαλεία στον τομέα της οπτικής απεικόνισης, ικανό να αποδώσει 2D και 3D γραφήματα. Είναι πάρα πολύ ευέλικτο και εύχρηστο, κάνοντας χρήση μιας DSL (domain specific language) σχεδιασμένης για αυτό, δηλαδή της ψευδογλώσσας του. Έγιναν δοκιμές σε F#, C# αλλά και Zig, και στις τρεις περιπτώσεις οι εφαρμογές μπόρεσαν να χρησιμοποιήσουν το Gnuplot για την απεικόνιση των δεδομένων που υπολόγισαν, ως γραφήματα. Ο λόγος είναι επειδή το Gnuplot είναι εκτελέσιμο πρόγραμμα, όχι βιβλιοθήκη, οπότε δεν χρειάζεται κάποιο C interface και bindings για να τρέξει από την έκαστη εφαρμογή. Αντιθέτως, η έκαστη εφαρμογή το καλεί σαν process (εκτελέσιμο πρόγραμμα), μόλις ολοκληρώσει τον υπολογισμό των δεδομένων. Με αυτό τον τρόπο το Gnuplot μπορεί να προσαρμοστεί απευθείας σε οποιοδήποτε project για την παραγωγή γραφημάτων από δεδομένα, χωρίς κόπο [41].

5.6 Zig

Η Zig είναι μία καινούργια γλώσσα προγραμματισμού, με το πλεονέκτημα ότι υποστηρίζει headers και γενικά βιβλιοθήκες της C, χωρίς κανένα ενδιάμεσο στάδιο από τον χρήστη. Έγινε δοκιμαστική χρήση της βιβλιοθήκης του XGBoost με την Zig και τα αποτελέσματα ήταν ενθαρρυντικά. Για μεταγλωττιστή βασίζεται στο LLMV για την ώρα, (αλλά αυτό μελλοντικά θα αλλάξει), οπότε υποστηρίζει μεγάλο εύρος αρχιτεκτονικών υλισμικού. Πρακτικά είναι μία Systems Programming γλώσσα, κάτι που

συνεπάγεται minimal overhead, η οποία μπορεί εύκολα να χρησιμοποιήσει υπάρχουσες βιβλιοθήκες της C αφού μπορεί να χρησιμοποιήσει C-headers όπως είναι. Διαθέτει υποστήριξη για SIMD και multithread. Πρόκειται για "καθαρόαιμη" high-performance computing (HPC) γλώσσα κάτι που την καθιστά εξαιρετική επιλογή για υπολογιστικές βιβλιοθήκες σε αυτή, όπως επίσης και πολλά άλλα χαρακτηριστικά που ξεφεύγουν από τον σκοπό της εργασίας και μεταβαίνουν σε μηχανική υπολογιστών. Βρίσκεται ακόμα στην έκδοση 0.10, οπότε είναι νωρίς να γίνει χρήση της σε μεγάλα projects, αλλά το βέβαιο είναι ότι διαθέτει όλες τις προδιαγραφές για μελλοντικές χρήσεις. Περισσότερες πληροφορίες στο ziglang.org

5.7 Compute Shaders

Ο compute processor της gpu είναι μία προγραμματιζόμενη μονάδα που δρα ανεξάρτητα από τους υπόλοιπους πυρήνες των shaders. Οι shaders γραμμένη στην GLSL που τρέχουν σε εκείνο τον επεξεργαστή, ονομάζονται Compute Shaders. Ένας Compute Shader έχει πρόσβαση στους ίδιους πόρους με τους υπόλοιπους επεξεργαστές των Shaders, όπως οι Fragment Shaders processors κτλ, δηλαδή σε textures, buffers κ.α. Δεν αποτελεί μέρος του pipeline των γραφικών, αλλά η δράση του φαίνεται σε αλλαγές στα buffers στην μνήμη της GPU. Ένας compute shader δρα σε ένα σύνολο αντικειμένων, που ονομάζεται workgroup. Το workgroup πιο συγκεκριμένα είναι μια αλληλουχία από shader invocations που τρέχουν τον ίδιο κώδικα συνήθως παράλληλα. Οι shaders του ίδιου workgroup μπορούν να μοιράζονται δεδομένα και να καθορίζουν τον συγχρονισμό μεταξύ τους [42].

Οι Compute Shaders χρησιμοποιούνται για γενικούς μαθηματικούς υπολογισμούς (που πλέον λαμβάνουν χώρα στην GPU αντί για την CPU). Δηλαδή ορίζεται ένα buffer (συνεχής μνήμη) στην κάρτα γραφικών και όλοι οι αριθμητικοί υπολογισμοί γίνονται μέσω των shaders, δηλαδή μικρών προγραμμάτων γραμμένων σε GLSL και compiled μέσω του Spir-V. Το Khronos-Group (ο οργανισμός υπεύθυνος για την OpenGL, Vulkan και πολλά άλλα APIs) ήδη εργάζεται στο Vulkan ML.

Φυσικά κάτι τέτοιο είναι πολύ εξειδικευμένο, αλλά αποτελεί την τελευταία επιλογή στους υπολογισμούς υψηλής απόδοσης (High-Performance Computing). Στις περισσότερες περιπτώσεις η Zig με multithread αλγορίθμους θα είναι υπεραρκετή για την πλειονότητα των προβλημάτων σε περίπτωση φυσικά που πρέπει να ακολουθηθεί ο δρόμος του αυτοσχέδιου (custom) λογισμικού για την επίλυση του προβλήματος.

5.8 Προσοχή Στον Σχεδιασμό της Επίλυσης

Όπως γίνεται αντιληπτό από τις προηγούμενες παραγράφους υπάρχει πληθώρα διαθέσιμων εργαλείων για κάθε περίπτωση. Είναι σημαντικό για τον χρήστη από όποιο γνωστικό υπόβαθρο και αν προέρχεται, είτε είναι Μηχανική, είτε Μαθηματικά, είτε Data Science να γνωρίζει τις διαθέσιμες επιλογές, που υπερτερεί και που υστερεί το κάθε εργαλείο και να μπορεί να επιλέξει το καταλληλότερο εργαλείο για το κάθε πρόβλημα. Υπενθύμιση, τα εργαλεία είναι το μέσο για την επίλυση του προβλήματος και η χρήση τους δεν αποτελεί αυτοσκοπό. Πάντα προσαρμοζόμαστε στις απαιτήσεις του προβλήματος και ποτέ δεν παραμορφώνουμε το πρόβλημα, ώστε να γίνει περισσότερο συμβατό με το

εργαλείο που διαθέτουμε την μεγαλύτερη εξοικείωση. Δεν αναλωνόμαστε στο πιο “διάσιμο”, αν δεν έχει σχεδιαστεί για το πρόβλημα που εξετάζεται. Ο χρήστης θα πρέπει να έχει εξοικείωση με πληθώρα οικοσυστημάτων και εργαλείων, ώστε να μπορεί να προσαρμόσει το καταλληλότερο στην ροή εργασιών του με τον αποδοτικότερο και αποτελεσματικότερο τρόπο.

Σε αυτό το σημείο θα γίνει αναφορά στην ομιλία του Mike Acton (Cppcon 2014) και τα τρία άτοπα στην παραγωγή λογισμικού που έχουν επικρατήσει.

a) Το λογισμικό είναι η πλατφόρμα

b) Ο κώδικας σχεδιάζεται γύρω από το μοντέλου του πραγματικού κόσμου

c) Ο κώδικας είναι πιο σημαντικός από την διάταξη των δεδομένων στην μνήμη του Η/Υ

Από την στιγμή που στην Υπολογιστική Μηχανική είναι απαραίτητος ο υπολογισμός μεγάλου πλήθους αριθμητικών πράξεων, το λογισμικό πρέπει να σχεδιάζεται με τις αρχές του Η/Υ.

A) Η πλατφόρμα είναι η πλατφόρμα

B) ο κώδικας πρέπει να σχεδιάζεται βάσει του μοντέλου των δεδομένων

C) Τα δεδομένα είναι προφανώς και ξεκάθαρα το πιο σημαντικό αντικείμενο [5]

Για να γίνει περισσότερο ξεκάθαρος ο στόχος του Acton στην ομιλία του, στην βελτιστοποίηση των αλγορίθμων η Πολυπλοκότητα ($O(n)$) [43],[44] αποτελεί το ένα σκέλος, το δεύτερο σκέλος είναι το hardware - αρχιτεκτονική. Ξεφεύγουμε κατά πολύ από το στόχο της εργασίας, αλλά δίνεται η εξής βιβλιογραφία για το συγκεκριμένο θέμα [45].

5.9 Πόρισμα

Η Μηχανική Μάθηση σαν μέσω έχει αποδείξει την χρησιμότητα της ήδη σε πολλές εφαρμογές και ο κλάδος της Μεταλλουργίας δεν αποτελεί εξαίρεση. Αν το πρόβλημα που μελετάται μπορεί να λυθεί αποτελεσματικότερα με μεθόδους Μηχανικής Μάθησης και με ανεκτό περιθώριο σφάλματος και κριθεί ότι το εκπαιδευμένο μοντέλο είναι εξίσου αποτελεσματικό με ένα αναλυτικό μοντέλο, αλλά χρειάστηκε λιγότερο χρόνο για την εκπαίδευση του, από ότι αντίστοιχα η σχεδίαση του αναλυτικού μοντέλου, τότε μπορεί να εφαρμοστεί σαν μία εναλλακτική. Φυσικά πάντα θα πρέπει να εξετάζεται βάσει του έκαστου προβλήματος και ποτέ να μην θεωρείται πανάκεια. Στο πρόβλημα που μελετήθηκε φάνηκε ότι αν είναι διαθέσιμα αρκετά δεδομένα, τότε μπορεί να υποκαταστήσει το αναλυτικό μοντέλο.

5 Με τον όρο ‘δεδομένα’ ο Acton εννοεί τα bytes στην μνήμη του υπολογιστή, και τον τρόπο που λειτουργεί το hardware, όχι τα δεδομένα ενός dataset. Δηλαδή ότι ο κώδικας σχεδιάζεται βάσει τον προδιαγραφών του hardware και όχι βάσει του προβλήματος όπως το αντιλαμβάνεται ο χρήστης στον “πραγματικό κόσμο”. Και ακριβώς σε αυτό το σημείο είναι που ξεκαθαρίζεται η έμφαση στην σύγκριση Data-Oriented Design vs Object-Oriented Programming (abstractions) στο High-Performance Computing.

6. Παράρτημα

Σε αυτό το κεφάλαιο διατίθεται όλες οι απαραίτητες πληροφορίες για την χρήση του *project*, από το *compilation* του πηγαίου κώδικα και την παραγωγή του εκτελέσιμου προγράμματος (**.exe**), μέχρι το εγχειρίδιο χρήσης του προγράμματος *CLI*, και την ανάγνωση του *documentation - API*, δηλαδή την βιβλιοθήκη του project σαν αναφορά και ευκολότερη πρόσβαση σε αυτό σε μελλοντικές περιπτώσεις που κριθεί αναγκαίο. Η προσθήκη τέτοιου documentation, πέρα από την χρησιμότητα έγινε και για λόγους πληρότητας, επειδή ο πυρήνας του project έχει σχεδιαστεί σαν βιβλιοθήκες γενικής χρήσης και δεν περιορίζεται αποκλειστικά στις ανάγκες της συγκεκριμένης εφαρμογής.

6.1 Λήψη του SDK

Αρχικά θα πρέπει να γίνει λήψη του SDK (Standard Development Kit) από την σελίδα της Microsoft. Το SDK είναι απαραίτητο, επειδή πέρα από το CLR (Common Language Runtime), περιέχει και τον μεταγλωττιστή (compiler) της C#. Ο συγκεκριμένος σύνδεσμος είναι για το .NET 6.0, προτείνεται να γίνει download ο πιο πρόσφατος και η αντίστοιχη αλλαγή στο .csproj αν κριθεί απαραίτητο.

Στην αριστερή στήλη της σελίδας που ανοίγει ο σύνδεσμος, θα υπάρχει μία επιλογή στους Installers για Windows10 (x64), γίνεται click, ώστε να αρχίσει η λήψη. Αντίστοιχα για τα linux στην επιλογή package manager instructions βρίσκονται οι απαραίτητες οδηγίες για τη κάθε διανομή (distro). Όταν ολοκληρωθεί η λήψη του installer(x64) από τον περιηγητή (browser), ο installer εκτελείται για να γίνει η εγκατάσταση του SDK στον υπολογιστή.

6.2 Παραγωγή Εκτελέσιμου Αρχείου

Με την ολοκλήρωση της εγκατάσταση του SDK μέσω της γραμμής εντολών, όπως το command prompt ή το Powershell στα Windows, εναλλακτικά το terminal στα Linux, μπορούν να χρησιμοποιηθούν οι εντολές που παρέχει το SDK. Αρχικά ο χρήστης πρέπει να ανοίξει το PowerShell στον φάκελο του project, στην περίπτωση των Windows, εναλλακτικά το terminal στα Linux και να εφαρμόσει τις εντολές:

```
// Ανοίγουμε την γραμμή εντολών στον φάκελο του project
>cd DataExplore
>dotnet build -c Release
>cd ..
>cd DataExe
>dotnet build -c Release
```

Drawing 3: Σειρά Εντολών για το build του project

Όταν ολοκληρωθεί το build τα αρχεία

- DataExe.deps
- DataExe.dll
- DataExe.exe
- DataExe.pdb
- DataExe.runtimeconfigs.dev.json
- DataExe.runtimeconfigs.json

Θα πρέπει να γίνουν αντιγραφή και επικόλληση στον φάκελο που περιέχει το *DataExplore.exe*. Ο λόγος είναι ότι το *DataExe.exe* διαθέτει μία εντολή, όπως θα δούμε παρακάτω που τρέχει το *DataExplore.exe* σαν process, οπότε χρειάζεται πρόσβαση σε αυτό. Το ιδανικό θα ήταν να υπήρχε ένα αρχείο *.bat* για το powershell, ώστε όλη η διαδικασία να γινόταν τελείως αυτόματα, αλλά δεν διατίθεται κάτι τέτοιο δυστυχώς.

6.3 Ρυθμίσεις Παραμέτρων

Οι παράμετροι που χρησιμοποιεί το πρόγραμμα για την παραγωγή των δεδομένων, τις ρυθμίσεις του φίλτρου και όλες τις λειτουργίες του, βρίσκονται στο αρχείο *config.ini* (configuration.ini δηλαδή). Τα αρχεία τύπου *.ini* είναι πολύ συνηθισμένα στο να φέρουν παραμέτρους προγραμμάτων. Είναι ένα απλό αρχείο κειμένου που μπορεί να ανοίξει με το notepad ή οποιοδήποτε άλλο επεξεργαστή κειμένου. Στην συγκεκριμένη περίπτωση διαθέτει τα παρακάτω πεδία.

Πεδία	Περιγραφή
CinNa2Omin	Κάτω φράγμα της συγκέντρωσης του Na2O
CinNa2Omax	Άνω φράγμα της συγκέντρωσης του Na2O
dCinNa2O	Μεταβολή της συγκέντρωσης του Na2O
MRmin	Κάτω φράγμα της τιμής του Mass Ratio
MRmax	Άνω φράγμα της τιμής του Mass Ratio
dMR	Μεταβολή της τιμής του Mass Ratio
SRmin	Κάτω φράγμα της τιμής του Seed Ratio
SRmax	Άνω φράγμα της τιμής του Seed Ratio
dSR	Μεταβολή της τιμής του Seed Ratio
Tmin	Κάτω φράγμα της τιμής της θερμοκρασίας
Tmax	Άνω φράγμα της τιμής της θερμοκρασίας
dT	Μεταβολή της θερμοκρασίας
Timemin	Κάτω φράγμα της τιμής του χρόνου
Timemax	Άνω φράγμα της τιμής του χρόνου
dt	Μεταβολή του χρόνου
SerializationType	Ποιες τιμές θα αποθηκευτούν στο .csv. Το config.ini περιέχει αναλυτική περιγραφή για τις τιμές διαθέσιμες επιλογές
Size	Το μέγεθος(MB) του κάθε τμήματος του ολικού dataset, βάσει αυτού παράγεται και ο ανάλογος αριθμός από .csv αρχεία
Filter	Ρύθμιση αν ο χρήστης επιθυμεί να χρησιμοποιήσει φίλτρο, για την παραγωγή μικρότερου dataset. Παίρνει τιμές true/false
FilterLength	Πόσες τιμές κρατάει το φίλτρο στην μνήμη του. Δηλαδή με πόσες προηγούμενες αποθηκευμένες τιμές, η νέα τιμή του LocalEfficiency θα συγκριθεί, ώστε να κριθεί αν οι πιο πρόσφατες τιμές που υπολογίστηκαν

	στο loop, θα αποθηκευτούν.
FilterThreshold	Είναι ο συντελεστής που χρησιμοποιείται στην σύγκριση της πρόσφατης τιμής του LocalEfficiency, με τις τιμές που έχει στην μνήμη του το φίλτρο.
CinNa2O> MR> SR> Temp	Η σειρά του loop, δηλαδή η σειρά που μεταβάλλονται οι αρχικές τιμές των πειραμάτων.

Table 6.1: config.ini

Για το *FilterThreshold* ισχύει:

$$\begin{aligned} &threshold \;=\; FilterThreshold \cdot MinLocalEfficiency \\ &if \left| values_{inMemory} - local_{efficiency} \right| > threshold \quad \text{then save at .csv} \end{aligned} \tag{6.1}$$

6.4 DataExe

Το project είναι ένα πρόγραμμα τύπου *CIL (command-line interface)*. Τα προγράμματα τύπου *CIL* είναι πολύ συνηθισμένα σε περιβάλλοντα *Unix* λόγω της αμεσότητας και της ευκολίας χρήσης τους από την γραμμή εντολών. Το project στην πραγματικότητα αποτελείται από δύο διαφορετικά προγράμματα, το *DataExe* και το *DataExplore*, *ο* χρήστης όμως τα χρησιμοποιεί σαν ένα ενιαίο πρόγραμμα που διαθέτει τις παρακάτω λειτουργίες που μπορεί να εισάγει σαν εντολές.

Table 6.2: tool-cli commands

Εντολή	Περιγραφή
--help	Τυπώνει στην κονσόλα την λίστα με όλες τις διαθέσιμες εντολές του *DataExe.exe*
--minmax	Τρέχει πλήρως το loop με τις παραμέτρους του *confing.ini*, υπολογίζει τις μέγιστες και ελάχιστες τιμές των *Concentration*, *Efficiency* και *LocalEfficiency*, και τις τυπώνει στην οθόνη μαζί με μερικά ακόμα διαθέσιμα στοιχεία
--sample	Δημιουργεί ένα μικρό *.csv* αρχείο σαν δείγμα
--clear	Διαγράφει όλο το τυπωμένο κείμενο στην κονσόλα
--generate-data	Διαβάζει τις τιμές στο *config.ini*, παράγει τα αρχεία *.csv* βάσει αυτών και τα αποθηκεύει στον φάκελο DataCSV μαζί με ένα αρχείο *context.txt* που περιέχει ορισμένα χαρακτηριστικά για το dataset
--show-graph	Τρέχει σαν process το *DataExplore.exe*. Ιδιαίτερα χρήσιμο σε συνδυασμό με την εντολή --*run-genetic*, για έλεγχο των αποτελεσμάτων του γενετικού αλγορίθμου
--generate-data --help	Εμφανίζει οδηγίες για την εντολή --*generate-data*
--run-genetic	Τρέχει τον Γενετικό Αλγόριθμο. Θα ακολουθήσει η οδηγία *"insert :size and :generation"* size: το μέγεθος του πληθυσμού των πειραμάτων

	generation: ο αριθμός των επαναλήψεων, N Το input όταν ζητηθεί η εντολή "insert :size and :generation" γράφεται με μορφή ‘200 30’. Εισάγονται οι δύο αριθμοί αντίστοιχα με ένα κενό ενδιάμεσα, χωρίς κάποια άλλη πληροφορία
--run-genetic --help	Τυπώνει κάποιες οδηγίες για την εντολή *--run-gentetic*, οι οποίες δεν ισχύουν σε αυτή την έκδοση του *DataExe.exe*
--exit	Τερματίζει το *DataExe.exe*

6.5 DataExplore

Το *DataExplore* είναι το δεύτερο πρόγραμμα του project που τρέχει η εντολή *--show-graph.* Χρησιμοποιείται για την απεικόνιση (visualization) των επιφανειών Concentration, Efficiecy και LocalEfficiency αντίστοιχα. Διαβάζει τις τιμές στο *config.ini* και παράγει τις επιφάνειες εφαρμόζοντας τα αναλυτικά μοντέλα από το αντίστοιχο κεφάλαιο. Πλεονέκτημα του είναι ότι ο χρήστης μπορεί να μεταβάλει τις τιμές των παραμέτρων από τα αριστερά και να μελετήσει την επιρροή τους από την ανάλογη μεταβολή των επιφανειών. Στο menu στο πάνω μέρος του παραθύρου, γίνεται η επιλογή του γραφήματος που προβάλλεται. Από τα sliders στην αριστερή πλευρά ορίζονται οι τιμές των υπόλοιπων παραμέτρων ως σταθερές. Η περιστροφή της οπτικής του γραφήματος γίνεται με τα βέλη από το πληκτρολόγιο. (πάνω ↑, κάτω ↓, αριστερά ←, δεξιά →).

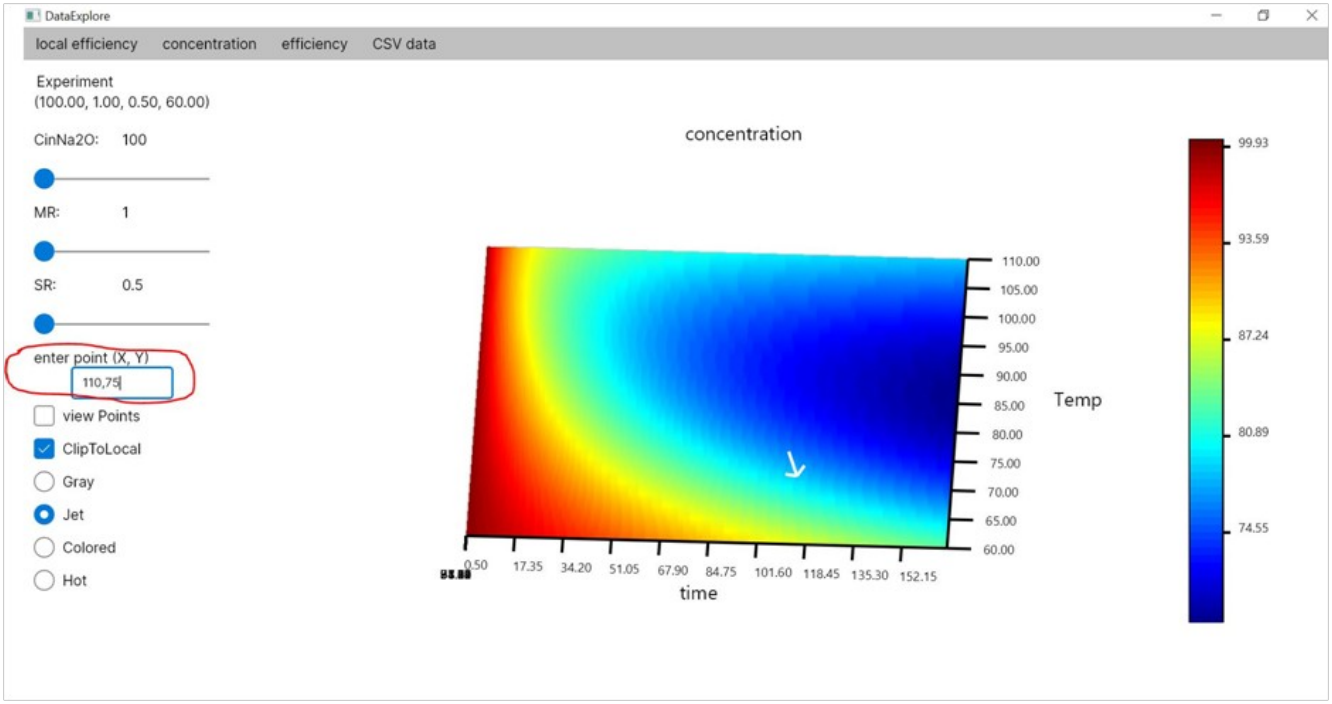

Illustration 6.1: DataExplore

6.5.1 EnterPoint

Στο κουτί *"enter point"* γράφεται η τιμή του σημείου της επιφάνειας για το οποίο προβάλλεται το διάνυσμα κατά το οποίο αυξάνεται η τιμή του Z σε μελλοντικό χρόνο. Αν το βέλος πάψει να είναι εμφανές σημαίνει ότι η τιμή που επιλέχθηκε είναι το τοπικό μέγιστο. (και το βέλος δεν μπορεί να δείξει κατεύθυνση που συνεπάγεται μείωση της τιμής Z). Στο κουτί "*enter point*" οι τιμές γράφονται ως :

$$X,Y \sim \text{και enter} \sim \tag{6.2}$$

Σε όλα τα γραφήματα η τιμή X είναι πάντα ο χρόνος, και η τιμή Y είναι η αντίστοιχη τιμή Y του κάθε γραφήματος που έχει επιλεχθεί από το menu στην πάνω πλευρά του παραθύρου. Προσοχή οι τιμές X, Y που θα οριστούν στο κουτί πρέπει να είναι εντός των πεδίων ορισμού. Δηλαδή των τιμών του *config.ini* που μελετούνται από τα πειράματα. Γίνεται έλεγχος ότι οι τιμές είναι αποδεκτές και από το ίδιο το πρόγραμμα. Σε περίπτωση που έγινε κάποιο λάθος, οι τιμές δεν θα γίνουν δεχτές. Για να μετακινηθεί το βέλος σε νέα θέση, γράφονται στο κουτί νέες τιμές X, Y *~και enter~*.

6.5.2 View Points

Η επιλογή *View Points* που βρίσκεται ακριβώς από κάτω προβάλλει τα σημεία που συνιστούν την επιφάνεια. Παρουσιάζει ενδιαφέρον με την επιλογή *Filter* ενεργοποιημένη, καθώς παρουσιάζει την πυκνότητα των πειραμάτων που αποθηκεύονται στο *.csv*.

6.5.3 Filter

Η επιλογή *filter* όταν είναι ενεργή, χρησιμοποιεί τις παραμέτρους του φίλτρου από το *config.ini* για να περιορίσει το πλήθος των σημείων που προβάλλονται στην επιφάνεια και αντίστοιχα λειτουργεί σαν ένδειξη της αποτελεσματικότητας του φίλτρου και αποτρέπει την αποθήκευση τιμών στο *.csv* που δεν παρουσιάζουν κάποιο ενδιαφέρον. Η διαφορά στο μέγεθος των παραγόμενων *.csv* είναι χαοτική, από 6GB σε ~20MB, κάτι που γίνεται κατανοητό συγκρίνοντας τις αντίστοιχες εικόνες.

6.5.4 ClipToLocal

Η επιλογή *ClipToLocal* αν απενεργοποιηθεί κρατάει τα ολικά ακρότατα που έχουν βρεθεί από τους πειραματισμούς με τις τιμές των sliders. Αν επαναενεργοποιηθεί επαναφέρει τα ακρότατα του γραφήματος στα ακρότατα της προβαλλόμενης επιφάνειας. Με αυτό τον τρόπο, γίνεται πολύ εύκολο να συγκριθεί η αποτελεσματικότητα των παραμέτρων που εξετάζονται, χωρίς να χρειαστεί να αποφανθούμε διαβάζοντας τα *.csv*.

6.5.5 Colormaps

Τέλος στο κάτω μέρος της στήλης βρίσκονται οι χρωματικές κλίμακες. Η επιλογή τους καθορίζει μόνο τις αποχρώσεις της επιφάνειας και της μπάρας στα δεξιά. Δεν έχει κάποια πιο πρακτική χρήση. Αναλόγως την επιφάνεια ενδέχεται κάποια χρωματική κλίμακα να είναι περισσότερη ευδιάκριτη από

τις υπόλοιπες και για αυτό τον λόγο κρίθηκε απαραίτητο να υπάρχει ποικιλία. Στις περισσότερες περιπτώσεις η χρωματική κλίμακα *Jet* είναι επαρκής.

Στο menu υπάρχει και μία τελευταία επιλογή, η CSV Data. Δεν πρόκειται να απασχολήσει τον χρήστη και για αυτό τον λόγο, δεν θα γίνει κάποια εκτεταμένη περιγραφή. Υλοποιήθηκε για ορισμένες λειτουργίες τις εφαρμογής που αργότερα κρίθηκαν παράταιρες.

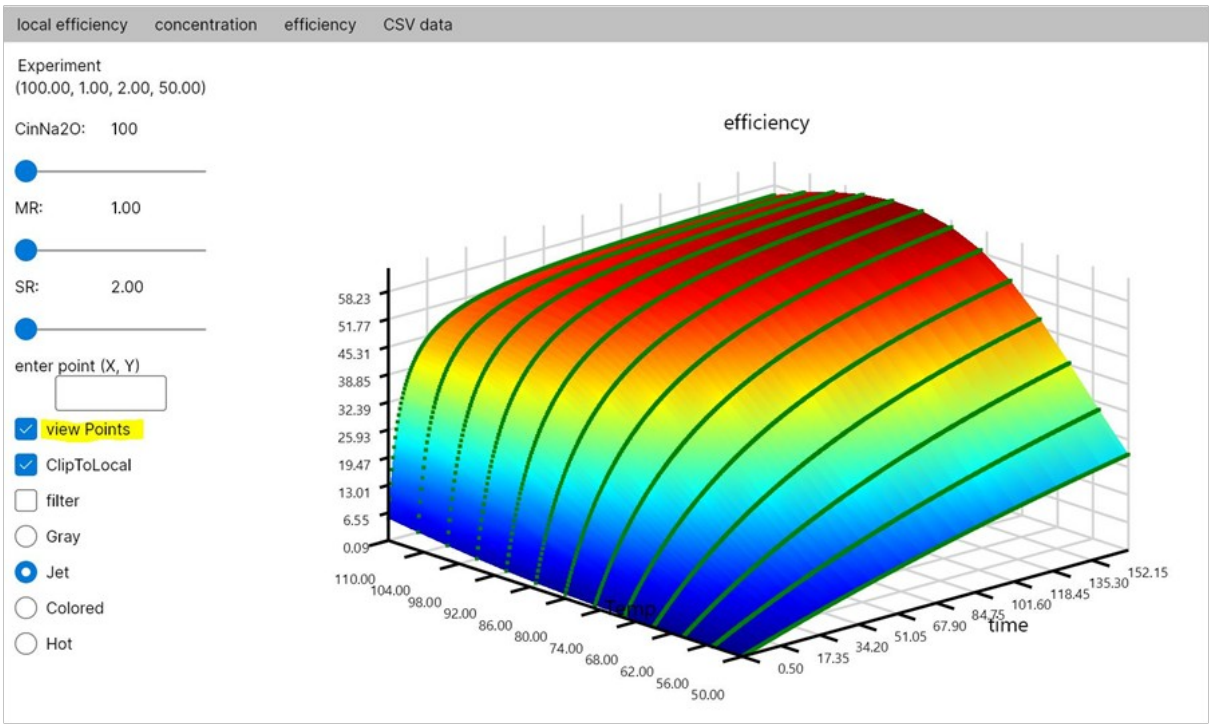

Illustration 6.2: Προβολή σημείων με ανενεργό φίλτρο

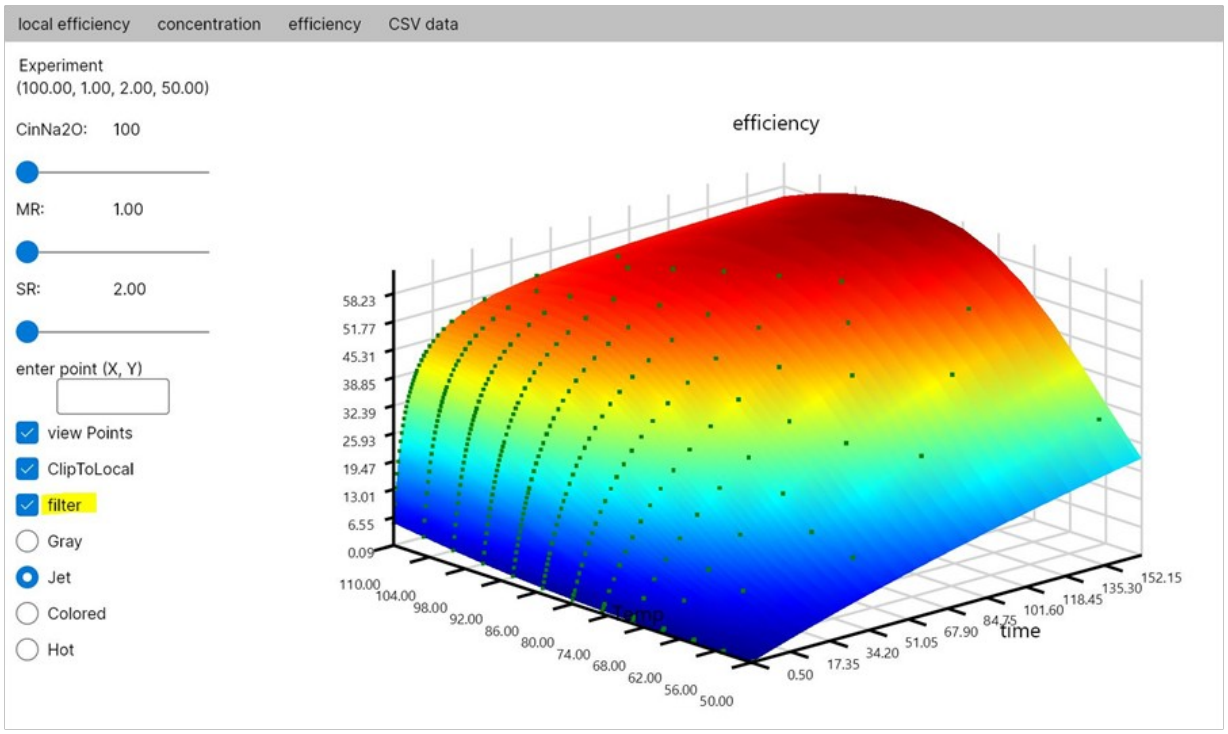

Illustration 6.3: Προβολή σημείων με ενεργό φίλτρο

6.6 Documentation-API

Το *documentation* του *project* παρέχεται σε μορφή στατικής σελίδας (static-site), δηλαδή τα απαραίτητα αρχεία που χρειάζονται, ώστε κάποιος περιηγητής (browser) να το τρέξει τοπικά στον υπολογιστή του χρήστη. Δεν είναι αναρτημένο στο διαδίκτυο, τρέχει *τοπικά* στον υπολογιστή του χρήστη.

Πριν μπορέσει ο περιηγητής να το ανοίξει, θα πρέπει πρώτα να τρέξει σαν σελίδα. Υπάρχουν πολλοί τρόποι για να γίνει αυτό, όπως μέσω docfx, php, nodeJS, python. Αρκεί ο χρήστης να έχει εγκατεστημένο ένα από τα παραπάνω. Η διαδικασία να τρέξει η στατική σελίδα είναι πάρα πολύ απλή, αλλά η εγκατάσταση των παραπάνω προγραμμάτων ίσως να δυσκολέψει τους χρήστες που δεν έχουν επιχειρήσει κάτι αντίστοιχο στο παρελθόν. Παρακάτω δίνονται δύο εναλλακτικές.

a) Λήψη του DocFx από την σελίδα https://github.com/dotnet/docfx/releases **ΟΧΙ** κάποια από τις εκδόσεις 3.0 beta, *αλλά την παλιότερη 2.58.4*

   Εκτέλεση βημάτων 1 & 4.

   https://dotnet.github.io/docfx/tutorial/walkthrough/walkthrough_create_a_docfx_project.html

   Πρώτα ορίζεται το docfx στο PATH, ώστε να μπορεί να χρησιμοποιηθεί απευθείας από οποιοδήποτε directory στο PowerShell. Μετά σύμφωνα με το βήμα 4, εκτελείται η εντολή **"***docfx serve"***.** Θα τυπωθεί μία διεύθυνση http://localhost:8080 , η οποία στην συνέχεια γίνεται copy+paste στην αναζήτηση του διαθέσιμου browser, π.χ Firefox. Τώρα ο χρήστης μπορεί να περιηγηθεί στο *API* του project.

   Σημείωση στα linux, για να τρέξει η παλιότερη έκδοση του Docfx 2.58.4 ο χρήστης θα χρειαστεί να εγκαταστήσει το Mono, ενώ η νεότερη έκδοση 3.0-beta (που δεν το χρειάζεται) δεν λειτουργεί. Για αυτό το λόγο είναι προτιμότερο να ακολουθήσει την δεύτερη επιλογή.

b) Η δεύτερη επιλογή είναι να τρέξει το static server με το module της python.

py -m http.server (Windows)

python -m http.server (Linux)

Αντίστοιχα θα τυπωθεί στην κονσόλα μία προσωρινή διεύθυνση η οποία θα πρέπει να γίνει copy+paste στον brower π.χ Firefox. Αν υπάρχει πρόβλημα με την μεταβλητή PATH, υπάρχουν πολύ οδηγοί online κάνοντας αναζήτηση *"windows 10 Set Path variable"*. Διαφορετικά στην αναζήτηση των Windows *"Edit the System"*.

Η αλήθεια είναι ότι οι περισσότερες διανομές Linux διαθέτουν προ-εγκαταστημένη την python, οπότε στην περίπτωση των linux μία εντολή στο τερματικό είναι αρκετή χωρίς καμία προ-διεργασία. Στα windows δυστυχώς αυτό δεν συμβαίνει και ενδεχομένως αρκετοί χρήστες να ζοριστούν.

Αν και το πιθανότερο είναι λίγοι να χρειαστούν το documentation του API, δεν αλλάζει το γεγονός ότι αποτελεί το απόλυτο ευρετήριο του project, καθώς η περιήγηση σε αυτό δείχνει όλες τις εσωτερικές λειτουργίες και την δομή του.

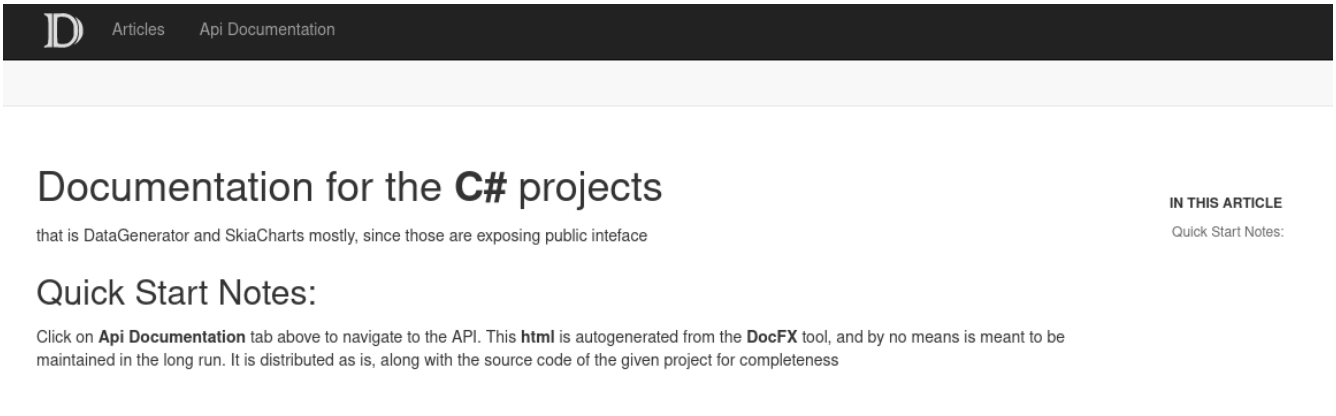

Illustration 6.4: static site (DocFx)

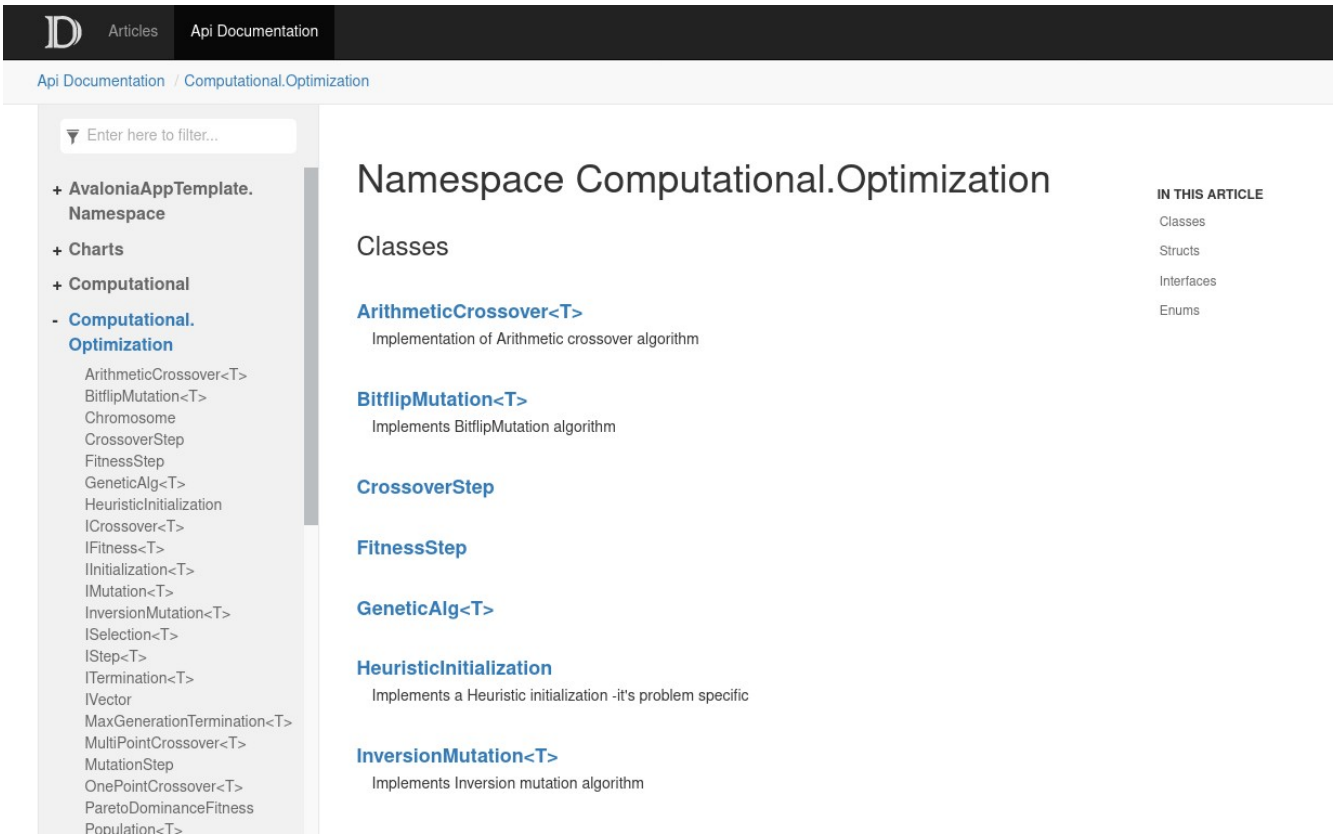

Illustration 6.5: API documentation

Βιβλιογραφία

1: Μούσουλος Λ., Μεταλλουργία Αλουμινίου Μαγνησίου, 1976
2: Χριστόφορος Θ. Σκουφάδης, Καταβύθιση Μονοένυδρης Αλούμινας από Υπέρκορα Αργιλικά Διαλύματα Καυστικού Νατρίου σε Ατμοσφαιρικές Συνθήκες, 2006
3: R.E. Smallman and R.J. Bishop, Modern Physical Metallurgy and Materials Engineering, 1999
4: Alcan, Τεχνική Έκθεση του τμήματος έρευνας και ανάπτυξης τηςALCAN, 2004
5: C. Misra, Solubility of aluminium trihydroxide (hydrargillite) insodium hydroxide solutions, 1970
6: Verdes G., Gout R., Castet S., Thermodynamic properties of the aluminate ion and of bayerite, boehmite, diaspore and gibbsite, 1992
7: Castet S., Verdes G., Gout R., Schott J., Thermodynamic Propertiesof the system Al 2 O 3 -H 2 O to 350 o C, based on solubility measurements ofgibbsite, bayerite, boehmite and diaspore, 1990
8: Panias D., Asimidis P., Paspaliaris I., Solubility of boehmite inconcentrated sodium hydroxide solutions: model development andassessment, 2001
9: Dimitrios Panias, Thermodynamic Deternination of the Stability Area of Boehmite in Al2O3-Na2-H2O Systems, 1999
10: D. Panias, P. Asimidis, I. Paspaliaris, Solubility of boehmite in concentrated sodium hydroxidesolutions: model development and assessment, 2000
11: C. Skoufadis, D. Panias, I. Paspaliaris, Kinetics of boehmite precipitation from supersaturated sodiumaluminate solutions, 2002
12: C. Skoufadis, D. Panias, I. Paspaliaris, Theoretical determination of electrochemicalproperties of boehmite-water interface, as a toolfor understanding the mechanism of boehmiteprecipitation, 2003
13: D. Panias, Role of boehmite/solution interface in boehmite precipitationfrom supersaturated sodium aluminate solutions, 2004
14: D. Panias, A. Krestou, Effect of synthesis parameters on precipitation ofnanocrystalline boehmite from aluminate solutions, 2007
15: , What is ML.NET and how does it work?, , https://docs.microsoft.com/en-us/dotnet/machine-learning/how-does-mldotnet-work
16: Mariette Awad and Rahul Khanna, Efficient Learning Machines: Theories, Concepts, and Applications for Engineers andSystem Designers, 2015
17: , Data transformations, , https://docs.microsoft.com/en-us/dotnet/machine-learning/resources/transforms
18: , Interpret model predictions using Permutation Feature Importance, , https://docs.microsoft.com/en-us/dotnet/machine-learning/how-to-guides/explain-machine-learning-model-permutation-feature-importance-ml-netv
19: Rukshan Pramoditha, 11 Dimensionality reduction techniques you should know in 2021, , https://towardsdatascience.com/11-dimensionality-reduction-techniques-you-should-know-in-2021-dcb9500d388b
20: Leo Breiman, Jerome Friedman, Charles J. Stone, R.A. Olshen, Classification and Regression Trees, 1984
21: Donald Knuth, The Art of Computer Programming, Vol. 1: Fundamental Algorithms, 3rd Edition, 1997
22: Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, Introduction to Algorithms, 3rd Edition, 2009
0: Tianqi Chen, Tong He, XGBoost, eXtreme Gradient Boosting,

23: Ian Goodfellow, Yoshua Bengio, Aaron Courville, Deep Learning,
24: Waldemar Dos Passos, Numerical Methods, Algorithms and Tools in C#, 2009
25: Kalyanmoy Deb, Multi-Objective Optimization using Evolutionary Algorithms: An Introduction, 2011
26: Mariette Awad and Rahul Khanna, Efficient Learning Machines: Theories, Concepts, and Applications for Engineers andSystem Designers, 2015
27: Abdullah Konak, David W. Coit, Alice E. Smith, Multi-objective using genetic algorithms: A tutorial, 2006
28: Tutorialspoint, Genetic Algorithms Tutorial , 2016
29: Prof. Rubin H. Landau, Prof. Manuel J. Páez, Prof. Cristian C. Bordeianu, Computational Physics: Problem Solving with Computers, 2007
30: Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, Introduction to Algorithms, 3rd Edition, 2009
31: , The ML.NET CLI command reference, 2021, https://docs.microsoft.com/en-us/dotnet/machine-learning/reference/ml-net-cli-reference
32: Hsiang-Fu Yu, Cho-Jui Hsieh, Kai-Wei Chang and Chih-Jen Lin,, Large Linear Classification When Data Cannot Fit In Memory,
33: ML.NET, FastTreeTweedie.cs,
34: Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang,Wei Chen, Weidong Ma, Qiwei Ye, Tie-Yan Liu, LightGBM: A Highly Efficient Gradient BoostingDecision Tree,
35: K. V. Rashmi, Ran Gilad-Bachrach, DART: Dropouts meet Multiple Additive Regression Trees, 2015
36: , Evaluate your ML.NET model with metrics, 2021
37: Abdullah Konak, David W. Coit, Alice E. Smith, Multi-objective using genetic algorithms: A tutorial, 2006
38: Richard Fabian, Data-Oriented Design, 2018
0: Artymiak, Jacek, LibreOffice Calc Functions and Formulas Tips, 2011
0: , Genetic Algorithms - Parent Selection, ,
https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_parent_selection.htm
0: , Genetic Algorithms - Parent Selection, ,
https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_parent_selection.htm
39: , Genetic Algorithms - Crossover, ,
https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_crossover.htm
40: , Genetic Algorithms - Mutation, ,
https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_mutation.htm
41: Thomas Williams & Colin Kelley, gnuplot 5.0 An Interactive Plotting Program, 2017
42: John Kessenich, Dave Baldwin and Randi Rost, The OpenGL Shading Language,Version 4.60.7, 2019
43: Donald Knuth, The Art of Computer Programming, Vol. 1: Fundamental Algorithms, 3rd Edition, 1997
44: Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, Introduction to Algorithms, 3rd Edition, 2009
45: Daniel Kusswurm, Modern X86 Assembly Language Programming, 2018