ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

UNIVERSITY OF PIRAEUS

Αυτοματοποιημένη Μηχανική Μάθηση για Προβλήματα Συσταδοποίησης

Πρόγραμμα Μεταπτυχιακών Σπουδών:

Πληροφοριακά Συστήματα κι Υπηρεσίες

Ειδίκευση: Μεγάλα Δεδομένα κι Αναλυτική

Πετράτος Δημήτρης

Επιβλέπων καθηγητής: Δουλκερίδης Χρήστος

Πανεπιστήμιο Πειραιώς

Φεβρουάριος 2023

Περίληψη

Η αύξηση των διαθέσιμων δεδομένων και της διαθέσιμης υπολογιστικής ισχύος, οδηγεί στην αναγνώριση της αξίας των εφαρμογών μηχανικής μάθησης σε πολλά πεδία επιστημών κι εφαρμογών. Η γνώση αξιοποίησης εργαλείων της επιστήμης δεδομένων και της μηχανικής μάθησης, καθιστά πολλές φορές δύσκολη την αξιοποίησή τους από ανθρώπους οι οποίοι δεν έχουν επαφή μαζί τους. Το πρόβλημα αυτό επιδιώκει να το λύσει η αυτοματοποιημένη μηχανική μάθηση, που στόχο της έχει την δημιουργία εργαλείων αυτοματισμού των διαδικασιών μηχανικής μάθησης, επιτρέποντας σε όλους να τα χρησιμοποιούν, χωρίς να έχουν το απαραίτητο γνωστικό υπόβαθρο.

Στην παρούσα διπλωματική εργασία ασχολούμαστε με την αυτοματοποιημένη μηχανική μάθηση για προβλήματα συσταδοποίησης. Η συσταδοποίηση είναι ένα πρόβλημα μη εποπτευόμενης μηχανικής μάθησης, κάτι το οποίο σημαίνει ότι δεν υπάρχει εγγενώς γνώση εξωτερικής πληροφορίας για την αξιολόγηση των αποτελεσμάτων της. Στα πλαίσια λοιπόν της εργασίας χρησιμοποιήσαμε εργαλεία βαθιάς μηχανικής μάθησης και τεχνικές μετα-μάθησης για να λύσουμε το πρόβλημα σύστασης βέλτιστου αλγορίθμου για ένα σύνολο δεδομένων και τεχνικές μπευζιανής βελτιστοποίησης για να βρούμε βέλτιστους συνδυασμούς υπερπαραμέτρων για τους επιλεχθέντες αλγορίθμους.

Abstract

The increasing availability of data and computational power, leads to the recognition of machine learning's value in many scientific and application fields. Utilization knowledge of data science's and machine learning's tools, often makes it difficult to non-experts when they want to use such tools. Automated machine learning aims to overcome this problem, by creating automation tools that can be used in a machine learning pipeline, allowing everyone to use them, without having the necessary domain knowledge.

In this thesis we deal with automated machine learning for clustering problems. Clustering problems are unsupervised machine learning problems, which means that there is no inherently external information to evaluate models' performance. For that purpose we used deep learning and meta-learning techniques to solve the problem of algorithm selection for a dataset and we approched hyperparameter optimization via bayesian optimization's tools.

Ευχαριστίες

Με την ολοκλήρωση της διπλωματικής μου εργασίας σηματοδοτείται κι η ολοκλήρωση των μεταπτυχιακών μου σπουδών. Σε αυτές τις προτάσεις θα ήθελα να ευχαριστήσω όσους συνέβαλαν στην εκπαιδευτική διαδικασία των περασμένων χρόνων, προπτυχιακών και μεταπτυχιακών, δηλαδή τους ανά τα έτη συμφοιτητές και καθηγητές μου.

Ιδιαίτερα για την ολοκλήρωση της παρούσας εργασίας που μου έδωσε την ευκαιρία να ασχοληθώ με την εξαιρετικά ενδιαφέρουσα περιοχή της αυτοματοποιημένης μηχανικής μάθησης, θα ήθελα να ευχαριστήσω τον καθηγητή του τμήματος Ψηφιακών Συστημάτων του Πανεπιστημίου Πειραιώς κ. Χρήστο Δουλκερίδη καθώς επίσης και τον υποψήφιο διδάκτορα του τμήματος Γιάννη Πουλάκη. Οι μεταξύ μας συζητήσεις και η καθοδήγησή τους, θεωρώ ότι με βοήθησαν ιδιαίτερα τόσο στο να έρθω σε επαφή με το θέμα της εργασίας και να το προσεγγίσω με τον τρόπο που εν τέλει το προσέγγισα, όσο και στο να προσπαθήσω να σκεφτώ out of the box.

Τέλος θα ήθελα να ευχαριστήσω την οικογένειά μου και τους φίλους μου για τη στήριξη που μου παρείχαν καθ΄ όλη τη διάρκεια των σπουδών μου.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Κεφάλαιο 1

Μηχανική Μάθηση

1.1 Εισαγωγή

Η ανάγκη των ημερών μας για επεξεργασία κι ανάλυση του ολοένα κι αυξανόμενου όγκου δεδομένων, έχει οδηγήσει τη Μηχανική Μάθηση σε άνθιση. Η Μηχανική Μάθηση είναι η τομή της επιστήμης των υπολογιστών και των μαθηματικών, η οποία σχετίζεται με την έννοια της μάθησης και τη σύνδεσή της με την τεχνητή νοημοσύνη. Για τους σκοπούς αυτούς αναπτύσσονται διάφοροι αλγόριθμοι, οι οποίοι χρησιμοποιώνται για να λύσουν διάφορα προβλήματα τόσο στις επιστήμες όσο και στην αγορά.

Η Μηχανική Μάθηση (Machine Learning) θα μπορούσε να κατατμηθεί σε δύο βασικές κατηγορίες, την εποπτευόμενη (supervised) και τη μη εποπτευόμενη (unsupervised) Μηχανική Μάθηση, χωρίς να λείπουν ενδιάμεσες καταστάσεις, όπως η ενισχυτική μάθηση (reinforcement learning) κα. Στην εποπτεύομενη Μηχανική Μάθηση χρησιμοποιούνται δεδομένα για τα οποία υπάρχει μια a priori αλήθεια, προκειμένου να εκπαιδεύσουμε τα μοντέλα στην αναγνώριση προτύπων μέσα απ΄ τα δεδομένα (πχ στο πρόβλημα της ταξινόμησης) ή στην πρόβλεψη τιμών (πχ σε προβλήματα παλινδρόμησης). Η μη εποπτευόμενη Μηχανική Μάθηση διαφοροποιείται απ΄ την πρώτη ως προς την έλλειψη της αλήθειας πάνω στην οποία εκπαιδεύουμε τα μοντέλα μας.

Ένα βασικό πρόβλημα το οποίο ανήκει σε αυτά της μη εποπτευόμενης μάθησης, είναι αυτό της συσταδοποίησης. Η συσταδοποίηση αφορά στην εύρεση ομάδων εντός ενός συνόλου δεδομένων χωρίς την ύπαρξη προκαθορισμένων κλάσεων όπως στην ταξινόμηση. Η έλλειψη προκαθορισμένων κλάσεων καθιστά τη συσταδοποίηση ανοιχτή ως προς την ερμηνεία των αποτελεσμάτων της.

Εξαιτίας της ευρείας εφαρμογής αλγορίθμων Μηχανικής Μάθησης σε διάφορους τομείς, υπάρχει μεγάλη ζήτηση συστημάτων, τα οποία με χρήση αυτοματισμού, θα επιτρέπουν σε μη ειδικούς (ανθρώπους οι οποίοι έχουν πεδίο γνώσης σε κάποια περιοχή πλην της Επιστήμης των Δεδομένων) να τους χρησιμοποιούν. Για το σκοπό αυτό η περιοχή της Αυτοματοποιημένης Μηχανικής Μάθησης (Automated Machine Learning, AutoML) αρχίζει να συγκεντρώνει ολοένα και περισσότερο ερευνητικό ενδιαφέρον.

Η παρούσα διπλωματική εργασία ασχολείται με την περιοχή της Αυτοματοποιημένης Μηχανικής Μάθησης σε προβλήματα συσταδοποίησης.

1.2 Αυτοματοποιημένη Μηχανική Μάθηση

Η Αυτοματοποιημένη Μηχανική Μάθηση στοχεύει στην αυτοματοποίηση των διαδικασιών στα πλαίσια εφαρμογών μηχανικής μάθησης. Η αυτοματοποίηση μπορεί να αφορά τα παρακάτω δομικά στοιχεία της ροής εργασιών που εκτελούνται όταν ασολούμαστε με τη μηχανική μάθηση:
1)Προετοιμασία των δεδομένων (Data Preparation): Η προετοιμασία των δεδομένων αφορά στην κατάλληλη επιλογή των δεδομένων που θα χρησιμοποιήσουμε, τον καθαρισμό τους κτλ
2)Επιλογή χαρακτηριστικών (Feature Engineering): Η επιλογή των χαρακτηριστικών αφορά στη βέλτιστη επιλογή χαρακτηριστικών (features) των δεδομένων μας, για να τροφοδοτήσουμε αλγορίθμους μηχανικής μάθησης.
3)Επιλογή αλγορίθμου (Algorithm Selection): Η επιλογή αλγορίθμου αφορά στη βέλτιστη επιλογή αλγορίθμου μηχανικής μάθησης, ο οποίος να ταιριάζει καλύτερα στα δεδομένα και το πρόβλημα το οποίο έχουμε να αντιμετωπίσουμε, μέσα από μια πληθώρα επιλογών αλγορίθμων.
4)Βελτιστοποίηση υπερπαραμέτρων (Hyperparameter Optimization, HPO): Η βέλτιστοποίηση των υπερπαραμέτρων, δεδομένου ενός αλγορίθμου μηχανικής μάθησης, αφορά στην εύρεση των υπερπαραμέτρων αυτού, για τις οποίες βελτιστοποιείται μια μετρική απόδοσης.

Περιορίζοντας το πρόβλημα της αυτοματοποιημένης μηχανικής μάθησης στα (3) και (4), δηλαδή στην αυτοματοποίηση της επιλογής αλγορίθμου και την αυτοματοποίηση της εύρεσης βέλτιστων υπερπαραμέτρων, μπορούμε να δούμε το πρόβλημα συνδυαστικά. Αυτό σημαίνει ότι μπορούμε να το δούμε σαν ένα ενοποιημένο πρόβλημα επιλογής αλγορίθμου και βελτιστοποίησης υπερπαραμέτρων (Combined Algorithm Selection and Hyperparameter optimization, CASH) [8]. Πιο φορμαλιστικά, ορίζουμε το πρόβλημα ως εξής:
Έστω $\mathcal{A}$ ένα σύνολο αλγορίθμων μηχανικής μάθησης και A_λ το σύνολο των υπερπαραμέτρων του αλγορίθμου A$\in \mathcal{A}$, τότε έχουμε να λύσουμε το πρόβλημα

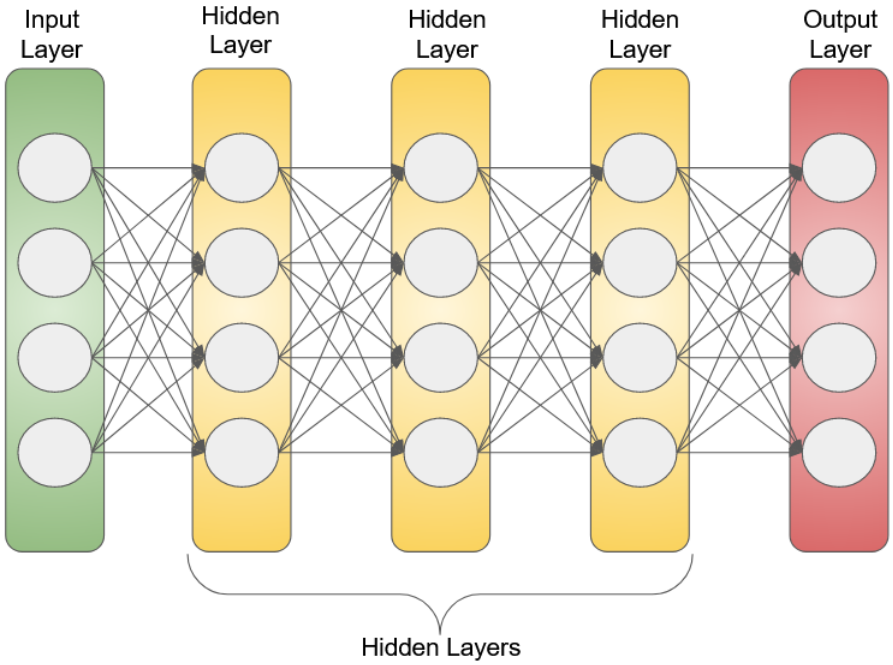
$$A^*,\ \lambda^* = \arg \min_{A\in\mathcal{A},\ \lambda\in\mathcal{A}_\lambda} \mathbb{E}_{(D_{train},D_{val})\sim\mathcal{D}} V(\mathcal{L}, A_\lambda, D_{train}, D_{val})$$

όπου το $V(\mathcal{L}, \mathcal{A}_\lambda, D_{train}, D_{val})$ είναι συνάρτηση κόστους του μοντέλου που παράγεται απ΄ τον αλγόριθμο $\mathcal{A}$ με υπερπαραμέτρους λ, ο οποίος εκπαιδεύεται στο σύνολο D_{train} και αξιολογείται βάσει του D_{val}.

1.3 Νευρωνικά Δίκτυα και Μηχανική Μάθηση

Τα νευρωνινά δίκτυα (Neural Networks) είναι υπολογιστικά μοντέλα μηχανικής μάθησης τα οποία αντλούν έμπευση απ΄ τους βιολογικούς νευρώνες. Αποτελούνται από σύνολα ατομικών μονάδων (τους νευρώνες) που συνδέονται μεταξύ τους, απαρτίζοντας ένα υπολογιστικό γράφημα.

Υπάρχουν διάφορες αρχιτεκτονικές νευρωνικών δικτύων, με την πιο συνήθη να αποτελούν τα νευρωνικά δίκτυα πρόσθιας διάδοσης (Feedforward Neural Networks). Τα feedforward νευρωνικά δίκτυα ορίζονται από ένα κατευθυνόμενο υπολογιστικό γράφημα κι ως προς αυτό διαφοροποιούνται κυρίως από άλλες αρχιτεκτονικές. Η τυπική δομή ενός feedforward νευρωνικού δικτύου αποτελείται από τριών ειδών επίπεδα (layers) στα οποία υπάρχουν οι νευρώνες: το επίπεδο εισόδου (input layer) που δέχεται το διάνυσμα εισόδου του νευρωνικού δικτύου, τα κρυφά επίπεδα (hidden layers) που έχουν τους ενδιάμεσους νευρώνες του νευρωνικού δικτύου και το επίπεδο εξόδου (output layer) που δίνει το τελικό αποτέλεσμα του νευρωνικού δικτύου.



Σχήμα 1.1: Τυπική δομή ενός feedforward νευρωνικού δικτύου

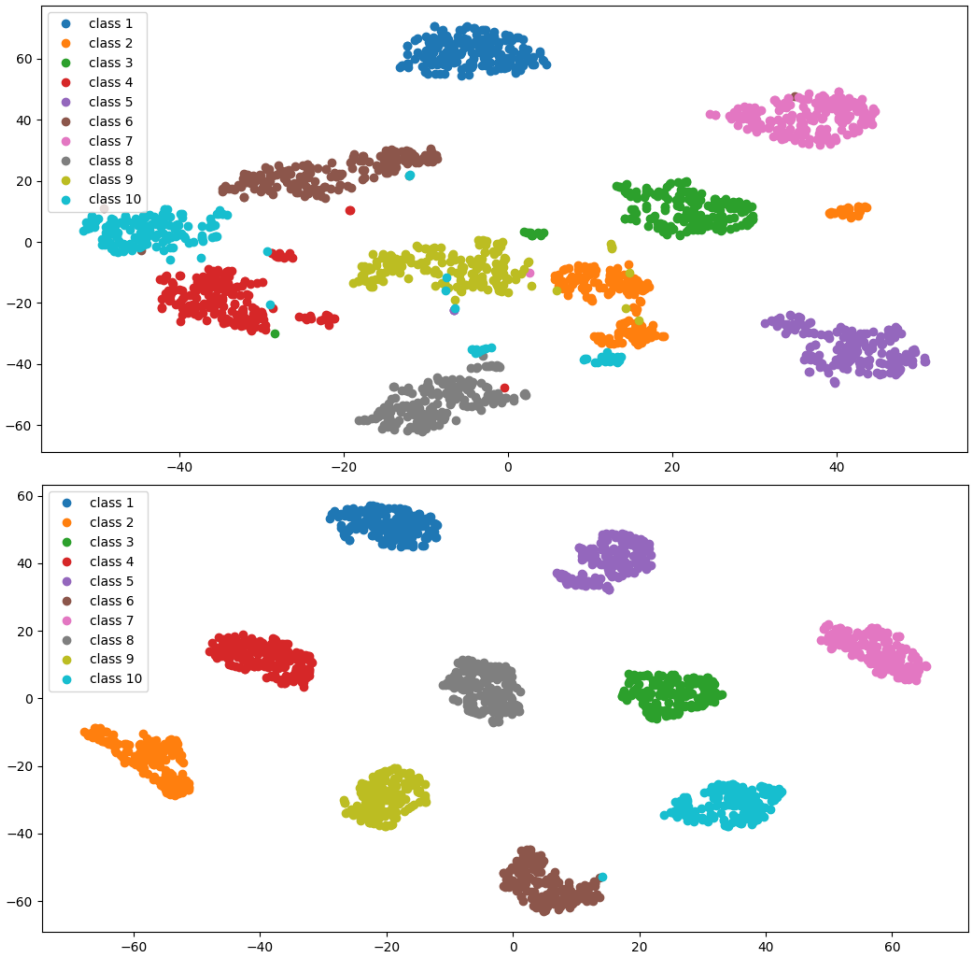
Άλλα βασικά στοιχεία των feedforward νευρωνικών δικτύων είναι τα βάρη (weights) μέσω των οποίων μεταδίδεται η πληροφορία απ΄ τους νευρώνες ενός επιπέδου στους νευρώνες του επόμενου, οι συναρτήσεις ενεργοποίησης (activation functions) που ορίζουν τον τρόπο με τον οποίο μεταδίδεται η πληροφορία από το ένα επίπεδο στο άλλο, η συνάρτηση κόστους (loss function) που ορίζει τον τρόπο εκπαίδευσης του δικτύου κα. Όλες αυτές οι παράμετροι του νευρωνικού δικτύου (πλήθος των hidden layers, πλήθος των units ανά επίπεδο κτλ) καλούνται υπερπαράμετροι του νευρωνικού δικτύου.

Για την εκπαίδευση ενός νευρωνικού δικτύου χρησιμοποιούμε τα δεδομένα που έχουμε στη διάθεσή μας (τιμές/διανύσματα στόχους, κλάσεις κτλ) και η πιο συνήθης διαδικασία εκπαίδευσης τέτοιων δικτύων είναι η οπίσθια διάδοση σήματος (backpropagation) για την οποία χρησιμοποιούμε τον κανόνα της αλυσίδας (chain rule) που αποτελεί τον τρόπο παραγώγισης σύνθετων συναρτήσεων.

Τα τελευταία χρόνια με την ολοένα αυξανόμενη διάθεση υπολογιστικών πόρων, παρατηρήθηκε ότι η χρήση νευρωνικών δικτύων με πολλά επίπεδα και πολλά units αποδίδει καλύτερα σε προβλήματα μεγάλης πολυπλοκότητας όπως η υπολογιστική όραση (computer vision) κα. Η χρήση περισσότερων hidden layers καθιστά ένα νευρωνικό δίκτυο βαθύ (deep) και η περιοχή που ασχολείται με βαθιά δίκτυα καλείται βαθιά μάθηση (deep learning). Η βαθιά μάθηση μας επιτρέπει να μετασχηματίζουμε τα δεδομένα εισόδου από το ένα κρυφό επίπεδο στο άλλο, με τρόπο τέτοιο, ώστε να μπορούμε εν τέλει να δημιουργούμε αναπαραστάσεις αυτού οι οποίες συλλαμβάνουν περισσότερη πληροφορία, χάρη στην οποία μπορούμε να λύσουμε αποτελεσματικότερα το εκάστοτε πρόβλημα.

Τα νευρωνικά δίκτυα, υπό το πρίσμα της μάθησης μέσω αναπαράστασης (representation learning), αποτελούν έναν τρόπο για τον μετασχηματισμό των δεδομένων εισόδου στο τελικό επίπεδο, με σκοπό στη διευκόλυνση περάτωσης της εκάστοτε εργασίας [14]. Για παράδειγμα, οι κλάσεις που μπορεί να μην είναι γραμμικώς διαχωρίσιμες μέσω των δεδομένων ειδόδου, μπορούν μέσω των μετασχηματισμών ενός νευρωνικού δικτύου να γίνουν γραμμικώς διαχωρίσιμες ως αναπαραστάσεις του τελικού (output) επιπέδου. Παρακάτω, χρησιμοποιώντας τη μέθοδο T-distributed Stochastic Neighbor Embedding (t-SNE) που χρησιμοποιείται για να μειώσουμε τη διάσταση των δεδομένων μας (dimensionality reduction) για το σύνολο δεδομένων Optical Recognition of Handwritten Digits Data Set του UCI Machine Learning

Repository, παρουσιάζουμε τη δυνατότητα που προσφέρουν τα νευρωνικά δίκτυα να μετασχηματίζουν τα δεδομένα και να καθιστούν ικανό το γραμμικό διαχωρισμό τους.



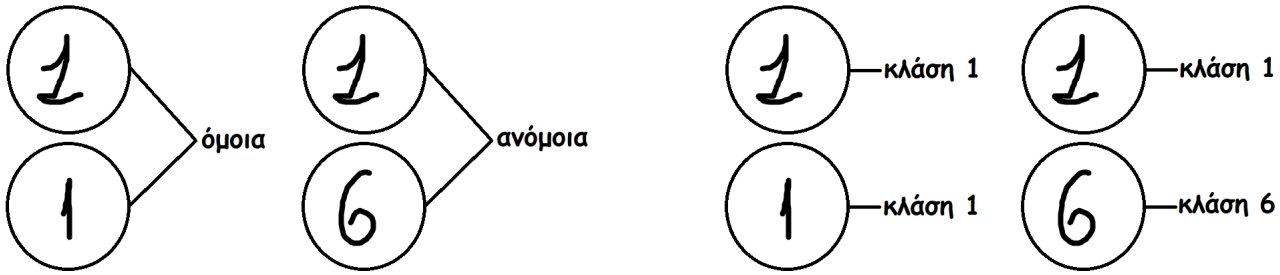
Σχήμα 1.2: Στην πρώτη εικόνα βλέπουμε την t-SNE προβολή του συνόλου δεδομένων σε 2 διαστάσεις για τα δεδομένα 10 διαφορετικών κλάσεων, ενώ στη δεύτερη εικόνα βλέπουμε την αντίστοιχη προβολή των δεδομένων, κρατώντας ως αναπαράστασή τους, αυτή που προκύπτει εκπαιδεύοντας ένα νευρωνικό δίκτυο με 6 κρυφά επίπεδα των 64 units, χρησιμοποιώντας ως συνάρτηση ενεργοποίησης τη ReLU. Το εν λόγο νευρωνικό δίκτυο εκπαιδεύτηκε με τρόπο τέτοιο, ώστε να ταξινομεί τα δεδομένα εισόδου σεκάθε μία εκ των 10 κλάσεων στις οποίες αυτά αντιστοιχούν. Βλέπουμε ότι στη δεύτερη περίπτωση, τα δεδομένα μας φαίνονται με τη βοήθεια του t-SNE για να μειώσουμε τις διαστάσεις τους από 64 σε 2 (χρησιμοποιώντας ως διάνυσμα αναπαράστασης για κάθε είσοδο, αυτό που προκύπτει απ΄ το τελευταίο κρυφό επίπεδο), ότι είναι σχεδόν γραμμικά διαχωρίσιμα, εν αντιθέσει με τα μη επεργεργασμένα δεδομένα του συνόλου δεδομένων.

Κατά τον ίδιο τρόπο, χρησιμοποιώντας συνελικτικά νευρωνικά δίκτυα (Convolutional Neural Networks, CNNs), μπορούμε να αποδώσουμε την πληροφορία μιας εικόνας σε ένα διάνυσμα στον $\mathbb{R}^n$. Τα συνελικτικά δίκτυα είναι feedforward δίκτυα τα οποία χρησιμοποιούν την πράξη της συνέλιξης σε συνελικτικά επίπεδα και μπορούν να χρησιμοποιήσουν τένσορες ως είσοδο, δίνοντάς τους τη δυνατότητα να διαχειρίζονται δεδομένα περισσότερων διαστάσεων (κανάλια, channels). Για το λόγο αυτό χρησιμοποιούνται ευρέως στον τομέα της ανάλυσης εικόνας, καθότι οι έγχρωμες εικόνες απαρτίζονται από τρία κανάλια στο (R,G,B) φάσμα.
Με τη βοήθεια προεκπαιδευμένων μοντέλων μηχανικής μάθησης και χρησιμοποιώντας συνελικτικά νευρωνικά δίκτυα, μπορούμε να εφαρμόσουμε αλγορίθμους μη εποπτευόμενης μηχανικής μάθησης (για παράδειγμα αλγορίθμους συσταδοποίησης) σε δεδομένα όπως είναι οι εικόνες. Η χρήση προεκπαιδευμένων μοντέλων σχετίζεται με την ερευνητική περιοχή της μηχανικής μάθησης που καλείται μάθησης με μεταφορά γνώσης

(transfer learning) κι αφορά στη χρήση εκπαιδευμένων μοντέλων για την επίλυση παραπλήσιων προβλημάτων.

Άλλες αρχιτεκτονικές αποτελούν τα αναδρομικά νευρωνικά δίκτυα (Recurrent Neural Networks, RNNs) που διαφοροποιούνται απ΄ τα feedforward δίκτυα, αφού νευρώνες του ίδιου επιπέδου μπορούν να επαναχρησιμοποιούνται στο ίδιο υπολογιστικό γράφημα και τα συνελικτικά νευρωνικά δίκτυα γραφημάτων (Graph Convolutional Neural Networks, GCNNs) που χρησιμοποιώνται για δεδομένα τα οποία έχουν τη μορφή γραφημάτων.

Μία άλλη περιοχή της μηχανικής μάθησης η οποία μας απασχολεί και στα πλαίσια της παρούσας εργασίας, είναι αυτή της μάθησης ομοιότητας (similarity learning). Η μάθηση ομοιότητας αποτελεί κομμάτι της εποπτευόμενης μηχανικής μάθησης και σχετίζεται με την εύρεση μιας συνάρτησης η οποία ορίζει την ομοιότητα δύο δεδομένων εισόδου. Η ουσιαστική διαφορά της με την ταξινόμηση, έγκειται στο ότι η σύγκριση γίνεται μεταξύ δύο διαφορετικών οντοτήτων και δεν αφορά στην αντιστοίχηση μιας οντότητας με μία κλάση.



Σχήμα 1.3: Διαφορά similarity learning προβλήματος και προβλήματος ταξινόμησης

1.4 Meta-Learning

Με τον όρο Μέτα-Μάθηση (Meta-Learning) αναφερόμαστε στην προσέγγιση της μηχανικής μάθησης κατά την οποία παρατηρώντας πως αλγόριθμοι μηχανικής μάθησης αποδίδουν σε διάφορες εργασίες (ταξινόμηση, παλινδρόμηση κτλ), συλλέγουμε πληροφορία για την απόδοση αυτή, τα μετα-δεδομένα (meta-data), και χρησιμοποιούμε τη γνώση αυτή σε νέες εργασίες. Κατ΄ αυτό τον τρόπο οι διαδικασίες που εκτελούνται στη μηχανική μάθηση μπορούν να αυτοματοποιηθούν, επιταχύνοντας τον κύκλο ροής τους (machine learning pipeline).

Η Μέτα-Μάθηση μπορεί να χρησιμοποιηθεί σε ένα AutoML σύστημα και στα δύο παραπάνω προαναφερθέντα προβλήματα του CASH. Στο κομμάτι της επιλογής βέλτιστου αλγορίθμου, μπορούμε να συνδέσουμε σύνολα δεδομένων με αντίστοιχα μέτα-χαρακτηριστικά και με τη χρήση αυτών να συστήσουμε το βέλτιστο αλγόριθμο που αντιστοιχεί σε ένα σύνολο δεδομένων, σχετικά με μια εργασία (ταξινόμηση, παλινδρόμηση κτλ). Αντίστοιχα, αναφορικά με την εύρεση βέλτιστου συνδυασμού υπερπαραμέτρων, μπορούμε να χρησιμοποιήσουμε μέτα-χαρακτηριστικά για να συστήσουμε υπερπαραμέτρους απ΄ τις οποίες θα ξεκινήσει η διαδικασία της βελτιστοποίησης (warm-start optimization).

Τα μετα-δεδομένα, γνωστά κι ως μέτα-χαρακτηριστικά (meta-features), αποτυπώνουν διάφορους συνδυασμούς χαρακτηριστικών στον κύκλο ροής εφαρμογής μιας διαδικασίας μηχανικής μάθησης όπως είναι ο συνδυασμός των υπερπαραμέτρων ενός αλγορίθμου, τα αποτελέσματα εφαρμογής του αλγορίθμου κα. Η σωστή επιλογή μετα-χαρακτηριστικών έχει να κάνει με την εφαρμογή και το πρόβλημα που έχουμε να λύσουμε.

Συνήθη μετα-χαρακτηριστικά αποτελούν πληροφορίες συσχετιζόμενες με στατιστικά χαρακτηριστικά των συνόλων δεδομένων ή χαρακτηριστικά απόδοσης αλγορίθμων, όπως αυτά που δίνονται στους πίνακες παρακάτω. Πιο ιδιαίτερες προσεγγίσεις ([4], [5]) κάνουν χρήση τενχικών βαθιάς μηχανικής μάθησης με σκοπό την εκμάθηση εξαγωγής μετα-χαρακτηριστικών μέσω νευρωνικών δικτύων. Τέτοια μετα-χαρακτηριστικά μπορούν να χρησιμοποιηθούν για τη σύσταση αλγορίθμων, χαρακτηρίζοντας σύνολα δεδομένων, έχοντας ως βασικό ισχυρισμό ότι όμοια σύνολα δεδομένων έχουν ίδια απόδοση μετρικών για τους ίδιους αλγορίθμους.

Παρακάτω δίνονται κάποια συνήθη μέτα-χαρακτηριστικά της βιβλιογραφίας, όπως αναφέρονται στο [6]

Όνομα	Τύπος	Λογική	Παραλλαγές
Nr instances	n	Speed, Scalability	$\frac{p}{n},\ \log(n), \log\left(\frac{p}{n}\right)$
Nr features	p	Curse of Dimensionality	$\log(p)$, % categorical features
Nr classes	c	Complexity, imbalance	ratio $\frac{min}{max}$ class
Nr missing values	m	Imputation effects	% missing values
Nr outliers	o	Data noisiness	o/n

Πίνακας 1.1: Συνήθη Απλά Μέτα-Χαρακτηριστικά

Όνομα	Τύπος	Λογική	Παραλλαγές
Skewness	$\frac{E(X-\mu_X)^3}{\sigma_X^3}$	Feature normality	$min,\ max,\ \mu, \sigma, q_1$
Kurtosis	$\frac{E(X-\mu_X)^4}{\sigma_X^4}$	Feature normality	$min,\ max,\ \mu, \sigma, q_1$
Correlation	$\rho(X_1, X_2)$	Feature interdependence	$min,\ max,\ \mu, \sigma, \rho_{XY}$
Covariance	$cov(X_1, X_2)$	Feature interdependence	$min,\ max,\ \mu, \sigma, cov_{XY}$
Concentration	$\tau(X_1, X_2)$	Feature interdependence	$min,\ max,\ \mu, \sigma, \tau_{XY}$
Sparsity	$Sparsity(X)$	Degree of discreteness	$min,\ max,\ \mu,\ \sigma$
Gravity	$Gravity(X)$	Inter-class dispersion	
ANOVA p-value	$p-val(X_1, X_2)$	Feature redundancy	$p-val_{XY}$
Coeff. of variation	σ_Y/μ_Y	Variation in target	
PCA $\rho\lambda_1$	$\sqrt{\frac{\lambda_1}{1+\lambda_1}}$	Variance in first PC	$\frac{\lambda_i}{\sum_j \lambda_j}$
PCA skewness		Skewness of first PC	PCA kurtosis
PCA 95%	$\frac{dim_{95\%var}}{p}$	Intrinsic dimensionality	
Class probability	$P(C)$	Class distribution	$min,\ max,\ \mu,\ \sigma$

Πίνακας 1.2: Συνήθη Στατιστικά Μέτα-Χαρακτηριστικά

Όνομα	Τύπος	Λογική	Παραλλαγές
Class entropy	$H(C)$	Class imbalance	
Norm. entropy	$\frac{H(C)}{\log_2 n}$	Feature informativeness	$min,\ max,\ \mu,\ \sigma$
Mutual information	$MI(C, X)$	Feature importance	$min,\ max,\ \mu,\ \sigma$
Uncertainty coeff.	$\frac{MI(C,X)}{H(C}$	Feature importance	$min,\ max,\ \mu,\ \sigma$
Equiv. Nr. feats	$\frac{H(C)}{\overline{MI(C,X)}}$	Intrinsic dimensionality	
Noise-signal ratio	$\frac{\overline{H(C)} - \overline{MI(C,X)}}{\overline{MI(C,X)}}$	Noisiness of data	

Πίνακας 1.3: Συνήθη Μέτα-Χαρακτηριστικά από τον τομέα της Θεωρία Πληροφορίας

Όνομα	Τύπος	Λογική	Παραλλαγές
Fisher's discrimin.	$\frac{(\mu_{c_1} - \mu_{c_2})^2}{\sigma_{c_1}^2 - \sigma_{c_2}^2}$	Separability classes c1, c2 c_1 and c_2	
Volume of overlap		Class distribution overlap	
Concept variation		Task complexity	
Data consistency		Data quality	

Πίνακας 1.4: Συνήθη Μέτα-Χαρακτηριστικά Πολυπλοκότητας

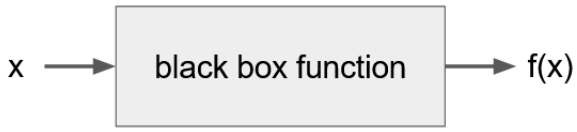
Όνομα	Τύπος	Λογική	Παραλλαγές
Nr nodes, leaves	$\lvert\eta\rvert,\ \lvert\psi\rvert$	Concept complexity	Tree depth
Branch length		Concept complexity	$min,\ max,\ \mu,\ \sigma$
Nodes per feature	$\lvert\eta\chi\rvert$	Feature importance	$min,\ max,\ \mu,\ \sigma$
Leaves per class	$\frac{\lvert\psi_c\rvert}{\lvert\psi\rvert}$	Class complexity	$min,\ max,\ \mu,\ \sigma$
Leaves agreement	$\frac{n_{\psi_i}}{n}$	Class separability	$min,\ max,\ \mu,\ \sigma$
Information gain		Feature importance	$min,\ max,\ \mu,\ \sigma,\ g_i$

Πίνακας 1.5: Συνήθη Model-based Μέτα-Χαρακτηριστικά

1.5 Hyperparameter Optimization

Στη μηχανική μάθηση, συνήθως θέλουμε να βελτιστοποιήσουμε (να βρούμε μέγιστο ή ελάχιστο) μιας black box αντικειμενική συνάρτηση (objective function) f. Η βελτιστοποίηση αφορά την εύρεση σημείου στο οποίο αυτή μεγιστοποιείται ή ελαχιστοποιείται. Η μετάβαση από πρόβλημα εύρεσης σημείου μεγίστου σε εύρεσης σημείου ελαχίστου (κι αντίστροφα) συνήθως είναι εύκολη (πχ αν θέλω να βρω σημείο μεγίστου της συνάρτησης f, τότε αν θέλω να βρω το αντίστοιχο σημείο ελαχίστου τότε αρκεί να βρω το σημείο μεγίστου της συνάρτησης $-f$), συνεπώς μπορούμε να δούμε το πρόβλημα της βελτιστοποίησης ως ενιαίο. Για παράδειγμα, θέλοντας να βρούμε ένα καλό μοντέλο ταξινόμησης χρησιμοποιώντας μηχανές διανυσμάτων στήριξης (Support Vector Machines, SVMs), μπορεί να θέλουμε να βρούμε το μοντέλο εκείνο για το οποίο ελαχιστοποιείται η ακρίβεια ταξινόμησης του testing set, χρησιμοποιώντας διάφορους πυρήνες (kernel functions). Έτσι μπορούμε να θεωρήσουμε τη συνάρτηση f, η οποία παίρνει ως ορίσματα τις υπερπαραμέτρους

του SVM κι επιστρέφει την ακρίβεια ταξινόμησης του testing set. Η συνάρτηση αυτή είναι δύσκολη ως προς τον υπολογισμό της, υπό την έννοια ότι δεν έχει κάποιο κλειστό τύπο και για το λόγο αυτό μπορούμε να τη θεωρήσουμε black box συνάρτηση.



Σχήμα 1.4: Black box συνάρτηση

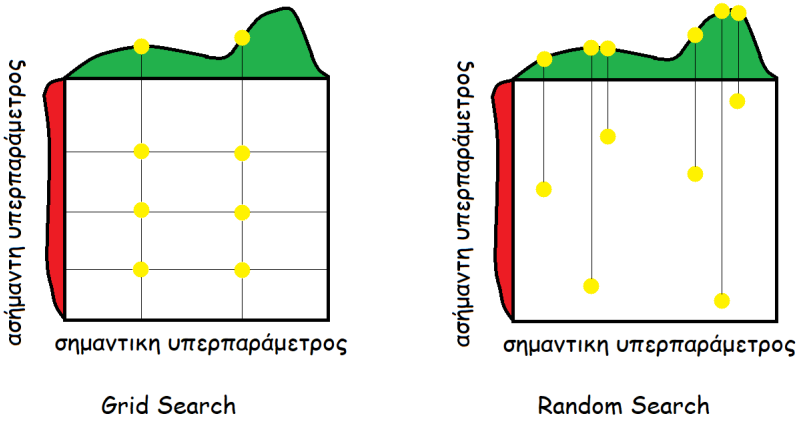
Για τη βελτιστοποίηση black box συναρτήσεων μπορούμε να χρησιμοποιήσουμε διάφορες μεθοδολογίες. Έχουν προταθεί πολλές λύσεις, όπου μεταξύ των πιο συνήθων τακτικών που χρησιμοποιούνται είναι οι:
1) Αναζήτηση σε πλέγμα (Grid Search)
2) Τυχαία αναζήτηση (Random Search)
3) Μέθοδοι μπευζιανής βελτιστοποίησης (Bayesian Optimization)

1.5.1 Grid and Random Search

Κατά το Grid Search δημιουργούμε ένα πλέγμα στο χώρο των υπερπαραμέτρων και κάνουμε αναζήτηση βέλτιστης τιμής της συνάρτησής μας πάνω σε αυτό το πλέγμα. Πιο συγκεκριμένα για έναν αλγόριθμο $\mathcal{A}$ θεωρούμε μια πεπερασμένη διακριτοποίηση $I_1, \ldots, I_k$ των υπερπαραμέτρων $\Lambda_1, \ldots, \Lambda_k$ και κάνουμε αναζήτηση στο καρτεσιανό γινόμενο $I_1 \times \ldots \times I_k$.

Κατά το Random Search, σε αντίθεση με το Grid Search, η αναζήτηση γίνεται σε τυχαίους συνδυασμούς των υπερπαραμέτρων του αλγορίθμου. Συνήθως γίνεται χρήση της ομοιόμορφης κατανομής για την επιλογή των τυχαίων τιμών των υπερπαραμέτρων.

Το παρακάτω παράδειγμα απ΄ το [9], δείχνει την ποιοτική διαφορά στη χρήση Grid και Random Search



Σχήμα 1.5: Στο συγκεκριμένο παράδειγμα προσπαθούμε να βελτιστοποιήσουμε μια συνάρτηση $f(x, y) = g(x)+h(y)$, όπου η παράμετρος $g(x)$ στον οριζόντιο άξονα παρουσιάζει περιοχές με μεγαλύτερα διαστήματα συνεισφοράς στην f, ενώ η $h(y)$ στον κατακόρυφο άξονα παρουσιάζει σχεδόν ομοιόμορφα μικρότερη. Ένα τέτοιο παράδειγμα θα μπορούσε να αποτελεί η προσπάθεια μεγιστοποίησης/ελαχιστοποίησης της συνάρτησης $f(x, y) = \cos(x) + e^{-100y}$.

Εν γένει λοιπόν αν διαφορετικές υπερπαράμετροι έχουν διαφορετική βαρύτητα, εφαρμόζοντας Random Search για ένα δεδομένο πλήθος επαναλήψεων, μπορεί να αποδειχθεί αποδοτικότερο από άποψη χρόνου και ποιοτικών αποτελεσμάτων σε σχέση με το Grid Search.
Επιπλέον το Grid Search έχει σταθερή πολυπλοκότητα ίση με $|I_1| \times \ldots \times |I_k|$, ενώ για το Random Search μπορούμε να θέσουμε ένα threshold επαναλήψεων στην αναζήτησή μας.

1.5.2 Bayesian Optimization

Ένας άλλος τρόπος με τον οποίο μπορούμε να κάνουμε βελτιστοπίηση μιας black box αντικειμενικής συνάρτησης είναι ο λεγόμενος Bayesian Optimization. Οι Bayesian αλγόριθμοι, είναι επαναληπτικοί αλγόριθμοι (χρησιμοποιούν ένα budget threshold επαναλήψεων, δηλαδή έναν αριθμό επαναλήψεων) που κάνουν χρήση πιθανοτικών μοντέλων της αντικειμενικής συνάρτησης στις επιμέρους επαναλήψεις, προκειμένου να κάνουν βελτιστοποίηση, χρησιμοποιώντας υπολογισμούς της αντικειμενικής συνάρτησης στις επιμέρους επαναλήψεις.

Πιο συγκεκριμένα χρησιμοποιούμε μια prior κατανομή για να εκτιμήσουμε ένα μοντέλο υποκατάστασης (surrogate model) σε κάθε επανάληψη. Μια prior κατανομή, είναι μια κατανομή που χρησιμοποιείται προτού ληφθεί υπόψιν μια μέτρηση. Ένα μοντέλο υποκατάστασης χρησιμοποιεί την prior κατανομή για να αποτυπώσει τις πεποιθήσεις μας για την αντικειμενική συνάρτηση.
Με τη βοήθεια μιας acquisition συνάρτησης, επιλέγουμε εκείνο το δείγμα x_t πάνω στο οποίο θα εκτιμήσουμε την αντικειμενική συνάρτηση. Αυτό το κάνει η acquisition συνάρτηση χρησιμοποιώντας το μοντέλο υποκατάστασης, επιλέγοντας εκείνο το x_t για το οποίο η αβεβαιότητα του μοντέλου υποκατάστασης και η ακρίβεια πρόβλεψης είναι μεγάλες [15]. Έτσι οδηγούμαστε τελικά στη βελτιστοποίηση της ίδιας της acquisition συνάρτησης, αντί της συνάρτησης που εξαρχής σκοπεύουμε να βελτιστοποιήσουμε, καθότι η βελτιστοποίησή της συνήθως (με κατάλληλη επιλογή της) είναι λιγότερο υπολογιστικά κοστοβόρα κι εύκολη.
Μια συνήθης acquisition συνάρτηση είναι η expected improvement

$$EI(x_t) = \int_{-\infty}^{\infty} \max(0, \tilde{y} - y) p_s(y|x_t) dy$$

όπου $\tilde{y}$ είναι η βέλτιστη τιμή της f η οποία εκτιμήθηκε στις προηγούμενες επαναλήψεις και p_s η posterior πιθανότητα, όπως εκτιμάται απ΄ το surrogate model s.

Συνήθως χρησιμοποιούνται Gaussian Processes για την εκτίμηση της posterior κατανομής p_s, ενώ συχνά χρησιμοποιείται και η μέθοδος Tree Parzen Estimator (TPE) [10]. Ένας Tree Parzen Estimator εκτιμά τις πιθανότητες $p(x_t|y)$ και $p(y)$, όπου η $p(x_t|y)$ δίνεται ως

$$p_s(x_t|y) = \begin{cases} l(x_t), & y < \tilde{y} \\ \\ g(x_t), & y \geq \tilde{y} \end{cases}$$

όπου l είναι η πυκνότητα που σχηματίζεται απ΄ τις παρατηρήσεις x_t για τις οποίες η αντικειμένική συνάρτηση υπολογισμένη στην παρατήρηση x_t είναι μικρότερη της τιμής $\tilde{y}$ (δηλαδή της βέλτιστης τιμής της αντικειμενικής συνάρτησης πάνω από το σύνολο των προηγούμενων παρατηρήσεων) και g είναι η πυκνότητα των υπόλοιπων παρατηρήσεων. Κατά τη μεθοδολογία επιλέγεται ένας σταθερός αριθμός γ έτσι ώστε $p_s(y < \tilde{y}) = \gamma$.

Παρακάτω δείχνουμε αναλυτικά πως χρησιμοποιώντας την παραπάνω ιδέα για την εκτίμηση της $p(x_t|y)$ μέσω της κλαδωτής συνάρτησης των l και g, μπορούμε να μεγιστοποιήσουμε την expected improvement, χρησιμοποιώντας βασικά θεωρήματα μαθηματικής ανάλυσης και θεωρίας πιθανοτήτων.
Αρχικά χρησιμοποιώντας τον ορισμό της max συνάρτησης και της αθροιστικότητας του ολοκληρώματος ως προς τα άκρα ολοκλήρωσης, έχουμε

$$EI(x_t) = \int_{-\infty}^{\tilde{y}} (\tilde{y} - y) p_s(y|x_t) dy$$

Απ΄ το θεώρημα του Bayes έχουμε ότι $p_s(y|x_t) = \dfrac{p_s(x_t|y)p_s(y)}{p_s(x)}$ κι απ΄ το θεώρημα ολικής πιθανότητας, ότι $p_s(x) = \displaystyle\int_{-\infty}^{\infty} p_s(x|y)p(y)dy$. Συνεπώς απ΄ τη αθροιστικότητα του ολοκληρώματος ως προς τα άκρα ολοκλήρωσης, τον ορισμό της $p_s(x|y)$ και την $p_s(x)$, έχουμε ότι

$$p_s(x) = \int_{-\infty}^{\tilde{y}} l(x)p_s(y)dy + \int_{\tilde{y}}^{\infty} g(x)p_s(y)dy = l(x)\gamma + g(x)(1-\gamma)$$

όπου το $\gamma = p_s(y < \tilde{y})$ όπως ορίστηκε και παραπάνω. Έτσι έχουμε ότι

$$EI(x_t) = \int_{-\infty}^{\tilde{y}} (\tilde{y} - y)\frac{p_s(x_t|y)p_s(y)}{p_s(x_t)}dy = \frac{1}{p_s(x_t)}\int_{-\infty}^{\tilde{y}} p_s(x_t|y)p_s(y)dy =$$
$$\frac{l(x_t)}{p_s(x_t)}\int_{-\infty}^{\tilde{y}} (\tilde{y} - y)p_s(y)dy = \frac{l(x_t)}{p_s(x_t)}\left[\tilde{y}\int_{-\infty}^{\tilde{y}} p_s(y)dy - \int_{-\infty}^{\tilde{y}} yp_s(y)dy\right] =$$
$$\frac{l(x_t)}{p_s(x_t)}\left[\tilde{y}\gamma - \int_{-\infty}^{\tilde{y}} yp_s(y)ds\right] = \frac{l(x_t)}{l(x_t)\gamma + g(x_t)(1-\gamma)}\left[\tilde{y}\gamma - \int_{-\infty}^{\tilde{y}} yp_s(y)ds\right] =$$
$$\left(\gamma + \frac{g(x_t)}{l(x_t)}(1-\gamma)\right)^{-1} \cdot \left[\tilde{y}\gamma - \int_{-\infty}^{\tilde{y}} yp_s(y)ds\right] \propto \left(\gamma + \frac{g(x_t)}{l(x_t)}(1-\gamma)\right)^{-1}$$

Η τελευταια σχέση, μας λέει ότι η μεγιστοποίηση του expected improvement εξαρτάται μόνο από τον όρο στην παρένθεση (καθότι ο όρος στην αγκύλη δεν περιέχει x_t) κι άρα x_t για τα οποία το $\gamma + \dfrac{g(x_t)}{l(x_t)}(1-\gamma)$ να είναι μικρό, οπότε επιλέγουμε πάντα κατάλληλα x_t.

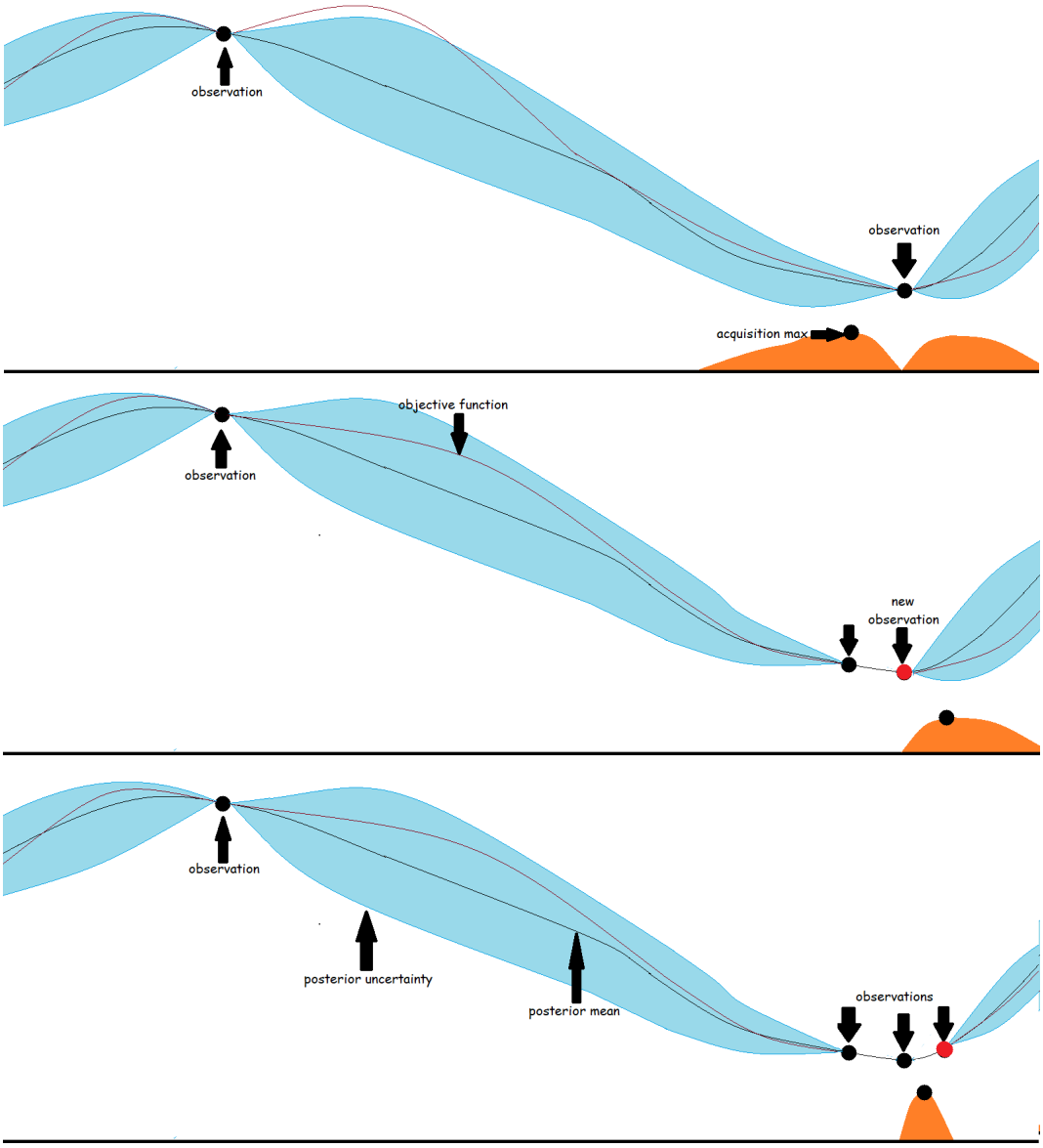
Ο γενικός σχεδιασμός στο Bayesian Optimization δίνεται παρακάτω

Για t από 1 μέχρι badget_threshold:
1) Βρες το επόμενο sampling poit x_t βελτιστοποιώντας την acquisition συνάρτηση:

$$x_t = \arg\max_x a(x|\mathcal{D}_{1:t-1})$$

2) Υπολόγισε την f στο x_t: $y_t = f(x_t)$
3) Βάλε το δείγμα στο σύνολο των προηγούμενων δειγμάτων $\mathcal{D}_{1:t} = \mathcal{D}_{1:t-1} \cup \{(x_t, y_t)\}$ ανανεώνοντας το στατιστικό μοντέλο

Παρακάτω δίνεται ένα παράδειγμα σε μορφή ακολουθίας σχεδίων για το πως εκτελείται η βελτιστοποίηση, όπως στο [8]



Σχήμα 1.6: Στα παραπάνω σχήματα βλέπουμε τα διαδοχικά βήματα που ακολουθούνται κατά τη διαδικασία της μπευζιανής βελτιστοποίησης. Μέσα από αυτή τη διαδικασία, τις διαδοχικές παρατηρήσεις που παίρνουμε και την acquisition function που χρησιμοποιούμε, βλέπουμε ποιοτικά πως μπορούμε να βρεθούμε στο σημείο ελαχιστοποίησης μιας συνάρτησης.

Κεφάλαιο 2

Related Work

Παρακάτω παρουσιάζονται συστήματα ή τρόποι για meta-feature extraction για μη εποπτευόμενη μηχανική μάθηση.

2.1 AutoClust

Το AutoClust [1] είναι ένα AutoML σύστημα για μη-εποπτευόμενη μηχανική μάθηση, το οποίο βασίζεται στην εξαγωγή internal CVIs ως μέτα-χαρακτηριστικά, έπειτα από εφαρμογή του αλγορίθμου συστασοποίησης Mean Shift σε ένα δοθέν σύνολο δεδομένων. Ο αλγόριθμος Mean Shift είναι ένας αλγόριθμος συσταδοποίησης για τον οποίο δεν απαιτείται a priori καθορισμός υπερπαραμέτρων, όπως για παράδειγμα το πλήθος των παραγόμενων συστάδων κτλ. Με τον τρόπο αυτό, το AutoClust επιτυγχάνει να διατηρήσει τον μη-εποπτευόμενο χαρακτήρα που επιδιώκει, καθότι ασχολείται με clustering αλγορίθμους. Πιο συγκεκριμένα τα CVIs που χρησιμοποιώνται στο AutoClust δίνονται στον παρακάτω πίνακα:

MF1	Silhouette Index
MF2	Dunn Index
MF3	C-Index
MF4	Calinski-Harabasz
MF5	Davies Bouldin
MF6	SDbw
MF7	CDBW
MF8	Tau Index
MF9	Ratkowsky Lance
MF10	McClain Rao

Έπειτα από τη σύσταση βέλτιστου αλγορίθμου με χρήση k-nearest neighbor προσέγγισης, το AutoClust ασχολείται και με το πρόβλημα του Hyper-parameter Optimization (HPO), με την εύρεση δηλαδή βέλτιστης παραμετροποίησης για τον αλγόριθμο που έχει επιλεχθεί. Για το σκοπό αυτό επιστρατεύονται νευρωνικά δίκτυα, τα οποία δέχονται ως είσοδο διανύσματα με τα ανωτέρω CVIs κι εκτιμούν τον extrernal CVI ARI. Τέλος, χρησιμοποιώντας Bayesian Optimization και πιο συγκεκριμένα Tree Structured Parzen Estimator, κάνει το optimization μεγιστοποιώντας το εκτιμώμενο ARI.

2.2 Distance Based Meta-Features

Στην εργασία τους [2], οι Ferrari και de Castro χρησιμοποιούν το κανονικοποιημένο διάνυσμα των αποστάσεων μεταξύ των instances προκειμένου μέσω αυτού να εξάγουν τα μέτα-χαρακτηριστικά ενός συνόλου δεδομένων. Πιο συγκεκριμένα, δεδομένου ενός συνόλου δεδομένων D, δημιουργούν το διάνυσμα $d = \frac{(d_{ij})_{ij}}{\| (d_{ij})_{ij} \|}$, όπου $d_{ij} = dist(X_i^{(D)}, X_j^{(D)})$ είναι η ευκλείδεια απόσταση του i-οστού και του j-οστού instance αντίστοιχα. Στη συνέχεια υπολογίζονται τα παρακάτω στατιστικά μέτρα του διανύσματος d και χρησιμοποιώνται ως μέτα-χαρακτηριστικά του συνόλου δεδομένων:

MF1	Mean of d
MF2	Variance of d
MF3	Standard deviation of d
MF4	Skewness of d
MF5	Kurtosis of d
MF6	% of values in the interval [0, 0.1]
MF7	% of values in the interval (0.1, 0.2]
MF8	% of values in the interval (0.2, 0.3]
MF9	% of values in the interval (0.3, 0.4]
MF10	% of values in the interval (0.4, 0.5]
MF11	% of values in the interval (0.5, 0.6]
MF12	% of values in the interval (0.6, 0.7]
MF13	% of values in the interval (0.7, 0.8]
MF14	% of values in the interval (0.8, 0.9]
MF15	% of values in the interval (0.9, 1.0]
MF16	% of values with absolute Z-score in the interval [0,1)
MF17	% of values with absolute Z-score in the interval [1,2)
MF18	% of values with absolute Z-score in the interval [2,3)
MF19	% of values with absolute Z-score in the interval [3,+∞)

Στην εργασία τους επίσης προτείνουν μια νέα μέθοδο συνδυαστικής κατάταξης (ranking combination method) και χρησιμοποιούν k-nearest neighbor μεθοδολογία για τη σύσταση βέλτιστου αλγορίθμου, ενώ συγκρίνουν τα αποτελέσματά τους με τα παρακάτω μέτα-χαρακτηριστικά που προκύπτουν από τον τομέα της θεωρίας πληροφορίας:

MF1	Log2 of the number of objects
MF2	Log2 of the number of attributes
MF3	Percentage of discrete attributes
MF4	Percentage of outliers
MF5	Mean entropy of discrete attributes
MF6	Mean concentration between discrete attributes
MF7	Mean absolute correlation between continuous attributes
MF8	Mean skewness of continuous attributes
MF9	Mean kurtosis of continuous attributes

2.3 AutoML4Clust

Το AutoML4Clust [3] είναι ένα AutoML σύστημα το οποίο βασίζεται σε επαναληπτικούς optimizers. Πιο συγκεκριμένα, δοθέντος ενός συνόλου δεδομένων D, μιας μετρικής M κι ενός budget_threshold L, εκτελείται ένας επαναληπτικός αλγόριθμος βελτιστοποίησης L φορές, προκειμένου να βελτιστοποιηθεί η τιμή της μετρικής M πάνω στο δοθέν σύνολο D. Οι Tschechlov et al. στην εργασία τους εστιάζουν στη βελτιστοποίηση 3 internal CVIs, των Calinski-Harabasz, Davies-Bound Index και Silhouette.

2.4 MARCO-GE

Το MARCO-GE [4] είναι ένα AutoML σύστημα το οποίο βασίζεται στην εξαγωγή μέτα-χαρακτηριστικών με χρήση Graph Convolutional Neural Network (GCNN). Πιο συγκεκριμένα από ένα σύνολο δεδομένων, αφού εφαρμόσουμε πρώτα preprocessing και Principal Component Analysis (PCA), δημιουργούμε ένα γράφημα ομοιότητας (similarity graph) μέσω συνημιτονοειδούς ομοιότητας (cosine similarity). Στη συνέχεια μέσω ενός GCNN φτιάχνουμε έναν classifier ο οποίος βασίζεται στο ranking των αλγορίθμων για τα σύνολα δεδομένων.
Το MARCO-GE ασχολείται και με το HPO βελτιστοποιώντας το γραμμικό συνδυασμό προεπιλεχθέντων internal CVIs χρησιμοποιώντας Bayesian Optimization και πιο συγκεκριμένα Tree Structured Parzen Estimator.

2.5 Dataset2Vec

Το Dataset2Vec [4] είναι μία εργασία η οποία ασχολείται με έναν τρόπο εκμάθησης εξαγωγής μέτα-χαρακτηριστικών μέσω νευρωνικών δικτύων, για προβλήματα εποπτευόμενης μηχανικής μάθησης. Πιο συγκεκριμένα, η βασική ιδέα της εργασίας πηγάζει απ΄ το θεώρημα αναπαράστασης των Kolmogorov κι Arnold, βάσει του οποίου οποιαδήποτε πραγματική συνάρτηση $f : \mathbb{R}^n \to \mathbb{R}$ δέχεται μοναδική αναπαράσταση της μορφής $f(\underline{x}) = f(x_1, \ldots, x_n) = \sum_{q=0}^{2n} \Phi_q \left(\sum_{p=1}^{n} \phi_{q,p}(x_p) \right)$. Οι Jomaa et al. γενικεύοντας το εν λόγω θεώρημα, θεωρούν ότι υπάρχει συνάρτηση $\phi : \{D \in \mathbb{R}^{N \times (M+T)} : N, M, T \in \mathbb{N}\} \to \mathbb{R}^{K_h}$, όπου $K_h \in \mathbb{N}$ η οποία δέχεται μια αναπαράσταση της μορφής

$$\phi(D) = h \left(\frac{1}{M^{(D)} T^{(D)}} \sum_{m=1}^{M^{(D)}} \sum_{t=1}^{T^{(D)}} g \left(\frac{1}{N^{(D)}} \sum_{n=1}^{N^{(D)}} f(X_{n,m}^{(D)}, Y_{n,t}^{(D)}) \right) \right)$$

Στη συγκεκριμένη περίπτωση με D συμβολίζουμε οποιδήποτε σύνολο δεδομένων, συνεπώς η συνάρτηση ϕ χρησιμοποιε΄ται ως εξαγωγέας μέτα-χαρακτηριστικών για το σύνολο δεδομένων D.
Ιδιαίτερα, όπως τονίζεται στην εργασία, σκοπός της είναι να βρεθεί η συσχέτιση των instances ανά feature ($X_{n,m}^{(D)}$) με τα αντίστοιχα target values ($Y_{n,t}^{(D)}$), μέσω της συνάρτησης f και στη συνέχεια μέσω ομαδοποίησης μέσης τιμής (average pooling) όπως φαίνεται και στον παραπάνω τύπο, να τροφοδοτηθούν τα νευρωνικά δίκτυα g κι h. Κατά τον τρόπο αυτό, το μοντέλο επιτυγχάνει να είναι schema agnostic, δηλαδή να αδιαφορεί για το σχήμα των συνόλων δεδομένων (πλήθος των instances και features) και να είναι ικανό να χρησιμοποιηθεί σε μια πληθώρα εφαρμογών εποπτευόμενης μηχανικής μάθησης όπως είναι η ταξινόμηση (classification), η παλινδρόμηση (regression) ανεξαρτήτου πλήθους των target values (multivariate regression) κτλ.
Οι συναρτήσεις f, g κι h παραπάνω, είναι πραγματικές συναρτήσεις $f : \mathbb{R}^2 \to \mathbb{R}^{K_f}$, $g : \mathbb{R}^{K_f} \to \mathbb{R}^{K_g}$ και $h : \mathbb{R}^{K_g} \to \mathbb{R}^{K_h}$ κι εκτιμώνται με τη βοήθεια νευρωνικών δικτύων.

Για την εκπαίδευση την εκτίμηση των συναρτήσεων χρησιμοποιείται ένα μοντέλο ομοιότητας, κατά το οποίο δύο σύνολα D_1 και D_2 δεδομένων θεωρούνται όμοια, αν το D_2 προκύπτει μέσω δειγματοληψίας του D_1. Στη συνέχεια εκπαιδεύεται ένα μέτα-μοντέλο, βάσει του οποίου προσπαθείται η ελαχιστοποίηση της συνάρτησης κόστους

$$-\frac{1}{|P|}\sum_{(x,x',i)\in P}\log(\hat{i}(x,x')) - \frac{1}{|W|}\sum_{(x,x',i)\in W}\log(1-\hat{i}(x,x'))$$

όπου $\hat{i}(x,x') = \exp^{-\gamma\|\phi(x)-\phi(x')\|}$ και $P = \{(x,x',i) \sim p|i=1\}$ το σύνολο των ζευγών των συνόλων δεδομένων τα οποία θεωρούνται όμοια και $W = \{(x,x',i) \sim w|i=0\}$ το σύνολο των ζευγών των συνόλων δεδομένων τα οποία θεωρούνται ανόμοια.

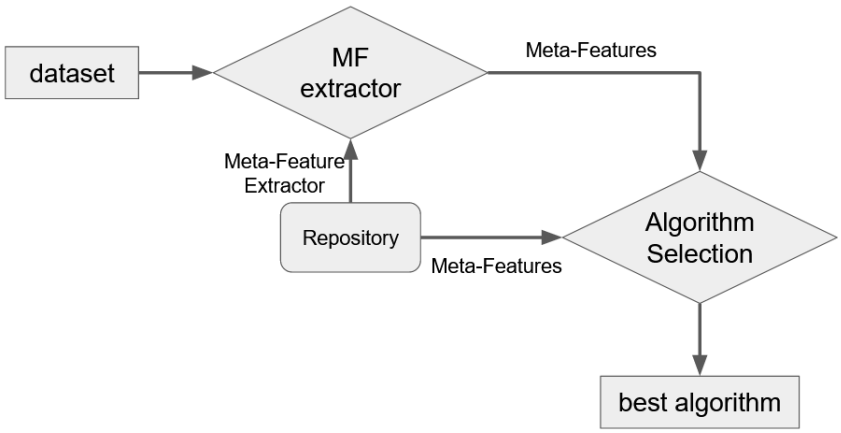
Κεφάλαιο 3

Αρχιτεκτονική

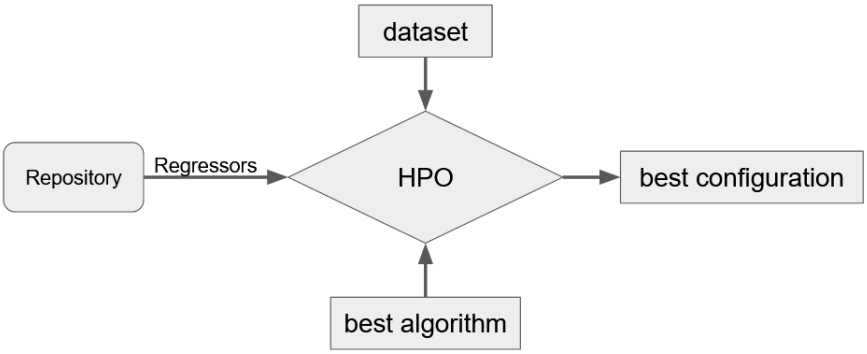
3.1 Σύστημα

Το AutoML σύστημα το οποίο σχεδιάζουμε έχει ως στόχο να λύσει τα προβλήματα του algorithm selection και του Hyper-parameter Optimization και χωρίζεται σε δύο φάσεις, την off-line και την on-line. Για το algorithm selection σχεδιάζουμε έναν meta-feature extractor ο οποίος αποτελεί μια τροποίηση του meta-feature extractor απ΄ το Dataset2Vec [5], ενώ για το Hyper-parameter Optimization επιστρατεύεται ο σχεδιασμός που έγινε για το αντίστοιχο meta-task στο AutoClust [1].
Η τροποποίηση του meta-feature extractor είναι σημαντική, καθότι το σύτημά μας αποτελεί ένα AutoML σύστημα που σκοπό έχει να λύσει προβλήματα συσταδοποίησης. Για να το κάνουμε αυτό, όπως περιγράφεται αναλυτικά παρακάτω, αναδιαμορφώνουμε τον extractor με τρόπο τέτοιο ώστε να μην κάνει χρήση τιμών στόχων (των target values σε προβλήματα παλινδρόμησης ή των τιμών των κλάσεων σε προβλήματα ταξινόμησης). Εν αντιθέσει με τον meta-feature extractor του Dataset2Vec, στον δικό μας meta-feature extractor χρησιμοποιούμε εγγενή πληροφορία των συνόλων δεδομένων (τα instances με τα αντίστοιχα χαρακτηριστικά τους), χωρίς την εξωτερική πληροφορία (ground truth), παρότι εν γένει στη Μηχανική Μάθηση θεωρείται άρρηκτα συνδεδεμένη με τα σύνολα δεδομένων τα οποία απαρτίζουν.
Κατά την off-line φάση εκπαιδεύουμε τον meta-feature extractor και χρησιμοποιούμε τα μέτα-χαρακτηριστικά των συνόλων δεδομένων που παρήχθησαν για να φτιάξουμε ένα αποθετήριο.
Για να γίνει αυτό, έχει προηγηθεί εξαντλητική αναζήτηση (Exhaustive Search) σε κάποια σύνολα δεδομένων προκειμένου να εξαχθούν διάφοροι δείκτες αξιολόγησης των αλγορίθμων και να χρησιμοποιηθούν ως κριτήρια ομοιότητα, όπως περιγράφουμε αναλυτικότερα παρακάτω. Τα ίδια σύνολα εκπαίδευσης στα οποία εκτελούμε εξαντλητική αναζήτηση χρησιμοποιούνται επίσης και για την εκπαίδευση του meta-feature extractor.

Κατά την on-line χρησιμοποιούμε τον meta-feature extractor και το αποθετήριο το οποίο φτιάξαμε στην off-line φάση για να συστήσουμε το βέλτιστο αλγόριθμο σε ένα νέο σύνολο δεδομένων, καθώς επίσης και για να κάνουμε το Hyper-parameter Optimization.



Σχήμα 3.1: Στο παραπάνω σχήμα βλέπουμε τη λογική με την οποία λειτουργεί η σύσταση αλγορίθμου στο AutoML σύστημα το οποίο σχεδιάσαμε κατά την on-line φάση λειτουργίας του.



Σχήμα 3.2: Στο παραπάνω σχήμα βλέπουμε τη λογική με την οποία λειτουργεί το Hyper-parameter Optimization στο AutoML σύστημα το οποίο σχεδιάσαμε κατά την on-line φάση λειτουργίας του.

Για καθένα από τα σύνολα δεδομένων και στις δύο φάσεις (on-line κι off-line), εκτελούμε το παρακάτω preprocessing:
1) Χρησιμοποιούμε label encoding προκειμένου να μετατρέψουμε τις κατηγορικές μεταβλητές σε αριθμητικές.
2) Διώχνουμε απ΄ το σύνολο δεδομένων όσα features έχουν σταθερά μία τιμή, καθότι θεωρούμε ότι αυτά τα features δεν έχουν καμία συνεισφορά στο task του clustering.
3) Διώχνουμε απ΄ το σύνολο δεδομένων εκείνα τα features για τα οποία τουλάχιστον το 30% αυτών λείπουν.
4) Εφαρμόζουμε Min-Max κανονικοποίηση των features στο διάστημα [0,1], προκειμένου να αποφύγουμε προβλήματα που άπτονται του scaling των διαστάσεων ανά feature.
5) Χρησιμοποιούμε KNN imputter προκειμένου να αντικαταστήσουμε τυχόντα missing values χρησιμοποιώντας 3 το πλήθος κοντινούς γείτονες κι ευκλείδεια μετρική.

3.2 Συμβολισμός

Σε αυτή την ενότητα δίνουμε τα πλαίσια του συμβολισμού που χρησιμοποιούμε παρακάτω στον ορισμό του meta-feature extractor, βάσει του συμβολισμού που ορίζεται στο [5].
Τα πινακοειδή (tabular) σύνολα δεδομένων αποτελούν σύνολα δεδομένων στη μορφή δισδιάστατων πινάκων. Οι γραμμές ενός πίνακα καλούνται εγγραφές (instances) κι αποτελούν ξεχωριστές οντότητες ενός συνόλου

δεδομένων, ενώ οι στήλες του καλούνται χαρακτηριστικά (features). Τα χαρακτηριστικά ορίζουν την πληροφορία των instances. Ένα σύνολο δεδομένων είναι αμετάβλητο ως προς οποιαδήποτε μετάθεση τόσο των γραμμών, όσο και των στηλών του. Αυτό σημαίνει ότι το σύνολο δεδομένων που ορίζεται από έναν πίνακα D_1 είναι το ίδιο με ένα σύνολο δεδομένων το οποίο ορίζεται από τον πίνακα D_2 που προκύπτει από οποιαδήποτε μετάθεση των γραμμών ή τον στηλών του D_1. Για παράγειγμα οι παρακάτω δύο πίνακες, ορίζουν το ίδιο σύνολο δεδομένων

$\mathbf{D_1}$	M_1	M_2	M_3
N_1	X_{11}	X_{12}	X_{13}
N_2	X_{21}	X_{22}	X_{23}
N_3	X_{31}	X_{32}	X_{33}
N_4	X_{41}	X_{42}	X_{43}

$\mathbf{D_2}$	M_2	M_3	M_1
N_1	X_{12}	X_{13}	X_{11}
N_4	X_{42}	X_{43}	X_{41}
N_3	X_{32}	X_{33}	X_{31}
N_2	X_{22}	X_{23}	X_{21}

Αντίστοιχα, υποθέτοντας ότι τα σύνολα δεδομένων μας έχουν εξωτερική πληροφορία (όπως στο Dataset2Vec), παραμένει η αμεταβλητότητα ως προς τις μεταθέσεις κι έτσι πάλι οι παρακάτω πίνακες ορίζουν το ίδιο σύνολο δεδομένων

$\mathbf{D_1}$	M_1	M_2	M_3	T_1	T_2
N_1	X_{11}	X_{12}	X_{13}	Y_{11}	Y_{21}
N_2	X_{21}	X_{22}	X_{23}	Y_{12}	Y_{22}
N_3	X_{31}	X_{32}	X_{33}	Y_{13}	Y_{23}
N_4	X_{41}	X_{42}	X_{43}	Y_{14}	Y_{24}

$\mathbf{D_2}$	M_2	M_3	M_1	T_2	T_1
N_1	X_{12}	X_{13}	X_{11}	Y_{12}	Y_{11}
N_4	X_{42}	X_{43}	X_{41}	Y_{42}	Y_{41}
N_3	X_{32}	X_{33}	X_{31}	Y_{32}	Y_{31}
N_2	X_{22}	X_{23}	X_{21}	Y_{22}	Y_{21}

Με βάση τα παραπάνω, μπορούμε να θεωρήσουμε ότι ένα σύνολο δεδομένων D είναι ένα σύνολο από $M^{(D)}$ το πλήθος χαρακτηριστικά και $T^{(D)}$ το πλήθος τιμές στόχους

$$D = \{X_1^{(D)}, \ldots, X_{M^{(D)}}^{(D)}, Y_1^{(D)}, \ldots, Y_{T^{(D)}}^{(D)}\},$$

όπου κάθε μεταβλητή $X_m^{(D)}$ είναι ένα σύνολο $M^{(D)}$ το πλήθος στοιχείων

$$X_m^{(D)} = \{X_{1,m}^{(D)}, \ldots, X_{N^{(D)},m}^{(D)}\},\ m = 1, \ldots, M^{(D)}$$

ενώ κάθε μεταβλητή στόχος $Y_t^{(D)}$ είναι ένα σύνολο $T^{(D)}$ το πλήθος στοιχείων

$$Y_t^{(D)} = \{Y_{1,t}^{(D)}, \ldots, Y_{N^{(D)},t}^{(D)}\},\ t = 1, \ldots, T^{(D)}$$

Χρησιμοποιώντας αυτό τον συμβολισμό που βλέπει ένα σύνολο δεδομένων ως το σύνολο των συνόλων των στοιχείων του, παρακάτω ορίζουμε τον meta-feature extractor της εργασίας μας.

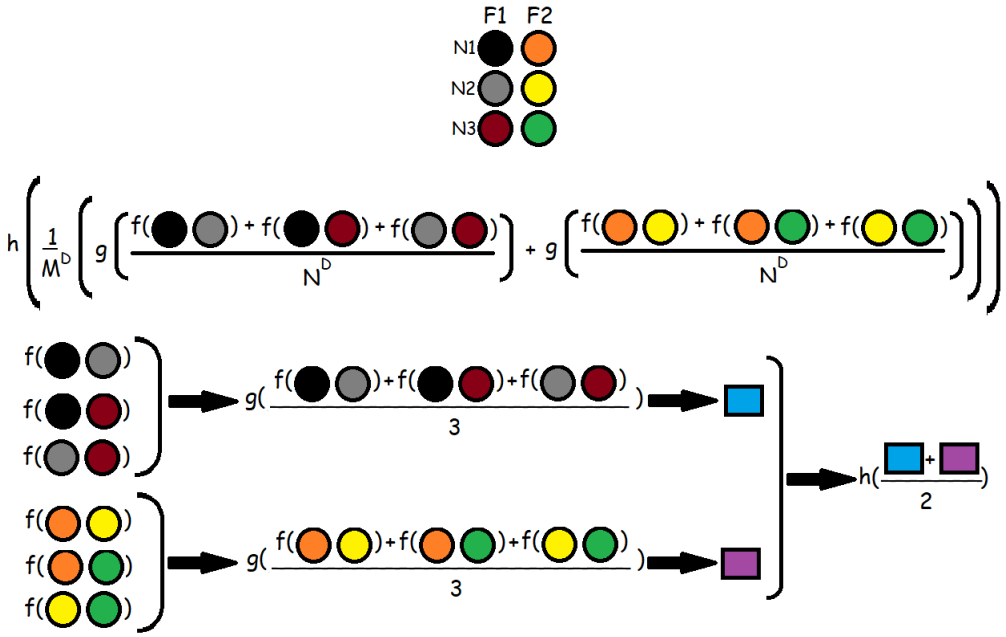
3.3 Δομή του Meta-Feature Extractor

Η παρούσα εργασία στοχεύει στο να διαμορφώσει την αρχιτεκτονική που χρησιμοποιείται στο Dataset2Vec για την εκμάθηση εξαγωγής μέτα-χαρακτηριστικών με χρήση νευρωνικών δικτύων, προκειμένου να μπορεί να χρησιμοποιηθεί σε προβλήματα μη-εποπτευόμενης μηχανικής μάθησης.
Για το σκοπό αυτό επιστρατεύεται η βασική ιδέα πάνω στην οποία στηρίζεται το Dataset2Vec, ότι δηλαδή υπάρχει μια συνάρτηση $\phi : \{D \in \mathbb{R}^{N \times M} : N,\ M \in \mathbb{N}\} \to \mathbb{R}^{K_h}$. Η βασική διαφοροποίηση με το Dataset2Vec είναι ότι δεν γίνεται χρήση δεδομένων τα οποία θεωρούνται ground truth όπως είναι τα target values με σκοπό να μπορεί να χρησιμοποιηθεί ως εξαγωγέας μέτα-χαρακτηριστικών σε AutoML συστήματα τα οποία ασχολούνται με προβλήματα μη-εποπτευόμενης μηχανικής μάθησης και πιο συγκεκριμένα συσταδοποίησης.

Για το λόγο αυτό, αντί να προσδοκούμε μέσω της συνάρτησης f να βρούμε τη συσχέτιση μεταξύ των $X_{n,m}^{(D)}$ και των αντίστοιχων $Y_{n,t}^{(D)}$, προσδοκούμε να βρούμε τη συσχέτιση των $X_{i,m}^{(D)},\ X_{j,m}^{(D)}$ για $i \neq j$.
Πιο συγκεκριμένα, θεωρούμε τη συνάρτηση

$$\phi(D) = h\left(\frac{1}{M^{(D)}}\sum_{m=1}^{M^{(D)}} g\left(\frac{2}{N^{(D)}\cdot(N^{(D)}-1)}\sum_{i,j=1\wedge i\neq j}^{N^{(D)}} f(X_{i,m}^{(D)}, X_{j,m}^{(D)})\right)\right)$$

όπου $f : \mathbb{R}^2 \to \mathbb{R}^{K_f}$, $g : \mathbb{R}^{K_f} \to \mathbb{R}^{K_g}$ και $h : \mathbb{R}^{K_g} \to \mathbb{R}^{K_h}$. Όπως και στο Dataset2Vec έτσι κι εδώ οι συναρτήσεις f, g κι h εκτιμώνται μέσω νευρωνικών δικτύων.
Θέλοντας να δούμε ακόμα πιο αναλυτικά το πως λειτουργεί το Column-wise Average Pooling μοντέλο, μπορούμε να δούμε ένα αναλυτικό παράδειγμα. Έστω ότι έχουμε ένα dataset D με 3 instances ($N^{(D)} = 3$) και 2 features ($M^{(D)} = 2$).



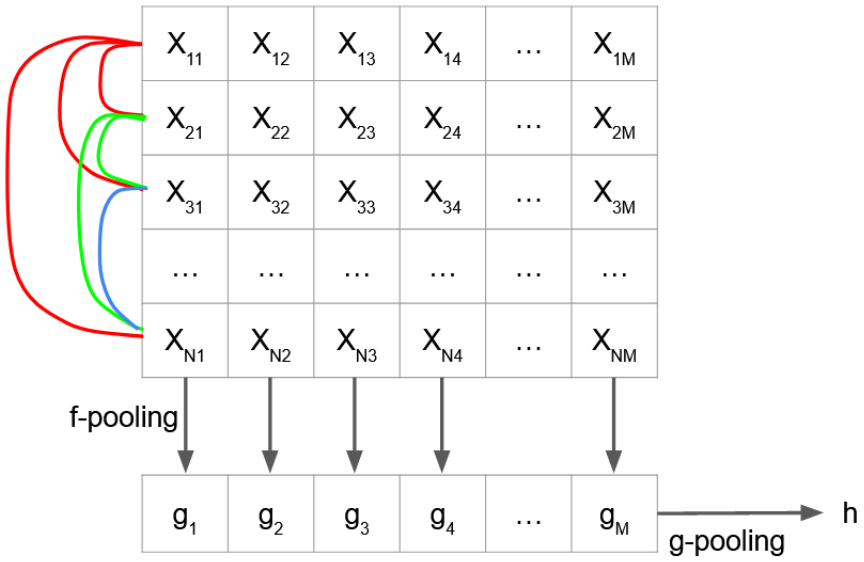
Σχήμα 3.3: Στο παραπάνω παράδειγμα βλέπουμε τα διαδιχικά βήματα εξαγωγής του τελικού διανύσματος των μετα-χαρακτηριστικών από ένα σύνολο δεδομένων με 3 instances και 2 features. Αρχικά χρησιμοποιούμε τη συνάρτηση f για τα ζεύγη των διαφορετικών instances για τα features. Στη συνέχεια με το μέσο όρο (ή το άθροισμα) που είναι ένα $\mathbb{R}^{K_f}$ διάνυσμα, τροφοδοτούμε τη συνάρτηση g και τέλος με την ίδια διαδικασία (με ένα $\mathbb{R}^{K_g}$ διάνυσμα) τροφοδοτούμε την h παίρνοντας το τελικό $\mathbb{R}^{K_h}$ διάνυσμα των μετα-χαρακτηριστικών

Την παραπάνω δομή στο εξής θα την ονομάζουμε Column-wise Average Pooling (CW-AP).

Ως εναλλακτική της CW-AP αρχιτεκτονικής, μπορούμε να δούμε την Column-wise Cumulative Pooling (CW-CP) που δίνεται απ΄ τη δομή

$$\phi(D) = h\left(\sum_{m=1}^{M^{(D)}} g\left(\sum_{i,j=1\wedge i\neq j}^{N^{(D)}} f(X_{i,m}^{(D)}, X_{j,m}^{(D)})\right)\right)$$

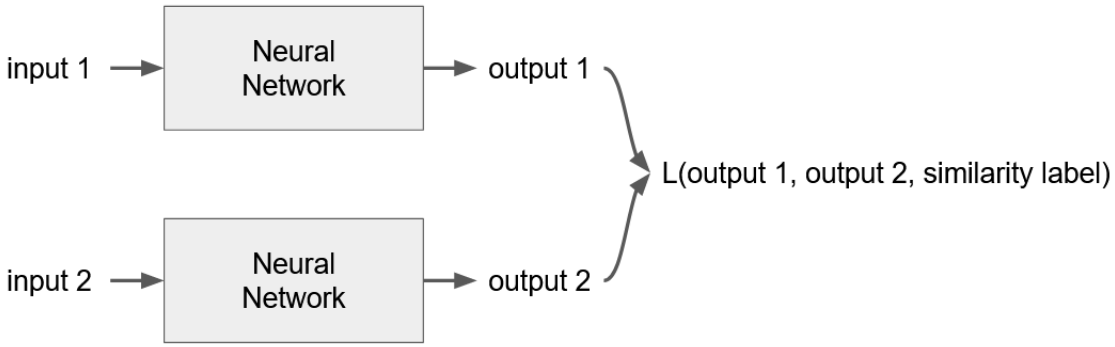
Όπως γίνεται σαφές, η διαφορά των CW-AP και CW-CP έγκειται στο γεγονός ότι στην πρώτη περίπτωση η ομαδοποίηση (pooling) γίνεται με χρήση της μέσης τιμής, ενώ στη δεύτερη με αθροιστικό τρόπο.



Σχήμα 3.4: Ροή εργασιών για την εξαγωγή μέτα-χαρακτηριστικών μέσω Column-wise Pooling

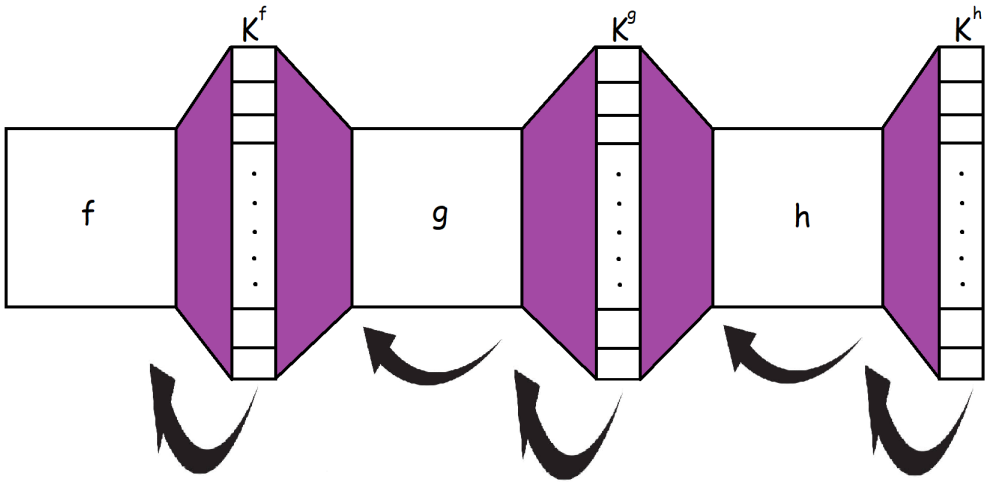
3.4 Εκπαίδευση του Meta-Feature Extractor

Για την εκπαίδευση του meta-feature extractor θα θεωρήσουμε ότι έχουμε ένα πρόβλημα ομοιότητας. Ως εκ τούτου θα χρησιμοποιήσουμε την αρχιτεκτονική των Σιαμαίων Νευρωνικών Δικτύων (Siamese Neural Networks). Τα Σιαμαία Νευρωνικά Δίκτυα χρησιμοποιώνται ευρέως στα προβλήματα μάθησης ομοιότητας (similarity learning). Παρακάτω δίνεται σχηματικά η βασική τους αρχιτεκτονική



Σχήμα 3.5: Αρχιτεκτονική ενός Σιαμαίου νευρωνικού δικτύου

Όπως φαίνεται παραπάνω ένα Σιαμαίο Νευρωνικό Δίκτυο πρόκειται για ταυτόχρονο forward pass δύο διαφορετικών inputs από το ίδιο νευρωνικό δίκτυο. Κατ΄ αυτό τον τρόπο παράγονται δύο διαφορετικά outputs τα οποία στη συνέχεια τροφοδοτούν μια συνάρτηση κόστους L.
Στην περίπτωση του meta-feature extractor που θέλουμε να εκπαιδεύσουμε, το νευρωνικό δίκτυο θα είναι η συστοιχία των νευρωνικών δικτύων των συναρτήσεων f, g κι h όπως ορίστηκαν παραπάνω. Τα δίκτυα των συναρτήσεων θα τα θεωρήσουμε ως ακολουθιακά (sequential), προκειμένου κατά την εκπαίδευση να μπορούμε να εφαρμόσουμε backpropagation για την ανανέωση των βαρών που απαρτίζουν τα δίκτυα.



Σχήμα 3.6: Ακολουθιακή ροή του backpropagation

Θα προσεγγίσουμε το πρόβλημα της ομοιότητας με τον εξής τρόπο:

Έστω δύο σύνολα δεδομένων D_1 και D_2, τότε τα D_1 και D_2 είναι όμοια αν και μόνο αν ο βέλτιστος αλγόριθμος για το D_1 είναι ο βέλτιστος αλγόριθμος για το D_2.

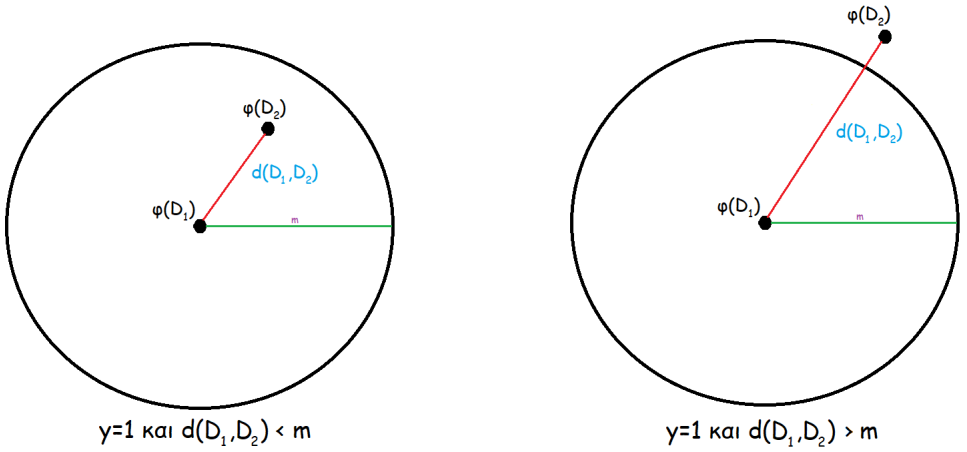
Η βασική συνάρτηση κόστους την οποία χρησιμοποιούμε είναι η Contrastive Loss [7] η οποία περιγράφεται αναλυτικά απ΄ τον παρακάτω τύπο

$$\mathcal{L} = \sum_{i \neq j} L(D_i, D_j)$$

όπου

$$L(D_i, D_j) = (1 - y) \cdot \frac{1}{2} \cdot d(D_i, D_j)^2 + y \cdot \frac{1}{2} \cdot (\max(0, m - d(D_i, D_j)))^2,$$

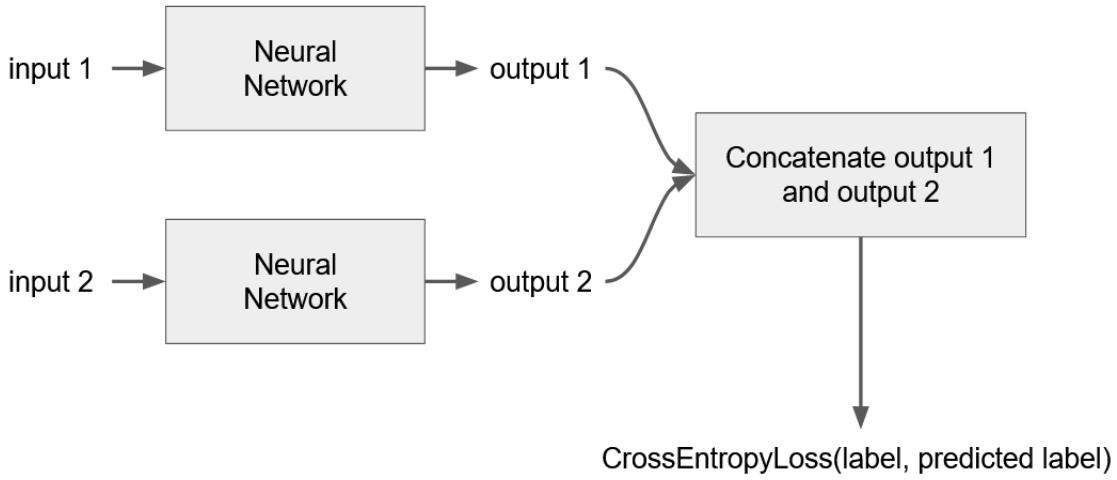
$d(D_i, D_j) = \|\phi(D_i) - \phi(D_j)\|_2$, y είναι η ετικέτα ομοιότητας (0, αν τα D_i και D_j είναι όμοια και 1 αν είναι ανόμοια) και m είναι μια tunable υπερπαράμετρος.Όπως φαίνεται απ΄ τον ορισμό της, η Contrastive Loss κάνει χρήση της απόστασης των representation vectors $\phi(D_i)$ και $\phi(D_j)$ γι΄ αυτό και χαρακτηρίζεται ως μετρική ομοιότητας απόστασης (distance metric). Αν τα D_i και D_j είναι όμοια τότε $y = 0$ και η συνάρτηση κόστους παίρνει την τιμή του μισού του τετραγώνου της απόστασης των representation vectors. Αν τα D_i και D_j είναι ανόμοια, τότε $y = 1$ κι η συνάρτηση κόστους παίρνει την τιμή του μισού του τετραγώνου του μέγιστου μεταξύ του 0 και του $m - d(D_i, D_j)$. Αυτό σημαίνει ότι η συνάρτηση κόστους παίρνει τιμή διαφορετική του μηδενός, αν τα D_i και D_j είναι ανόμοια, μόνο την περίπτωση που η απόσταση των representation vectors είναι μικρότερη της τιμής του m και στην περίπτωση όπου χρησιμοποιούμε ευκλείδεια μετρική όπως παραπάνω, αυτό συμβαίνει αν και μόνο αν το $\phi(D_j)$ βρίσκεται εντός της ανοικτής μπάλας με κέντρο το $\phi(D_i)$ κι ακτίνα m. Ένα τέτοιο παράδειγμα δίνεται στο παρακάτω σχήμα



Σχήμα 3.7: Οπτικοποίηση του margin της Contrastive Loss

Αυτό σημαίνει ότι η παράμετρος m ορίζει τον βαθμό στον οποίο θα κάνουμε penalize αναπαραστάσεις ανόμοιων συνόλων δεδομένων. Όσο μικρότερο είναι το m τόσο μικρότερο είναι και το περιθώριο που έχουν αναπαραστάσεις ανόμοιων συνόλων δεδομένων (σε μορφή διανυσμάτων) για να είναι κοντά.

Μια άλλη συνάρτηση κόστους την οποία μπορούμε να χρησιμοποιήσουμε είναι η Cross Entropy Loss. Η Cross Entropy Loss χρησιμοποιείται σε προβλήματα ταξινόμησης κι ως εκ τούτου θα χρειαστεί να τροποποιήσουμε το Σιαμαίο Νευρωνικό Δίκτυο με τον παρακάτω τρόπο:



Σχήμα 3.8: Εκπαίδευση του σιαμαίου νευρωνικού με Cross Entropy Loss

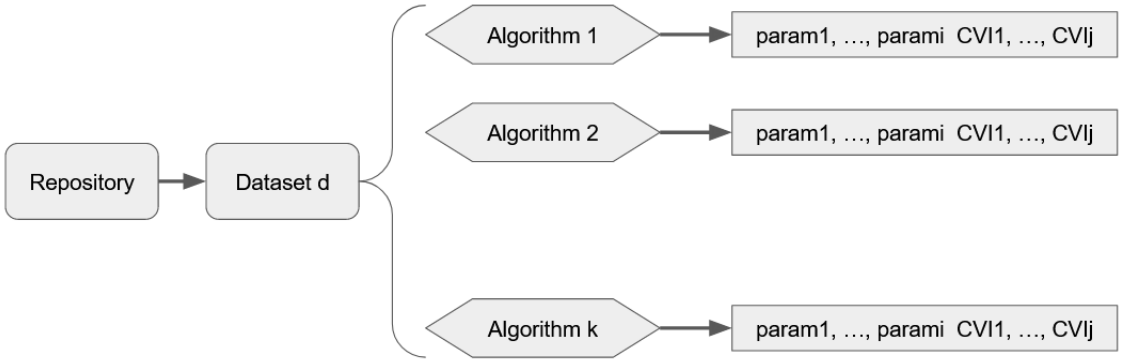
Παίρνοντας τα δύο outputs από τα inputs μας, τα συνενώνουμε και τροφοδοτούμε ένα fully connected δίκτυο το οποίο χρησιμοποιούμε για τον χαρακτηρισμό των inputs ως όμοια ή ανόμοια.
Η Cross Entropy Loss δίνεται απ΄ τον παρακάτω τύπο

$$\mathcal{L} = -\frac{1}{|class0|}\sum_{i=1}^{|class0|} y_i \cdot log\hat{y}_i - \frac{1}{|class1|}\sum_{i=1}^{|class1|} (1-y_i)\cdot log(1-\hat{y}_i)$$

όπου $|classi|$ είναι το πλήθος των ζευγών που ανήκουν στην κλάση i, προκείμενου να αξιοποιήσουμε τα βάρη των κλάσεων που έχουμε στη διάθεσή μας.

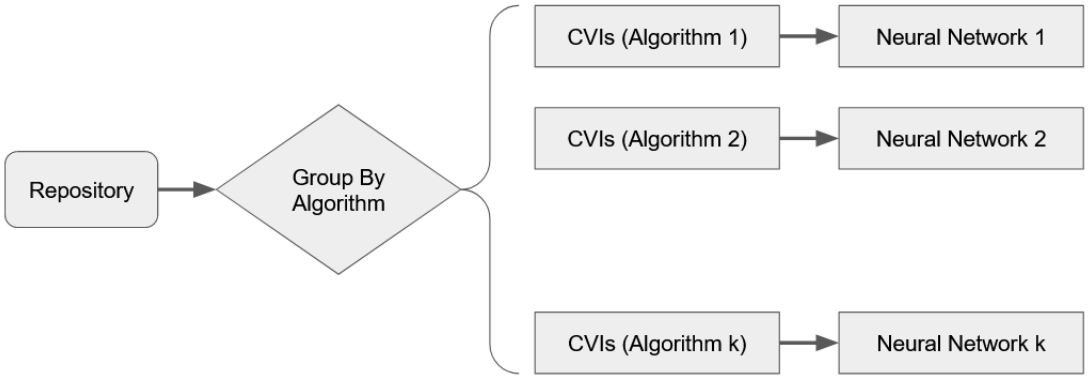
3.5 Hyperparameter Optimization

Για το HPO θα χρησιμοποιήσουμε την προσέγγιση που χρησιμοποιείται στο AutoClust. Κατά την off-line φάση εκτελούμε εξαντλητική αναζήτηση στα σύνολα δεδομένων που έχουμε στο repository για τους διάφορους αλγορίθμους τους οποίους εξετάζει το σύστημά μας. Σε αυτή την εξαντλητική αναζήτηση για κάθε αλγόριθμο εξετάζουμε διαφορετικές παραμετροποιήσεις αυτού και παίρνουμε μετρήσεις για διάφορα internal κι extrenal Cluster Validity Indices (CVIs).



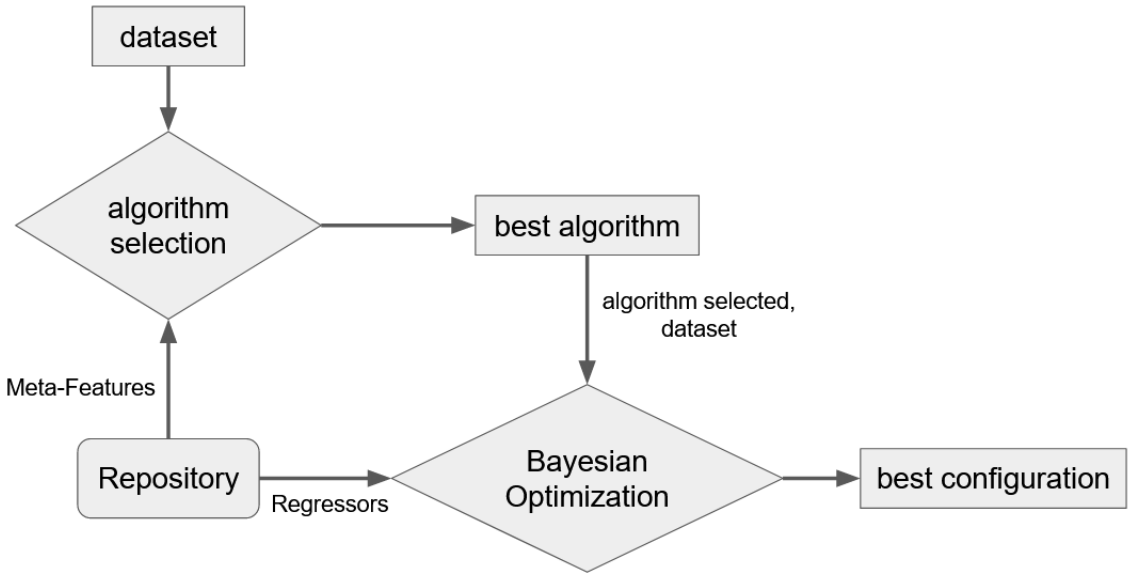
Σχήμα 3.9: Κατά την off-line φάση, έχοντας σύνολα δεδομένων στη διάθεσή μας, αφού τα περάσουμε απ΄ το preprocessing που περιγράφεται παραπάνω, εφαρμόζουμε εξαντλητική αναζήτηση για τους διάφορους αλγορίθμους που εξετάζουμε. Αυτό σημαίνει ότι για κάθε αλγόριθμος, παίρνουμε μετρήσεις για τα CVIs που επιλέγουμε για τις διάφορες παραμετροποιήσεις των αλγορίθμων.

Στη συνέχεια, κατά την off-line φάση, επιλέγουμε κάποιους CVIs ως features και για κάθε έναν απ΄ τους k αλγορίθμους εκπαιδεύουμε ένα νευρωνικό δίκτυο βάσει αυτών, χρησιμοποιώντας τα σύνολα δεδομένων που έχουν ως βέλτιστο τον αντίστοιχο αλγόριθμο, προκειμένου να εκτιμήσουμε τον external CVI ARI.



Σχήμα 3.10: Κατά την off-line φάση, για τους διάφορους αλγορίθμους που έχουμε στη διάθεσή μας, εκπαιδεύουμε νευρωνικά δίκτυα ως regressors των οποίων τα inputs είναι ένα διάνυσμα εκ των προτέρων προσδιορισμένων CVIs και τα outputs είναι ένα CVI που παίζει το ρόλο της τιμής στόχου. Κατ΄ αυτό τον τρόπο προσδοκούμε να βρούμε τις συσχετίσεις των inputs CVIs με την τιμή στόχο την οποία θέλουμε στα πλαίσια του HPO να βελτιστοποιήσουμε.

Τέλος, κατά την on-line φάση, όταν έρχεται ένα καινούργιο σύνολο δεδομένων κι έχουμε επιλέξει βέλτιστο αλγόριθμο, έστω τον b, για αυτό κατά τη διαδικασία του algorithm selection, εφαρμόζουμε Bayesian Optimization με τη βοήθεια του αντίστοιχου νευρωνικού δικτύου NN_b για την εύρεση βέλτιστης παραμετροποίησης του b.



Σχήμα 3.11: Στο παραπάνω σχήμα βλέπουμε τη ροή εργασιών του AutoML συστήματος κατά την on-line φάση λειτουργίας του, χρησιμοποιώντας μπευζιανή βελτιστοποίηση.

Το σύστημά μας προσφέρει τη δυνατότητα χρήσης του bayesian optimization με χρήση του Tree-structured Parzen Estimator (TPE) μέσω της βιβλιοθήκης Optuna της Python ή optimization με Random Search.

Κεφάλαιο 4

Πειραματισμός κι Αποτελέσματα

Για τον πειραματισμό μας χρησιμοποιήσαμε 85 σύνολα δεδομένων με πηγές τα Kaggle, UCI Machine Learning Repository και OpenML, εξετάζοντας την απόδοση των συνόλων δεδομένων αυτών για 8 clustering αλγορίθμους, τους Agglomerative Clustering, Affinity Propagation, K-Means, OPTICS, Spectral Clustering, BIRCH, DBSCAN και Mean Shift.Για λόγους πληρότητας, οφείλουμε να δούμε τους αλγορίθμους αυτούς ως αντικείμενα διαφορετικών κατηγοριών. Πιο συγκεκριμένα οι παραπάνω αλγόριθμοι χωρίζονται σε 4 κατηγορίες

· Αλγόριθμοι Hierarchical Clustering (HC)
· Αλγόριθμοι Centroid-Based Clustering (CBC)
· Αλγόριθμοι Density-Based Clustering (DBC)
· Αλγόριθμοι Graph-Based Clustering (GBC)

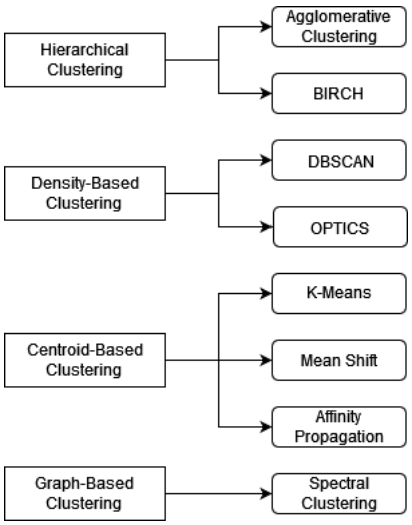
Hierarchical Clustering
Η ιεραρχική συσταδοποίηση αναφέρεται στη συσταδοποίηση μέσω δημιουργίας ενός ιεραρχικού δενδροδιαγράμματος. Τα φύλλα του συγκεκριμένου δένδρου αναφέρονται στα instances ενός συνόλου δεδομένων και κάθε επόμενο κλαδί του δένδρου, αναφέρεται στα παραγόμενα clusters που προκύπτουν.

Centroid-Based Clustering
Στο Centroid-Based Clustering βασιζόμαστε στην παραγωγή των συστάδων διά αντιπροσώπου, που ονομάζονται κεντροειδή (centroids), τα οποία τις χαρακτηρίζουν και παράγονται απ΄ τα instances που ανήκουν στα εν λόγο clusters. Τα κεντροειδή σε αυτούς τους αλγορίθμους δεν ανήκουν αναγκαστικά στις συστάδες τις οποίες αντιπροσωπεύουν.

Density-Based Clustering
Οι Density-Based Clustering αλγόριθμοι βασίζονται, όπως μαρτυρά και το όνομά τους, στην εύρεση συστάδων βάσει της πυκνότητας των σημείων (instances) στο χώρο στον οποίο αυτά ανήκουν. Η ουσιαστική διαφορά των DBC με τους HC και CBC που αναφέρονται παραπάνω, είναι ότι δεν χρειάζεται ο εκ των προτέρων προσδιορισμός του πλήθους των παραγόμενων συστάδων, αλλά ο προσδιορισμός παραμέτρων που αφορούν την πυκνότητα.

Graph-Based Clustering
Ως Graph-Based Clustering αλγορίθμους, αναφερόμαστε σε αυτούς που προκύτουν χρησιμοποιώντας γραφήματα. Η χρήση γραφημάτων είναι ιδιαίτερα εύστοχη, καθώς τα γραφήματα χρησιμοποιούνται ευρέως και μπορούν να αποτυπώσουν άριστα τη συσχέτιση μεταξύ των instances ενός συνόλου δεδομένων.

Παρακάτω δίνεται η αντιστοίχιση των αλγορίθμων που αναφέρθηκαν παραπάνω, με την κατηγορία στην οποία ανήκουν



Σχήμα 4.1: Κατηγοριοποίηση αλγορίθμων συσταδοποίησης

Αρχικά εκτελέστηκε εξαντλητική αναζήτηση (exhaustive search) στους παραπάνω αλγορίθμους για διαφορετικές τιμές των υπερπαραμέτρων τους κι απ΄ τα προκύπτοντα clusters πήραμε μετρήσεις για διάφορους δείκτες συσταδοποίησης. Οι δείκτες αυτοί είναι είτε εσωτερικοί (internal clustering validity indices) είτε εξωτερικοί (external clustering validity indices). Οι internal δείκτες αφορούν πληροφορία η οποία πηγάζει αποκλειστικά και μόνο απ΄ τα παραγόμενα clusters, ενώ οι external δείκτες κάνουν χρήση του ground truth το οποίο αφορά τα σύνολα δεδομένων μας.
Ο κυριότερος δείκτης στον οποίο εστιάζουμε είναι το Adjusted Rand Index (ARI). Το ARI μετράει την ομοιότητα των παραγόμενων clusters και των κλάσεων του συνόλου δεδομένων μας. Παρακάτω δίνουμε τον ορισμό του:
Έστω $\mathcal{X} = \{X_1, \ldots, X_s\}, \mathcal{Y} = \{Y_1, \ldots, Y_k\}$ δύο διαμερίσεις ενός συνόλου $\mathcal{S}$. Ορίζοντας ως $n_{ij} = |X_i \cap Y_j|$ μπορούμε να φτιάξουμε τον παρακάτω πίνακα συνάφειας (contingency matrix)

$\mathcal{X}, \mathcal{Y}$	Y_1	Y_2	$\ldots$	Y_k	sums
X_1	n_{11}	n_{12}	$\ldots$	n_{1k}	a_1
X_2	n_{21}	n_{22}	$\ldots$	n_{2k}	a_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
X_s	n_{s1}	n_{s2}	$\ldots$	n_{sk}	a_s
sums	b_1	b_2	$\ldots$	b_k	

Τότε ορίζουμε το ARI ως

$$\text{ARI} = \frac{\sum_{ij} \binom{n_{ij}}{2} - [\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}]/\binom{n}{2}}{\frac{1}{2}[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}] - [\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}]/\binom{n}{2}}$$

Στη δική μας περίπτωση, τα $\mathcal{X}$ και $\mathcal{Y}$ είναι τα labels των παραγόμενων clusters και τα labels των κλάσεων των συνόλων δεδομένων.

Έπειτα απ΄ την εξαντλητική αναζήτηση θεωρούμε την πραμετροποίηση κάθε αλγορίθμου η οποία μας δίνει το μέγιστο ARI ανά αλγόριθμο και μέσω αυτής βγάζουμε τη διάταξη (ranking) των αλγορίθμων.
Εξαιτίας της μη καλής αντιπροσώπευσης των κλάσεων βέλτιστων αλγορίθμων επιλέξαμε να κρατήσουμε τους Agglomerative Clustering, K-Means, Spectral Clustering και DBSCAN προκειμένου να έχουμε αντιπροσώπευση από όλο το φάσμα των κατηγοριών. Τότε παίρνουμε την παρακάτω κατανομή, χωρίζοντας παράλληλα τα σύνολα δεδομένων σε δύο κατηγορίες, για εκπαίδευση των μοντέλων και για testing:

	General	Training	Testing
Spectral Clustering	51	40	11
Agglomerative Clustering	16	11	5
DBSCAN	11	9	2
K-Means	7	5	2

Πίνακας 4.1: Κατανομή αλγορίθμων

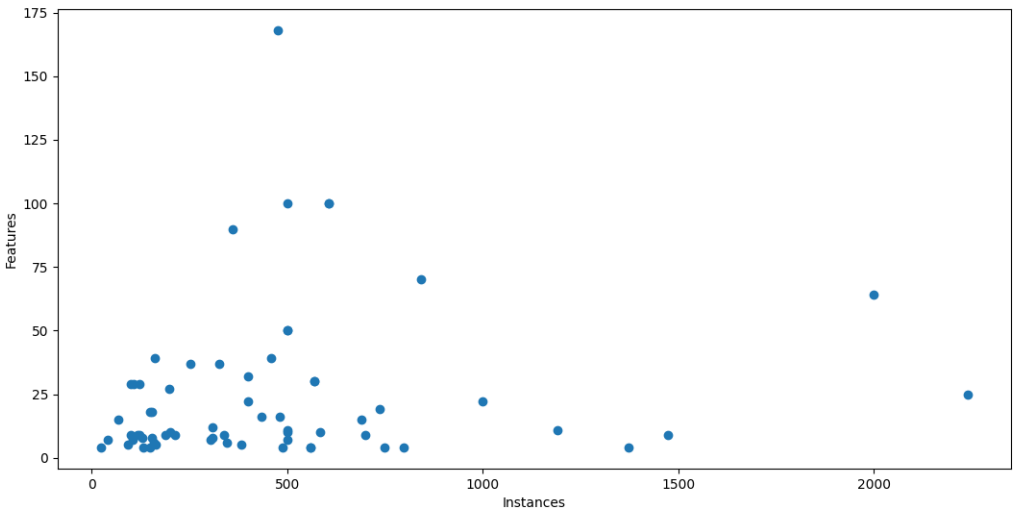
Τα training datasets είναι τα παρακάτω:

analcatdata_authorship, analcatdata_dmft, analcatdata_negotiation, analcatdata_wildcat, appendicitis, ar1, ar4, ar6, arsenic-female-lung, arsenic-male-bladder, autoUniv-au7-500, blood transfusion, bolts, Breast Cancer Wisconsin (Diagnostic), Breast Cancer Wisconsin (Prognostic), breast-cancer-coimbra, breast-cancer-wisconsin, bupa, calendarDOW, CastMetal1, Chronic KIdney Disease, clean1, cleveland-nominal, collins, Congressional Voting Records, Contraceptive Method Choice, corral, credit-approval, data_banknote_authentication, data_student, dataset_194_eucalyptus, datatrieve, diggle_table_a2, Engine1, ESL, fertility_Diagnosis, fl2000, forest-type-mapping-training, fri_c2_500_50, fri_c3_500_10, fri_c3_500_50, fri_c4_500_100, glass, grub-damage, hayes-roth, heart-hungarian, heart-switzerland, heart-va, hepatitis, Hill_Valley_without_noise_Testing, Hill_Valley_without_noise_Training, indian-liver-patients, iris, kc3, LED-display-domain-7digit, lenses, libras, lithium-ion batteries, lowbwt, Lower Back Pain Symptoms, lymphography, marketing_campaign, mc2, meanWhile1, mfeat-karhunen

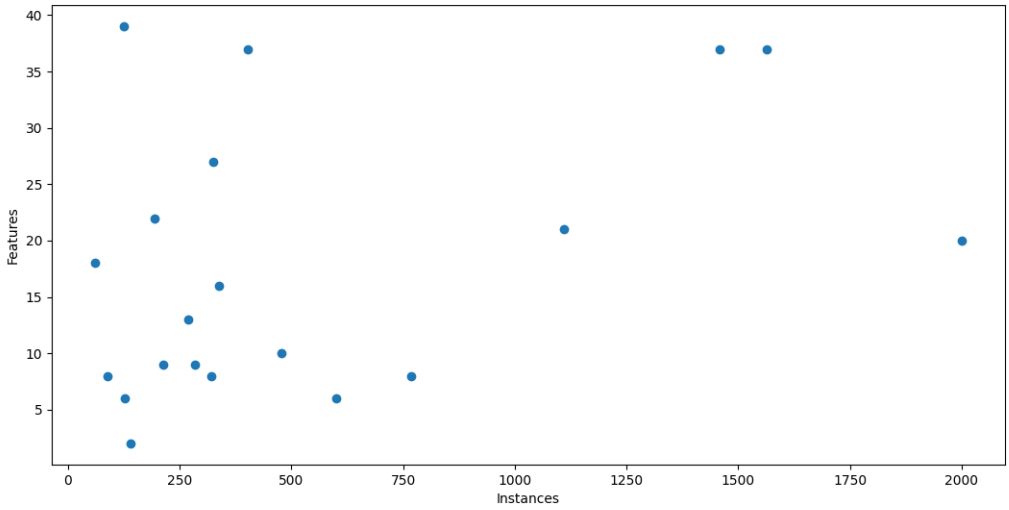
Ενώ ως testing datasets χρησιμοποιήθηκαν τα παρακάτω:

mobile_price_classification_train-train, pima, MindCave2, monks-problems-2, mu284, mux6, mw1, newton_hema, pc1, pc1_req, pc3, pc4, PopularKids, prnn_fglass, prnn_viruses, forest-type-mapping-testing, heart-statlog, parkinsons, post-operative, primary-tumor

Στα παρακάτω scatter plots δίνεται η εικόνα της διάστασης των συνόλων δεδομένω που χρησιμοποιήθηκαν, με βάση το πλήθος των instances και το πλήθος των features καθενός εξ αυτών



Σχήμα 4.2: Διαστάσεις των training συνόλων δεδομένων. Ως σημεία στο γράφημα παρουσιάζονται τα σύνολα δεδομένων. Στον οριζόντιο άξονα παρουσιάζονται το πλήθος των instances του κάθε συνόλου, ενώ στον κατακόρυφο το πλήθος των χαρακτηριστικών.



Σχήμα 4.3: Διαστάσεις των testing συνόλων δεδομένων. Ως σημεία στο γράφημα παρουσιάζονται τα σύνολα δεδομένων. Στον οριζόντιο άξονα παρουσιάζονται το πλήθος των instances του κάθε συνόλου, ενώ στον κατακόρυφο το πλήθος των χαρακτηριστικών.

Παρακάτω δίνεται η περιγραφή των αλγορίθμων που εξετάζουμε.

Agglomerative Clustering

Ο αλγόριθμος Agglomerative Clustering εφαρμόζει "από τα κάτω" ιεραρχική συσταδοποίηση. Αυτό σημαίνει ότι το δενδρόγραμμα ιεράρχισης ξεκινά από τα φύλλα, τα οποία αποτελούν κάθε instance του συνόλου δεδομένων. Σε κάθε επόμενο επίπεδο του δένδρου εκτελείται συσταδοποίηση ανά instance, προσθέτοντας σε κάθε προηγούμενη συστάδα εκείνο το instance το οποίο απέχει ελάχιστη απόσταση, βάσει κάποιας μετρικής.

K-Means

Ο K-Means είναι ένας επαναληπτικός centroid-based αλγόριθμος συσταδοποίησης. Τα βήματα που ακολουθούμε στον K-Means δίνονται παρακάτω

1. Ορίζουμε το πλήθος των k παραγόμενων συστάδων
2. Αρχικοποιούμε τα κεντροειδή, επιλέγοντας τυχαία k instances του συνόλου δεδομένων μας.
3. Για κάθε instance υπολογίζουμε την απόστασή του από όλα τα κεντροειδή και το αναθέτουμε στη συστάδα εκείνη απ΄ της οποίας το κεντροειδές απέχει ελάχιστη απόσταση.
4. Υπολογίζουμε τα νέα κεντροειδή των συστάδων ως τη μέση τιμή των instances τα οποία ανήκουν στη συγκεκριμένη συστάδα.
5. Επαναλαμβάνουμε τα βήματα 3 και 4 μέχρις ότου τα κεντροειδή της επόμενης επανάληψης να απέχουν απόσταση μικρότερη ή ίση ενός δεδομένου threshold απόστασης, από αυτά της προηγούμενης επανάληψης.

DBSCAN

Ο DBSCAN (Density Based Spatial Clustering of Application with Noise) εφαρμόζει συσταδοποίηση με βάση την πυκνότητα. Για τον DBSCAN ένα cluster ορίζεται ως μια περιοχή υψηλής πυκνότητας. Βασικά στοιχεία που ορίζουν την έννοια της πυκνότητας είναι οι πραάμετροι min_samples και eps. Το min_samples ορίζει το πλήθος των instances που μπορούν να ορίσουν ένα κεντρικό σημείο (core sample), ενώ το eps ορίζει μέσω της απόστασης την γειτονιά των κεντρικών σημείων. Κατ΄ αυτό τον τρόπο ο DBSCAN είναι ικανός στο να αναγνωρίσει θόρυβο (noise) στα δεδομένα μας, ορίζοντάς τον ως μια ξεχωριστή συστάδα.

Spectral Clustering

Ο αλγόριθμος Spectral χρησιμοποιεί ένα γράφημα ομοιότητας προκειμένου να εφαρμόσει συσταδοποίηση. Μπορούμε να θεωρήσουμε ότι ο αλγόριθμος εκτελείται με βάση τα παρακάτω βήματα:
1. Δημιουργούμε ένα γράφημα ομοιότητας για τα instances του συνόλου δεδομένων μας.
2. Υπολογίζουμε τα k πρώτα ιδιοδιανύσματα του Λαπλασιανού πίνακα του γραφήματος (τα ιδιοδιανύσματα δηλαδή, τα οποία ανήκουν στις k μικρότερες ιδιοτιμές του πίνακα).
3. Χρησιμοποιώντας τα k ιδιοδιανύσματα που βρήκαμε στο βήμα 2, μετασχηματίζουμε τα δεδομένα μας, εφαρμόζοντας ουσιαστικά μείωση των διαστάσεων του συνόλου δεδομένων μας.
4. Εφαρμόζουμε στο μετασχηματισμένο σύνολο δεδομένων του βήματος 3 έναν αλγόριθμο συσταδοποίησης, όπως για παράδειγμα τον K-Means.

Κατά τη διαδικασία της εξαντλητικής αναζήτησης των υπερπαραμέτρων των ανωτέρω αλγορίθμων, έγινε χρήση της βιβλιοθήκης scikit-learn της Python. Οι υπερπαράμετροι οι οποίοι εξετάστηκαν για κάθε αλγόριθμο δίνονται παρακάτω

Hyper-paramater	Agglomerative Clustering	K-Means
1	n_clusters$\in [2, 20]$	n_clusters$\in$[2,20]
2	affinity$\in$ {euclidean, l1, l2, manhattan, cosine}	tol$\in$[0.00001,0.0002]
3	linkage$\in$ {ward, complete, average, single}	algorithm$\in$ {full, elkan}

Hyper-paramater	DBSCAN	Spectral Clustering
1	p$\in \{1, 2\}$	eigen_solver=amg
2	eps$\in$[0.001,1]	n_clusters$\in$[2,20]
3	metric$\in$ {euclidean, cosine}	n_components$\in$[2,10]
4		affinity=nearest_neighbors
5		n_neighbors$\in$[1,10]
6		assign_labels$\in$ {kmeans, discretize}

Πίνακας 4.2: Χώρος υπερπαραμέτρων των προς εξέταση αλγορίθμων.

4.1 Δομή δικτύων

Κατά την εκπαίδευση δοκιμάστηκαν οι παρακάτω αρχιτεκτονικές εκπαίδευσης δικτύων των συναρτήσεων f, g κι h για το Column-wise Cummulative Pooling μοντέλο

D2V	Fully Connected
f	Dense(8);Dense(8);Dense(8)
g	Dense(16);Dense(16);Dense(8)
h	Dense(8);Dense(8);Dense(8)
Fully Connected - CE	Dense(16);Dense(16);Dense(8)

Πίνακας 4.3: Αρχιτεκτονική των δικτύων για τις συναρτήσεις f, g και h που χρησιμοποιήθηκαν για τον πειραματισμό.

Παντού χρησιμοποιήθηκε ως activation functions η ReLU (Rectified Linear Unit) η οποία δίνεται απ' τον τύπο

$$\mathrm{ReLU(x)} = \mathrm{(x)^+} = \mathrm{max(0,x)}$$

Η εκπαίδευση έγινε με Mini-batch training χρησιμοποιώντας batches των 100 ζευγών με shuffled ζεύγη ανά epoch, για 10 epochs κι ως optimizer έγινε χρήση του SGD (Stochastic Gradient Descent optimizer) της PyTorch με learning rate=10^{-5}.

4.2 Αποτελέσματα

4.2.1 Algorithm Selection

Αρχικά όσον αφορά την αξιολόγηση του algorithm selection, θα συγκρίνουμε το average ranking των αλγορίθμων που προκύπτουν από εκτέλεση ενός k-Nearest Neighbor query. Αυτό σημαίνει ότι θεωρώντας το διάνυσμα των μέτα χαρακτηριστικών ενός συνόλου δεδομένων από τα σύνολα που χρησιμοποιούμε για testing, χρησιμοποιώντας τη συνήθη ευκλείδεια μετρική, βρίσκουμε τα k-πλησιέστερα διανύσματα μέτα-χαρακτηριστικών από το σύνολο των datasets που χρησιμοποιήσαμε για την εκπαίδευση του μοντέλου μας. Χρησιμοποιώντας τα rankings των γειτόνων αυτών, βγάζουμε το μέσο ranking των αλγορίθμων. Έτσι, αν x_i=(AgglomerativeClustering=a_i, KMeans=b_i, SpectralClustering=c_i, DBSCAN=d_i) είναι το διάνυσμα κατάταξης των αλγορίθμων, τότε το average ranking για k γείτονες, δίνεται ως

$$\text{average\_ranking}=\frac{1}{k}\sum_{i=1}^{k} x_i$$

Χρησιμοποιώντας το διάνυσμα του average ranking, μπορούμε να αξιολογήσουμε τη δυνατότητα του μοντέλου μας να αναπαράγει το ranking των αλγορίθμων. Χρησιμοποιούμε ως μέτρο το συντελεστή συσχέτισης κατάταξης του Spearman (Spearman's Rank Correlation, SRC) που ορίζεται ως

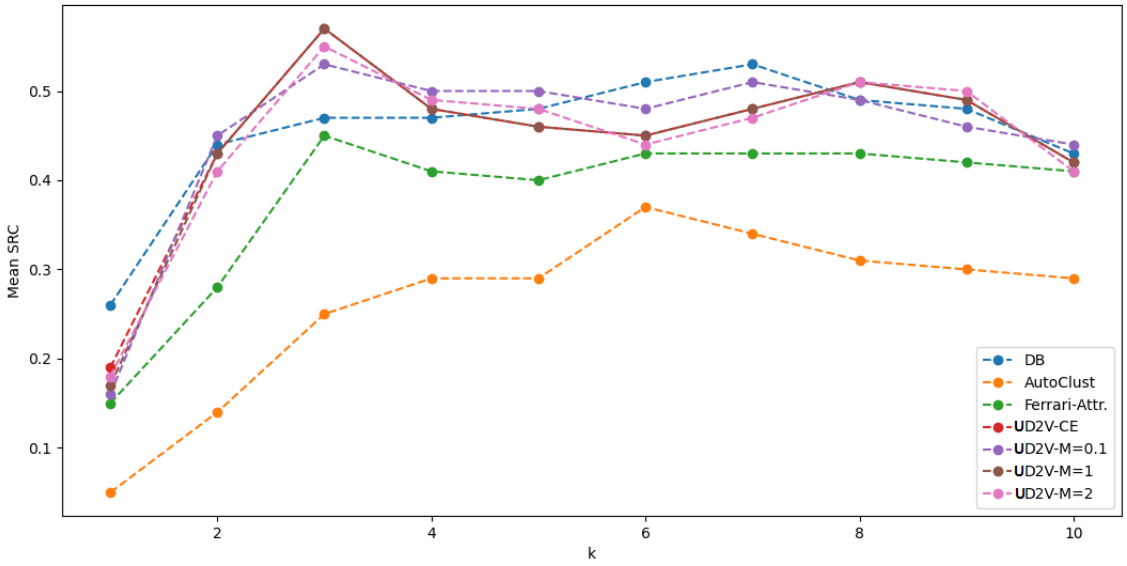
$$SRC = 1 - \frac{6\sum_i s_i^2}{n(n^2-1)}$$

όπου s_i είναι η διαφορά των παρατηρήσεων ανά διάσταση, δηλαδή $a_1 - a_2,\ b_1 - b_2,\ c_1 - c_2$ και $d_1 - d_2$, αντίστοιχα, και n είναι το πλήθος της διάστασης των παρατηρήσεων, όπου στην περίπτωσή μας ισούται με 4. Ο δείκτης SRC παίρνει τιμές στο διάστημα [-1, 1], όπου μεγαλύτερες τιμές σημαίνουν μεγαλύτερη συσχέτιση (η τιμή 1 ισοδυναμεί με την ταύτιση των κατατάξων), μικρότερες (αρνητικές) τιμές υποδεικνύουν αρνητική συσχέτιση κατατάξεων, ενώ η τιμή 0 υποδηλώνει ότι οι κατατάξεις των αλγορίθμων είναι ασυσχέτιστες.
Για διάφορες τιμές του k λοιπόν, παίρνουμε το average ranking για κάθε σύνολο δεδομένων που χρησιμοποιούμε για testing και το συγκρίνουμε μέσω του SRC με το πραγματικό ranking των αλγορίθμων του. Στη συνέχεια παίρνουμε τη μέση τιμή των τιμών του SRC για τα σύνολα δεδομένων και την παρουσιάζουμε στον παρακάτω πίνακα, χρησιμοποιώντας διάφορα μέτα-χαρακτηριστικά στο query. Πιο συγκεκριμένα χρησιμοποιούμε τα Distance-based (DB) μέτα-χαρακτηριστικά (Ferrari et al.), τα Attributes (Ferrari Attributes) μέτα-χαρακτηριστικά (Ferrari et al.), τα μέτα-χαρακτηριστικά του AutoClust και τα μέτα-χαρακτηριστικά του μοντέλου μας, όπως αυτά προέκυψαν εκπαιδεύοντάς το με Cross Entropy (UD2V-CE) και Contrastive Loss (UD2V-M) για διάφορες τιμές της παραμέτρου M (margin). Λόγω αδυναμίας υπολογισμού ορισμένων δεικτών κατά την εφαρμογή του αλγορίθμου Mean Shift και ιδιαίτερα του SDbw, για το AutoClust χρησιμοποιήθηκαν 45 σύνολα δεδομένων στην off-line φάση, 17 σύνολα δεδομένων στην on-line φάση κι όλοι οι δείκτες εκτός του SDbw για τους πειραματισμούς μας.

Mean SRC±StD\k	1	2	3	4	5
DB	**0.26**±0.46	**0.44**±0.42	0.47±0.33	0.47±0.39	0.48±0.41
AutoClust	0.05±0.59	0.14±0.54	0.25±0.53	0.29±0.49	0.29±0.50
Ferrari Attributes	0.15±0.54	0.28±0.50	0.45±0.51	0.41±0.48	0.40±0.52
UD2V-CE	0.19±0.51	0.43±0.39	**0.57**±0.26	0.48±0.34	0.46±0.41
UD2V-M=0.1	0.18±0.49	0.41±0.41	**0.57**±0.26	**0.50**±0.34	**0.49**±0.41
UD2V-M=1	0.17±0.48	0.43±0.39	**0.57**±0.26	0.48±0.43	0.46±0.41
UD2V-M=2	0.18±0.49	0.41±0.39	0.55±0.28	0.49±0.33	0.48±0.41

Mean SRC±StD\k	6	7	8	9	10
DB	**0.51**±0.45	**0.53**±0.47	0.49±0.45	0.48±0.42	**0.43**±0.43
AutoClust	0.37±0.46	0.34±0.52	0.31±0.48	0.30±0.51	0.29±0.54
Ferrari Attributes	0.43±0.47	0.43±0.43	0.43±0.47	0.42±0.48	0.41±0.47
UD2V-CE	0.45±0.46	0.48±0.41	0.51±0.42	0.49±0.42	0.42±0.44
UD2V-M=0.1	0.47±0.47	0.50±0.43	**0.52**±0.43	0.47±0.41	0.42±0.44
UD2V-M=1	0.45±0.46	0.48±0.41	0.51±0.42	0.49±0.42	0.42±0.44
UD2V-M=2	0.44±0.45	0.47±0.41	0.51±0.42	**0.50**±0.41	0.41±0.41

Πίνακας 4.4: Στον παραπάνω πίνακα βλέπουμε το μέσο SRC±Std για τα διάφορα μετα-χαρακτηριστικά που χρησιμοποιήθηκαν στα k-nn queries. Οι διάφορες στήλες αντιπροσωπεύουν διαφορετικά k στα queries.



Σχήμα 4.4: Στον παραπάνω πίνακα βλέπουμε το μέσο SRC για κάθε σύνολο μετα-χαρακτηριστικών για τις διάφορες τιμές του k στο k-nn query. Στον οριζόντιο άξονα υπάρχουν οι διάφορες τιμές του k ενώ στον κατακόρυφο οι τιμές του μέσου SRC. Βλέπουμε ότι για τα UD2V μετα-χαρακτηριστικά πιάνουμε μέγιστο SRC για k=3 και για k μεγαλύτερο του 4, το μέσο SRC φαίνεται να σταθροποιείται. Από τους βασικούς ανταγωνιστές, τα distance-based μετα-χαρακτηριστικά φαίνεται να είναι ελαφρώς αυξανόμενο μέχρι k=7 κι έπειτα φθίνει, χωρίς να φτάνει ποτέ τη μέγιστη τιμή των UD2V μετα-χαρακτηριστικών.

Δεδομένης της μεγάλης τυπικής απόκλισης του μέσου SRC δεν μπορούμε να βγάλουμε ασφαλή συμπεράσματα με αυτό ως μέτρο σύγκρισης των μετα-χαρακτηριστικών, παρόλαυτά φαίνεται ότι για k=1 τα Distance-based μετα-χαρακτηριστικά υπερτερούν όλων των υπολοίπων, ενώ για k=3 τα μετα-χαρακτηριστικά που προέκυψαν απ΄ το UD2V για κάθε τρόπο εκπαίδευσης υπερτερούν της μέγιστης τιμής που πήραμε για

το μέσο SRC όλων των υπολοίπων, ανεξάρτητα του k.

Παρακάτω δίνεται αναλυτικά η σύγκριση για το SRC για k=3 χρησιμοποιώντας τα Distance-based μέτα-χαρακτηριστικά και τα UD2V-CE μετα-χαρακτηριστικά, αντίστοιχα.

Testing Dataset	SRC-DB	SRC-UD2V-CE	p-value-DB	p-value-UD2V-CE
mobile_price_classification_train-train	0.4	0.4	0.6	0.6
pima	0.894427	**1.0**	0.105573	**0.0**
MindCave2	0.4	0.4	0.6	0.6
monks-problems-2	0.2	0.4	0.8	0.6
mu284	0.2	0.2	0.8	0.8
mux6	0.948683	0.948683	0.051317	0.051317
mw1	0.632456	0.632456	0.367544	0.367544
newton_hema	0.948683	0.948683	0.051317	0.051317
pc1	-0.2	**0.316228**	0.8	**0.683772**
pc1_req	0.210819	0.210819	0.789181	0.789181
pc3	0.737865	0.737865	0.262135	0.262135
pc4	0.737865	0.737865	0.262135	0.262135
PopularKids	**1.0**	0.8	**0.0**	0.2
prnn_fglass	0.316228	0.316228	0.683772	0.683772
prnn_viruses	0.4	0.4	0.6	0.6
forest-type-mapping-testing	0.105409	**0.737865**	0.894591	0.262135
heart-statlog	0.105409	**0.737865**	0.894591	**0.262135**
parkinsons	0.8	0.8	0.2	0.2
post-operative	0.316228	0.316228	0.683772	0.683772
primary-tumor	0.4	0.4	0.6	0.6

Πίνακας 4.5: Αποτελέσματα για το SRC χρησιμοποιώντας 3-nn query στα testing datasets.

Παρατηρούμε ότι για k=3 το μοντέλο UD2V-CE δίνει καλύτερο SRC 4 στις 20 φορές σε σχέση με τη χρήση Distance-based μετα-χαρακτηριστικών, ενώ τα Distance-based μετα-χαρακτηριστικά μόλις 1 στις 20. Η επισήμανση για k=3 γίνεται, καθότι γι΄ αυτή την τιμή του k παρατηρείται μέγιστη διαφορά μεταξύ των μέσων τιμών του SRC μεταξύ των κύριων ανταγωνιστών.

Στη συνέχεια θα συγκρίνουμε το Mean Reciprocal Rank (MRR), το οποίο μετρά την κατάταξη του βέλτιστου συνιστώμενου αλγορίθμου

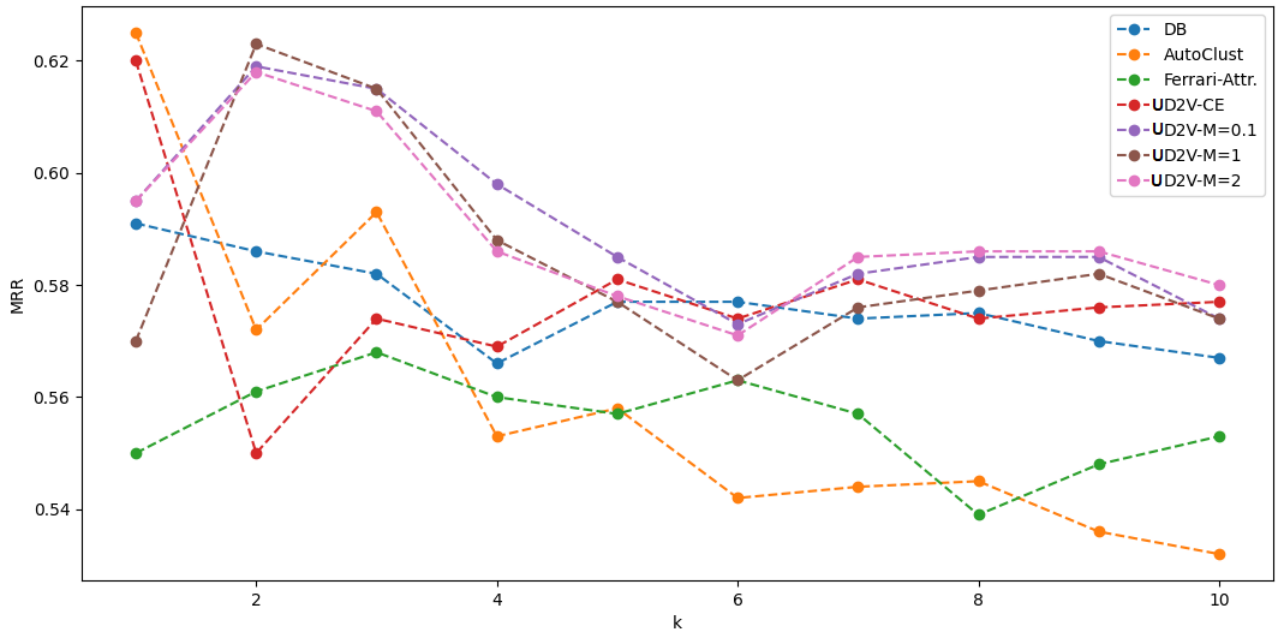
$$MRR = \frac{1}{|D|} \sum_{i \in D} \frac{1}{rank_i}$$

όπου D είναι τα σύνολα δεδομένων τα οποία εξετάζουμε και $rank_i$ είναι το average ranking του αλγορίθμου με τη βέλτιστη επίδοση για το i-οστό σύνολο δεδομένων. Το MRR παίρνει τιμές στο διάστημα (0,1], όπου MRR=1 σημαίνει ότι ο αντίστοιχος βέλτιστος αλγόριθμος συστήνεται ως τέτοιος από το σύστημά μας για κάθε σύνολο δεδομένων.

Meta-Features\k	1	2	3	4	5	6	7	8	9	10
DB	0.591	0.586	0.582	0.566	0.577	**0.577**	0.574	0.575	0.570	0.567
AutoClust	**0.625**	0.572	0.593	0.553	0.558	0.542	0.544	0.545	0.536	0.532
Ferrari Atributes	0.55	0.561	0.568	0.560	0.557	0.563	0.557	0.539	0.548	0.553
UD2V-CE	0.620	0.550	0.574	0.569	0.581	0.574	0.581	0.574	0.576	0.577
UD2V-M=0.1	0.595	0.619	**0.615**	**0.598**	**0.585**	0.573	0.582	0.585	0.585	0.574
UD2V-M=1	0.570	**0.623**	**0.615**	0.588	0.577	0.563	0.576	0.579	0.582	0.574
UD2V-M=2	0.595	0.618	0.611	0.586	0.578	0.571	**0.585**	**0.586**	**0.586**	**0.580**

Πίνακας 4.6: Στον παραπάνω πίνακα βλέπουμε το MRR για τα διάφορα μετα-χαρακτηριστικά που χρησιμοποιήσαμε στα k-nn queries. Οι διάφορες στήλες αντιπροσωπεύουν διαφορετικά k στα queries.

Παρατηρούμε ότι ο meta-feature extractor UD2V-M=1 για k=2 παράγει το βέλτιστο MRR μεταξύ των UD2V meta-feature extractors. Συνολικά βλέπουμε ότι ο UD2V-M=2 κι UD2V-M=0.1 είναι καλύτερος των βασικών ανταγωνιστών του 8 στις 10 φορές, ο UD2V-M=1 7 στις 10 (1 φορά ισόβαθμος με τον DB) ενώ ο UD2V-CE 5 στις 10 φορές.



Σχήμα 4.5: Στον παραπάνω πίνακα βλέπουμε το MRR που παίρνουμε για κάθε σύνολο μετα-χαρακτηριστικών για τις διάφορες τιμές του k στο k-nn query. Στον οριζόντιο άξονα υπάρχουν οι διάφορες τιμές του k ενώ στον κατακόρυφο άξονα οι τιμές του MRR. Βλέπουμε ότι το MRR των UD2V μετα-χαρακτηριστικών έχει μέγιστη τιμή για k=2 με μεγάλη διαφορά από τους ανταγωνιστές τους. Η μέγιστη τιμή που παίρνουμε είναι για k=1 με τα AutoClust μετα-χαρακτηριστικά, με μικρή διαφορά απ΄ τις βέλτιστες μετρήσεις που παίρνουμε για τα UD2V μετα-χαρακτηριστικά. Για k μεγαλύτερο του 3, βλέπουμε ότι το MRR των μετα-χαρακτηριστικών μας αρχίζει να φθίνει και φαίνεται ότι συγκλίνει με το MRR των ανταγωνιστών, όντας πάντα οριακά καλύτερο αυτών.

Έπειτα συνεχίζουμε μετρώντας τα σύνολα δεδομένων στα οποία ο βέλτιστος αλγόριθμος δίνεται ως πρώτη ή ως δεύτερη επιλογή στο ranking των αλγορίθμων μέσω της knn προσέγγισης.

Meta-Features\k	1	2	3	4	5	6	7	8	9	10	Mean
DB	7	**11**	**11**	**11**	**11**	**11**	**11**	**11**	**11**	**11**	**10.6**
AutoClust	7	6	10	9	8	9	9	9	9	9	8.5
Ferrari Atributes	5	10	**11**	10	**11**	10	**11**	**11**	**11**	**11**	10.1
UD2V-CE	7	**11**	**11**	**11**	**11**	**11**	**11**	**11**	**11**	**11**	**10.6**
UD2V-M=0.1	7	**11**	**11**	**11**	**11**	**11**	**11**	**11**	**11**	**11**	**10.6**
UD2V-M=1	6	**11**	**11**	**11**	**11**	**11**	**11**	**11**	**11**	**11**	10.5
UD2V-M=2	7	10	**11**	**11**	**11**	**11**	**11**	**11**	**11**	**11**	10.5

Πίνακας 4.7: Στον παραπάνω πίνακα βλέπουμε την επιλογή του πραγματικού βέλτιστου αλγορίθμου στο top-1 των αλγορίθμων που συστήνονται. Αυτό σημαίνει ότι ο πραγματικός βέλτιστος αλγόριθμος είναι ο βέλτιστος αλγόριθμος που συστήνεται απ΄ τα k-nn queries.

Meta-Features\k	1	2	3	4	5	6	7	8	9	10	Mean
DB	**14**	**13**	17	16	**18**	**16**	16	**16**	**16**	**16**	15.8
AutoClust	12	12	13	11	12	13	13	12	11	11	12
Ferrari Atributes	12	**13**	17	**18**	17	15	16	**16**	15	**16**	15.5
UD2V-CE	13	**13**	**19**	16	17	**16**	16	**16**	**16**	**16**	15.8
UD2V-M=0.1	13	12	**19**	16	17	**16**	**17**	**16**	**16**	**16**	**15.9**
UD2V-M=1	13	**13**	**19**	16	17	**16**	16	**16**	**16**	**16**	15.8
UD2V-M=2	13	**13**	18	15	**18**	**16**	16	**16**	**16**	**16**	15.7

Πίνακας 4.8: Στον παραπάνω πίνακα βλέπουμε την επιλογή του πραγματικού βέλτιστου αλγορίθμου στο top-2 των αλγορίθμων που συστήνονται. Αυτό σημαίνει ότι ο πραγματικός βέλτιστος αλγόριθμος είναι ο πρώτος ή ο δεύτερος αλγόριθμος που συστήνεται απ΄ τα k-nn queries. Οι γραμμές αντιπροσωπεύουν τα διάφορα μετα-χαρακτηριστικά που χρησιμοποιήθηκαν για τον πειραματισμό, ενώ οι στήλες τα διάφορα k στα αντίστοιχα queries.

Βλέπουμε ότι τα μετα-χαρακτηριστικά του UD2V-M=0.1 δίνουν κατά μέσο όρο τα καλύτερα αποτελέσματα στους top-2 αλγορίθμους με τα UD2V-CE, UD2V-M=0.1 και UD2V-M=1 για k=3 να βρίσκουν τον βέλτιστο αλγόριθμο στο top-2 19 στις 20 φορές.
Στην top-1 αξιολόγηση τα περισσότερα μέτα-χαρακτηριστικά φαίνεται ότι αποδίδουν σχεδόν το ίδιο καλά, με τα Distance-based, UD2V-M=0.1 και UD2V-M=2 να βρίσκουν κατά μέσο όρο το ίδιο πλήθος top-1 αλγορίθμων. Αυτό ενδεχομένως να οφείλεται στην ικανότητα των μέτα-χαρακτηριστικών να βρίσκουν εύκολα ως βέλτιστο τον Spectral Clustering για τον οποίο έχουμε 11/20 αντιπροσώπευση στα testing datasets.

Παρακάτω δίνονται τα αποτελέσματα που πάρθηκαν με χρήση διάφορων meta-learners. Πιο συγκεκριμένα οι μετρήσεις αφορούν Random Forest Classifier (RFC), Ada Boost Classifier (ABC), Gradient Boosting Classifier (GBC), Histogram Gradient Boosting Classifier (HCBC), K-Nearest Neighbors Classifier (KnnC) για k=5 και Multilayer Perceptron (MLP) 5×Dense(32)-ReLU. Μετράμε την ακρίβεια ταξινόμησης, τα macro F1 score και το weighted F1 score καθότι η εκπαίδευση των meta-learners έγινε με τις ανομοιογενείς κλάσεις Best Algorithm.
Το F_1 score είναι ο αρμονικός μέσος των precision και του recall

$$F_1 = \frac{2}{recall^{-1} + precision^{-1}} = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

όπου τα precision και recall υπολογίζονται απ΄ τον confusion matrix μέσω των τύπων

$$precision = \frac{True\ Positive}{True\ Positive + False\ Positive},\ recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

Το macro F_1 score είναι είναι ο αριθμητικός μέσος των F_1 scores των επιμέρους κλάσεων, δηλαδή στην περίπτωσή μας

$$F_1^{macro} = \frac{F_1^1 + F_1^2 + F_1^3 + F_1^4}{4}$$

Αντίστοιχα το weighted F_1 score δίνεται ως το άθροισμα των F_1 scores για κάθε κλάση, λαμβάνοντας υπόψιν και το βάρος της κλάσης, δηλαδή στη δική μας περίπτωση δίνεται ως

$$F_1^{weighted} = weight_1 \cdot F_1^1 + weight_2 \cdot F_1^2 + weight_3 \cdot F_1^3 + weight_4 \cdot F_1^4$$

Παρατηρήθηκε ότι για όλους τους meta-learners τα UD2V μέτα-χαρακτηριστικά δίνουν όλα τα ίδια αποτελέσματα. Ως εκ τούτου συνοψίζονται όλα μαζί στον παρακάτω πίνακα

UD2V-CE	RFC	ABC	GBC	HGBC	KnnC	MLP
Train Accuracy	0.92	0.4	0.92	0.66	0.63	0.67
Train F1-Macro	0.88	0.32	0.88	0.35	0.43	0.51
Train F1-Weighted	0.92	0.41	0.92	0.56	0.57	0.64
Test Accuracy	**0.5**	0.2	**0.45**	**0.6**	0.55	0.45
Test F1-Macro	**0.24**	0.11	**0.22**	**0.26**	0.24	0.17
Test F1-Weighted	**0.46**	0.19	**0.43**	**0.48**	0.47	0.38

Πίνακας 4.9: Στον παραπάνω πίνακα βλέπουμε τις μετρικές για τους διάφορους meta-learners χρησιμοποιώντας UD2V-CE μετα-χαρακτηριστικά.

UD2V-M=0.1	RFC	ABC	GBC	HGBC	KnnC	MLP
Train Accuracy	0.92	0.4	0.92	0.66	0.63	0.75
Train F1-Macro	0.88	0.32	0.88	0.35	0.43	0.57
Train F1-Weighted	0.92	0.41	0.92	0.56	0.57	0.7
Test Accuracy	0.45	0.2	**0.45**	**0.6**	0.55	**0.55**
Test F1-Macro	0.22	0.11	**0.22**	**0.26**	0.24	**0.26**
Test F1-Weighted	0.43	0.19	**0.43**	**0.48**	0.47	**0.49**

Πίνακας 4.10: Στον παραπάνω πίνακα βλέπουμε τις μετρικές για τους διάφορους meta-learners χρησιμοποιώντας UD2V-M=0.1 μετα-χαρακτηριστικά.

UD2V-M=1	RFC	ABC	GBC	HGBC	KnnC	MLP
Train Accuracy	0.92	0.4	0.92	0.66	0.63	0.66
Train F1-Macro	0.88	0.32	0.88	0.35	0.43	0.5
Train F1-Weighted	0.92	0.41	0.92	0.56	0.57	0.63
Test Accuracy	0.45	0.2	**0.45**	**0.6**	0.55	0.45
Test F1-Macro	0.22	0.11	**0.22**	**0.26**	0.24	0.18
Test F1-Weighted	0.43	0.19	**0.43**	**0.48**	0.47	0.41

Πίνακας 4.11: Στον παραπάνω πίνακα βλέπουμε τις μετρικές για τους διάφορους meta-learners χρησιμοποιώντας UD2V-M=1 μετα-χαρακτηριστικά.

UD2V-M=2	RFC	ABC	GBC	HGBC	KnnC	MLP
Train Accuracy	0.92	0.4	0.92	0.66	0.66	0.7
Train F1-Macro	0.88	0.32	0.88	0.35	0.51	0.56
Train F1-Weighted	0.92	0.41	0.92	0.56	0.62	0.69
Test Accuracy	0.45	0.2	**0.45**	**0.6**	0.55	0.5
Test F1-Macro	0.22	0.11	**0.22**	**0.26**	0.36	0.25
Test F1-Weighted	0.43	0.19	**0.43**	**0.48**	0.50	0.47

Πίνακας 4.12: Στον παραπάνω πίνακα βλέπουμε τις μετρικές για τους διάφορους meta-learners χρησιμοποιώντας UD2V-M=2 μετα-χαρακτηριστικά.

Distance-based	RFC	ABC	GBC	HGBC	KnnC	MLP
Train Accuracy	1	0.64	1	0.9	0.61	0.4
Train F1-Macro	1	0.38	1	0.89	0.4	0.17
Train F1-Weighted	1	0.55	1	0.9	0.56	0.38
Test Accuracy	0.45	0.55	0.45	0.45	**0.55**	0.3
Test F1-Macro	0.16	0.18	0.16	0.16	**0.4**	0.16
Test F1-Weighted	0.35	0.40	0.36	0.36	**0.51**	0.27

Πίνακας 4.13: Στον παραπάνω πίνακα βλέπουμε τις μετρικές για τους διάφορους meta-learners χρησιμοποιώντας Distance-based μετα-χαρακτηριστικά.

Ferrari-Attributes	RFC	ABC	GBC	HGBC	KnnC	MLP
Train Accuracy	1	0.56	1	0.93	0.63	1
Train F1-Macro	1	0.4	1	0.93	0.46	1
Train F1-Weighted	1	0.51	1	0.93	0.59	1
Test Accuracy	0.4	0.5	0.35	0.35	0.4	0.3
Test F1-Macro	0.23	0.25	0.2	0.14	0.24	0.18
Test F1-Weighted	0.38	0.45	0.33	0.3	0.41	0.32

Πίνακας 4.14: Στον παραπάνω πίνακα βλέπουμε τις μετρικές για τους διάφορους meta-learners χρησιμοποιώντας Ferrari-Attributes μετα-χαρακτηριστικά.

AutoClust	RFC	ABC	GBC	HGBC	KnnC	MLP
Train Accuracy	1	0.7	1	0.87	0.7	0.66
Train F1-Macro	1	0.3	1	0.86	0.44	0.2
Train F1-Weighted	1	0.61	1	0.87	0.68	0.53
Test Accuracy	0.27	**0.55**	0.27	0.44	0.38	0.5
Test F1-Macro	0.11	**0.3**	0.18	0.26	0.15	0.16
Test F1-Weighted	0.23	**0.41**	0.24	0.42	0.3	0.33

Πίνακας 4.15: Στον παραπάνω πίνακα βλέπουμε τις μετρικές για τους διάφορους meta-learners χρησιμοποιώντας AutoClust μετα-χαρακτηριστικά.

Παρατηρούμε ότι UD2V-CE παράγει τα καλύτερα αποτελέσματα για 4 στους 6 meta-learners, ενώ την καλύτερη ακρίβεια στα testing datasets την παίρνουμε για τον HGBC meta-learner για όλα τα UD2V μετα-χαρακτηριστικά. Για τον ABC meta-learner τα UD2V χαρακτηριστικά φαίνεται να υστερούν πολύ σε σχέση με τους ανταγωνιστές τους, για τον KnnC υστερούν ελάχιστα μόνο ως προς τα F_1 scores, ενώ για τον MLP υπερτερούν μόνο λίγο σε σχέση με τα χαρακτηριστικά του AutoClust.
Βέλτιστο F_1-Macro εντοπίζεται για τα Distance-based με τον KnnC meta-learner και F_1-Weighted για τα Distance-based με τον KnnC με ελάχιστη διαφορά από τα UD2V-M=0.1με τον MLP. Εν γένει όμως, δεν μπορούμε να πούμε ότι τα αποτελέσματα όσον αφορά το F_1-Macro θεωρούνται καλά προκειμένου να βγάλουμε ασφαλή συμπεράσματα ως προς τα ποια μετα-χαρακτηριστικά και με ποιον meta-learner δίνουν καλύτερα αποτελέσματα. Το πρόβλημα αυτό θα μπορούσαμε να το αποδώσουμε στην ανομοιογένια των κλάσεων των αλγορίθμων τόσο στο σύνολο εκπαίδευσης, όσο και στο σύνολο αξιολόγησης και θα μπορούσαμε να το αντιμετωπίσουμε μελλοντικά με επιλογή συνόλων δεδομένων εκπαίδευσης με πιο ομοιόμορφη κατανομή βέλτιστων αλγορίθμων.

Τέλος εφαρμόζουμε Leave One Out Cross Validation στα αντίστοιχα μοντέλα, χρησιμοποιώντας μόνο τα testing datasets

	RFC	ABC	GBC	HGBC	KnnC	MLP
DB-Accuracy	**0.7**	0.6	**0.65**	0.55	0.6	0.15
AutoClust-Accuracy	0.22	0.38	0.27	0.5	0.27	0.16
Ferrari Attributes-Accuracy	0.6	0.5	0.6	0.55	0.5	0.5
UD2V-CE/M=0.1/M=1-Accuracy	0.55	**0.65**	0.55	0.55	**0.65**	0.6
UD2V M=2-Accuracy	0.6	**0.65**	0.6	0.55	**0.65**	**0.65**

Πίνακας 4.16: Στον παραπάνω πίνακα βλέπουμε την ακρίβεια μέσω Leave One Out Cross Validation (LOOCV) για τα διάφορα μετα-χαρακτηριστικά που χρησιμοποιήθηκαν στον πειραματισμό, εφαρμόζοντάς το στα testing datasets. Οι στήλες αντιπροσωπεύουν τους διάφορους meta-learners που χρησιμοποιήθηκαν.

Βλέπουμε ότι τα UD2V μέτα-χαρακτηριστικά δίνουν καλύτερη ακρίβεια για 3 στα 6 μοντέλα, για 1 δίνουν τα ίδια με σχεδόν όλους τους ανταγωνιστές, ενώ για τους meta-learners RFC και GBC, τα Distance-based μέτα-χαρακτηριστικά φαίνεται να υπερτερούν όλων.

4.2.2 HPO

Για το κομμάτι του HPO όπως περιγράψαμε στο κεφάλαιο 3.4, χρησιμοποιούμε τους παρακάτω 9 internal CVIs:
1) Silhouette index [11]
2) Dunn index [11]
3) C-Index [11]
4) Calinski Harabazs [11]
5) Davies Bouldin [11]
6) CDBW [12]
7) Tau index [13]
8) Ratkowky Lance [13]
9) McClain-Rao index [13]

Παίρνοντας μετρήσεις για τους 4 αλγορίθμους που εξετάζουμε με βάση την εξαντλητική αναζήτηση που κάναμε, εκπαιδεύουμε 4 νευρωνικά δίκτυα με αρχιτεκτονική 10×Dense(128) χρησιμοποιώντας ReLU ως activation function και παίρνουμε τις παρακάτω μετρήσεις εκπαίδευσης:

	Train Samples	Test Samples	Train MAE	Test MAE	Train MSE	Test MSE
K-Means	760	304	0.0861	0.0606	0.0134	0.0053
Spectral Clustering	81,658	22,543	0.0894	0.0881	0.0188	0.0141
Agglomerative Clustering	6,156	2,736	0.0689	0.1334	0.0077	0.0428
DBSCAN	1,435	543	0.0806	0.0437	0.0247	0.0036

Πίνακας 4.17: Στον παραπάνω πίνακα βλέπουμε τις κατανομές των συνόλων δεδομένων εκπαίδευσης και πειραματισμού των regressors που χρησιμοποιούνται για το HPO (πλειάδες από τα ανωτέρω CVIs). Παράλληλα δίνεται και τα Mean Absolute Error (MAE) - Mean Squared Error (MSE) αυτών, αναφορικά με το target value που είναι το ARI.

Ως κριτήρια σύγκρισης των optimizers θα θεωρήσουμε για το Random Search και τους Bayesian optimizers την ελάχιστη εκτέλεση στην επανάληψη της εμφάνισης της βέλτιστης τιμής καθώς επίσης και το Μέσο Απόλυτο Σφάλμα (Mean Absolute Error, MAE). Ως μέσο απόλυτο σφάλμα ορίζουμε την τιμή

$$MAE = \frac{1}{\#steps} \sum_{t=1}^{\#steps} |y_t - \tilde{y}|$$

όπου y_t είναι η πραγματική τιμή του ARI για την παραμετροποίηση που προκύπτει στο t βήμα και $\tilde{y}$ είναι η βέλτιστη τιμή του ARI που βρίσκουμε μέσα απ΄ τη διαδικασία της εξαντλητικής αναζήτησης.

Ως ελάχιστη εκτέλεση στην επανάληψη εμφάνισης της βέλτιστης τιμής ($\min_{step}$), ορίζουμε την ελάχιστη τιμή της επανάληψης που απαιτείται για να πετύχουμε βέλτιστη τιμή της μετρικής που βελτιστοποιούμε. Για παράδειγμα αν σε 25 επαναλήψεις του αλγορίθμου βελτιστοποίησης, η βέλτιστη τιμή παρατηρείται κατά την 13 επανάληψη του αλγορίθμου βελτιστοποίησης, τότε θα έχουμε ότι $\min_{step} = 13$. Ιδιαίτερα, έχουμε ότι για n το πλήθος επαναλήψεις, το $\min_{step}$ δίνεται ως

$$\min_{step} = arg \max_{i=1,...,n} \{\text{compute ARI at step i}\}$$

Για 50 επαναλήψεις και χρησιμοποιώντας Bayesian Optimization με TPE (TPE) και Random Search (RS), παίρνουμε τα παρακάτω αποτελέσματα για τα σύνολα δεδομένων που χρησιμοποιούμε για testing

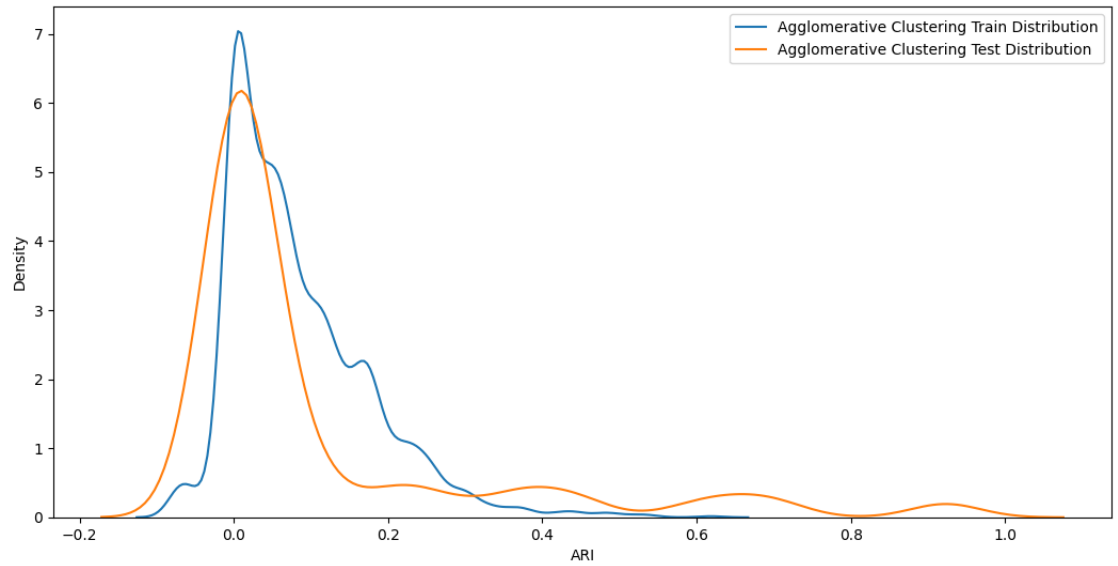
Testing Dataset	$\min_{steps}$-RS	$\min_{steps}$-TPE	MAE-RS	MAE-TPE	Algorithm
mobile_price_classification	3	33	0.24193	0.19117	Agglomerative
pc1_req	1	17	0.04794	0.03120	Agglomerative
pc3	4	4	0.10090	0.11970	Agglomerative
pc4	9	9	0.09391	0.12061	Agglomerative
prnn_viruses	29	20	0.50488	0.38408	Agglomerative
pima	5	5	0.14118	0.12950	Spectral
MindCave2	19	31	0.15902	0.13364	Spectral
monks-problems-2	11	28	0.05009	0.05210	Spectral
mu284	21	7	0.10282	0.09505	Spectral
mux6	4	23	0.12990	0.12717	Spectral
mw1	32	36	0.31585	0.31136	Spectral
newton_hema	47	19	0.09347	0.07882	Spectral
PopularKids	13	12	0.04019	0.03502	Spectral
prnn_fglass	3	11	0.18898	0.20681	Spectral
parkinsons	24	14	0.33555	0.26533	Spectral
post-operative	19	47	0.114891	0.11733	Spectral
pc1	17	20	0.19234	0.126783	DBSCAN
primary-tumor	47	47	0.15764	0.13466	DBSCAN
forest-type-mapping	3	33	0.24193	0.19117	KMeans
heart-statlog	6	6	0.26849	0.26700	KMeans

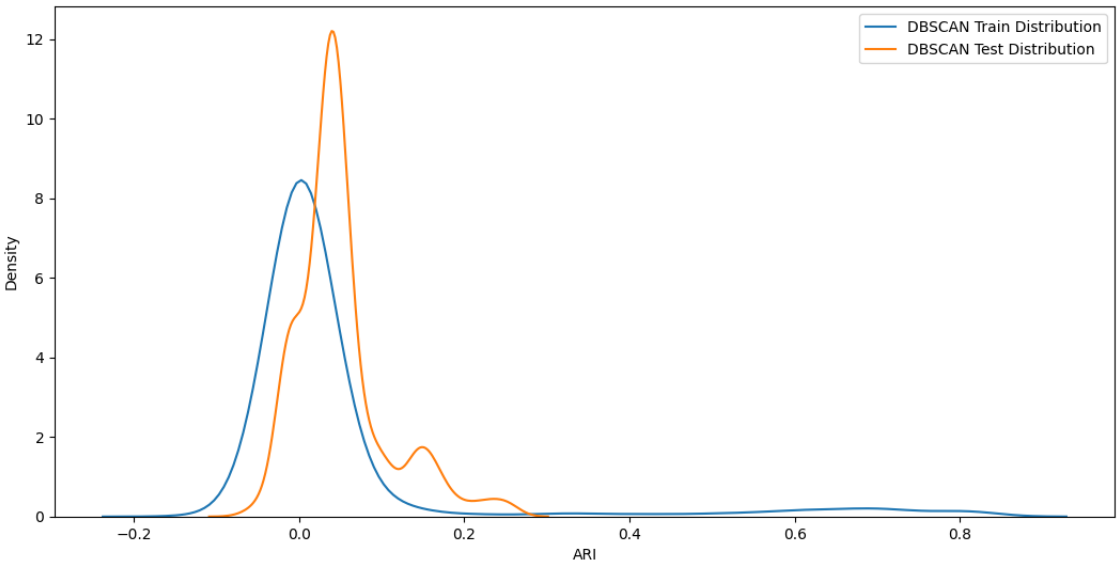
Πίνακας 4.18: Στον παραπάνω πίνακα βλέπουμε τις μετρήσεις που πάρθηκαν για το HPO.Οι γραμμές προσδιορίζουν τα διάφορα σύνολα δεδομένων, ενώ οι στήλες τις αντίστοιχες μετρικές που παρατηρήθηκαν. Η στήλη Algorithm προσδιορίζει τον εκ των προτέρων προσδιορισμένο ως βέλτιστο αλγόριθμο.

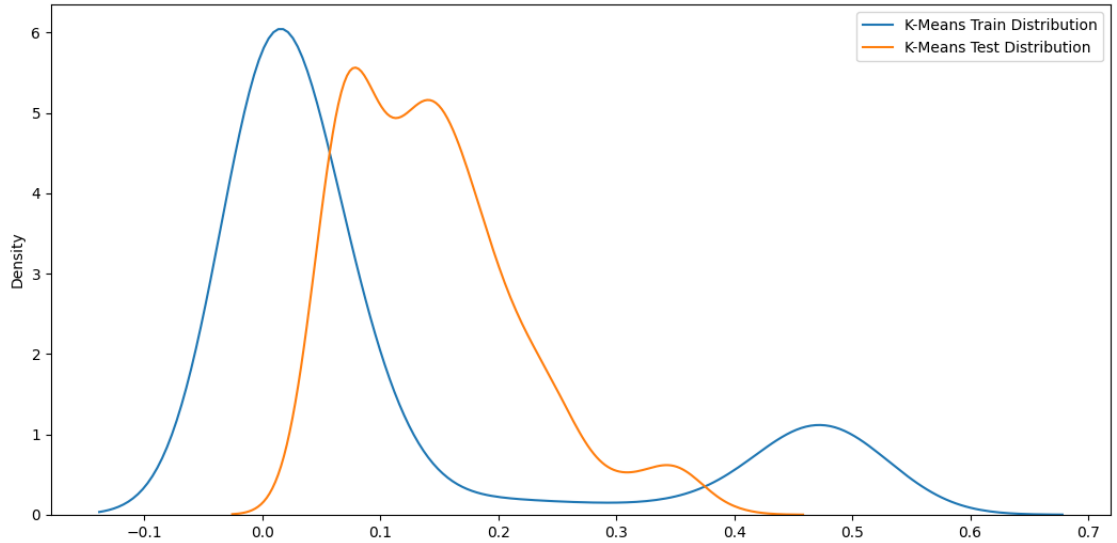
Παρατηρούμε ότι το με RS φαίνεται να έχουμε καλύτερα αποτελέσματα αναφορικά με το $\min_{step}$ (9 στις 20 φορές έχουμε καλύτερο $\min_{steps}$ για RS και μόλις 5 στις 20 για TPE), ενώ για το MAE 15/20 φορές χρησιμοποιώντας TPE παίρνουμε μικρότερο MAE.

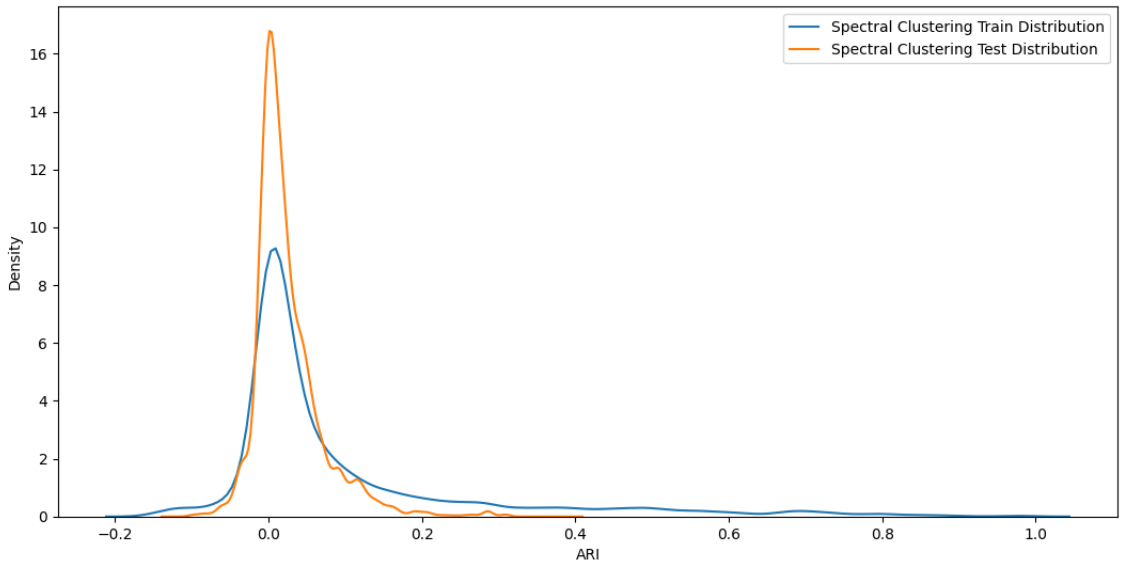
Το μεγάλο MAE σε ορισμένα σύνολα δεδομένων, οφείλεται στο γεγονός ότι η κατανομή του ARI διαφέρει στα σύνολα δεομένων με τα οποία εκπαιδεύουμε τα νευρωνικά δίκτυα με τα οποία εκτιμούμε το ARI σε σχέση με τα

σύνολα των οποίων το ARI θέλουμε να εκτιμήσουμε. Το γεγονός αυτό φαίνεται στα παρακάτω γραφήματα των πυκνοτήτων του ARI:









Σχήμα 4.6: Στους παραπάνω πίνακες βλέπουμε την κατανομή του πραγματικού ARI για τις διάφορες παραμετροποιήσεις των 4 προς εξέταση αλγορίθμων. Με μπλε χρώμα παρουσιάζεται η κατανομή του ARI των συνόλων δεδομένων που χρησιμοποιήθηκαν για training ενώ με πορτοκαλί χρώμα των συνόλων δεδομένων που χρησιμοποιήθηκαν για testing

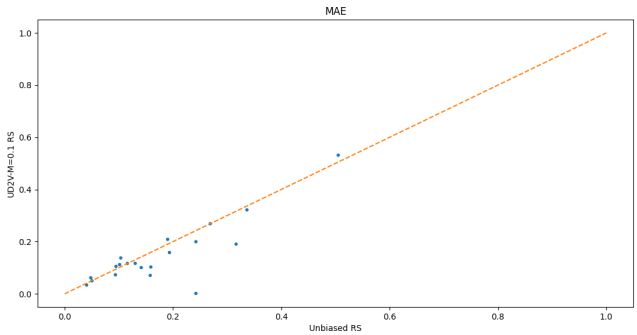
Η εκπαίδευση των νευρωνικών δικτύων με δεδομένα πιο ομοιόμορφης κατανομής ή αξιολόγηση συνόλων δεδομένων με κατανομή πραγματικού ARI όμοια της κατανομής εκπαίδευσης του αντίστοιχου νευρωνικού δικτύου, θα απέδιδε καλύτερα ως προς την αξιολόγηση.

Στη συνέχεια έχοντας ως budget_threshold τις 15 επαναλήψεις βρίσκουμε τα αντίστοιχα αποτελέσματα, χρησιμοποιώντας ως αλγόριθμο εκείνον ο οποίος προκύπτει απ΄ την knn σύσταση, χρησιμοποιώντας τα μέτα-χαρακτηριστικά του UD2V-M=0.1 για k=3.

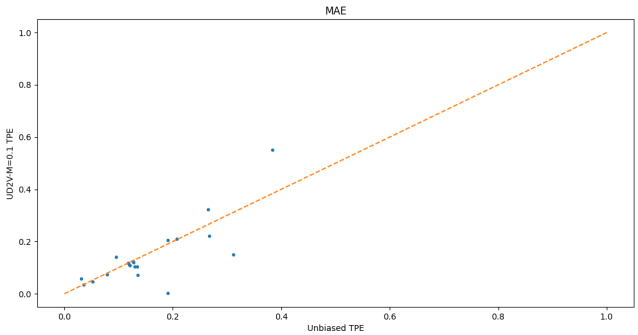
Testing Dataset	$\min_{steps}$-RS	$\min_{steps}$-TPE	MAE-RS	MAE-TPE
mobile_price_classification	14	7	0.00352	0.00358
pc1_req	6	6	0.06216	0.05743
pc3	7	12	0.11401	0.11179
pc4	6	6	0.10650	0.10954
prnn_viruses	14	7	0.53279	0.55110
pima	6	6	0.10243	0.10323
MindCave2	6	6	0.10486	0.10460
monks-problems-2	3	15	0.05024	0.04745
mu284	4	4	0.13954	0.14039
mux6	14	13	0.11848	0.11900
mw1	6	6	0.19101	0.14955
newton_hema	7	7	0.07488	0.07455
PopularKids	13	6	0.03472	0.03583
prnn_fglass	14	15	0.20897	0.21000
parkinsons	14	14	0.32241	0.32190
post-operative	13	4	0.11767	0.11833
pc1	12	4	0.15911	0.12442
primary-tumor	1	1	0.07113	0.07154
forest-type-mapping	3	3	0.20069	0.20519
heart-statlog	6	14	0.26875	0.22132

Πίνακας 4.19: Στον παραπάνω πίνακα βλέπουμε τις μετρήσεις που πάρθηκαν για το HPO έπειτα από τη σύσταση αλγορίθμου με 3-nn query για τα UD2V-M=0.1 μετα-χαρακτηριστικά. Οι γραμμές προσδιορίζουν τα διάφορα σύνολα δεδομένων, ενώ οι στήλες τις αντίστοιχες μετρικές που πάρθηκαν.

Παρατηρούμε ότι οι τιμές των MAE για τα Random Search και TPE χρησιμοποιώντας τη σύσταση του UD2V-M=0.1 και 15 επαναλήψεις, κυμαίνονται σχεδόν στα ίδια επίπεδα με αυτά των πραγματικών βέλτιστων αλγορίθμων για 50 επαναλήψεις. Πιο συγκεκριμένα έχουμε τους παρακάτω πίνακες



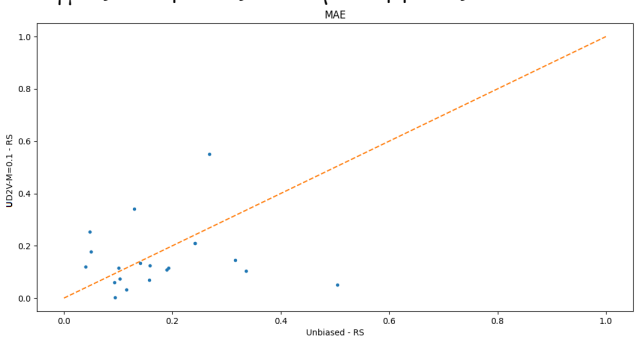
RS MAE Pearson's Correlation = 0.85519367



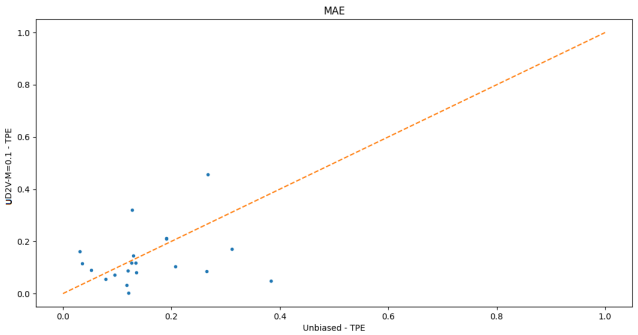
TPE MAE Pearson's Correlation = 0.79875063

Σχήμα 4.7: Στους παραπάνω πίνακες συγκρίνουμε το MAE για την RS και την TPE προσέγγιση αντίστοιχα, χρησιμοποιώντας το βέλτιστο αλγόριθμο όπως αυτός προκύπτει απ΄ το Exchaustive Search (οριζόντιος άξονας - Unbiased) και απ΄ τα UD2V-M=0.1 μετα-χαρακτηριστικά με 3-nn query. Για να συγκρίνουμε τη δυναμική σύστασης του UD2V-M=0.1 μοντέλου αναφορικά με το πλήθος επαναλήψεων, χρησιμοποιήθηκαν 50 επαναλήψεις (στο RS και το TPE) για τα testing σύνολα δεδομένων στην Unbiased αξιολόγηση, ενώ 15 επαναλήψεις για την UD2V-M=0.1. Παρατηρούμε ότι τόσο στο RS, όσο και στο TPE υπάρχει μεγάλη θετική γραμμική συσχέτιση για την Unbiased και την UD2V-M=0.1 εφαρμογή, αλλά όχι τέλεια (≈ 1). Με πορτοκαλί χρώμα προβάλλεται η ευθεία y=x ενώ τα σημεία αντιπροσωπεύουν (στις αντίστοιχες προβολές των αξόνων) τα MAE για τα testing σύνολα δεδομένων. Συμπερασματικά μπορούμε να πούμε ότι παρότι χρησιμοποιούμε περισσότερες επαναλήψεις για την Unbiased αξιολόγηση των αλγορίθμων, με εξαίρεση κάποια σύνολα δεδομένων στα οποία συστήνεται αλγόριθμος διαφορετικός του βέλτιστου, η σύσταση βάσει του UD2V-M=0.1 δίνει MAE πολύ κοντά στο πραγματικό βέλτιστο.

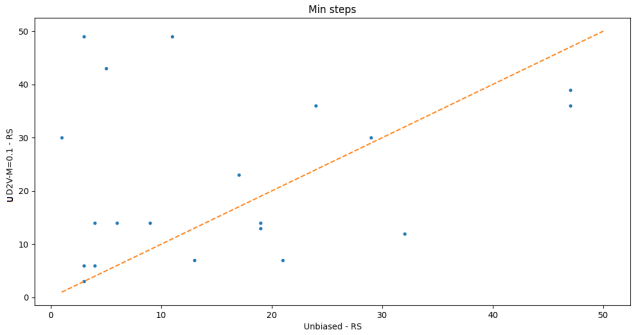
Στη συνέχεια για να μπορέσουμε να συγκρίνουμε και το min_steps παραθέτουμε την αντίστοιχη σύγκριση για 50 επαναλήψεις και για τις δύο προσεγγίσεις



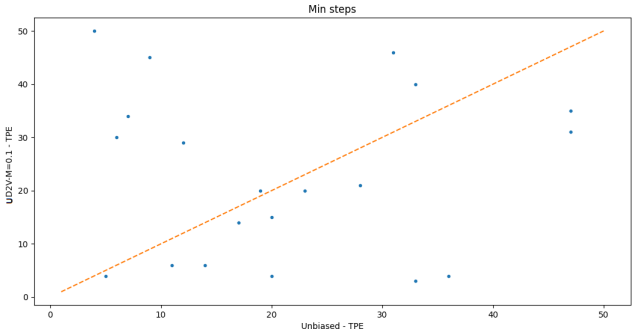
RS MAE Pearson's Correlation = 0.07005391



TPE MAE Pearson's Correlation = 0.2176178



RS min_steps Pearson's Correlation = 0.238456



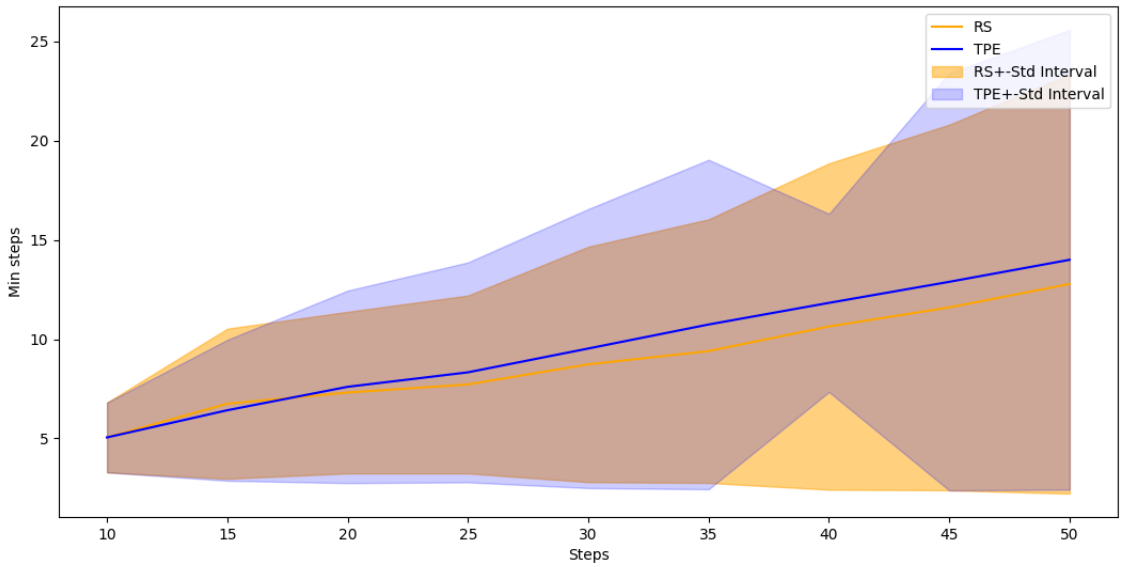
TPE min_steps Pearson's Correlation = 0.00888285

Σχήμα 4.8: Στους παραπάνω πίνακες συγκρίνουμε το MAE και το min_steps για την RS και την TPE προσέγγιση αντίστοιχα, χρησιμοποιώντας το βέλτιστο αλγόριθμο όπως αυτός προκύπτει απ΄ το Exchaustive Search (οριζόντιος άξονας - Unbiased) και απ΄ τα UD2V-M=0.1 μετα-χαρακτηριστικά με 3-nn query. Όλες οι μετρικές πάρθηκαν χρησιμοποιώντας 50 επαναλήψεις. Παρατηρούμε ότι χρησιμοποιώντας 50 επαναλήψεις και για τον UD2V-M=0.1 παίρνουμε μικρότερη (σχεδόν μηδενική) γραμμική συσχέτιση για το MAE των συνόλων δεδομένων για RS ενώ ελάχιστα καλύτερη είναι η συσχέτιση για TPE. Παρόμοια είναι η συσχέτιση (εξαιρετικά χαμηλή) και για τη μετρική min_steps. Από τη σύγκριση της συσχέτισης του MAE στα σχήματα 4.7 και 4.8, μπορούμε να πούμε ότι για τα διάφορα σύνολα δεδομένων έχουμε καλύτερη συσχέτιση με τις τιμές που δίνουν οι παραμετροποιήσεις των regressors χρησιμοποιώντας τους κανονικούς βέλτιστους αλγόριθμους, χρησιμοποιώντας λιγότερες το πλήθος επαναλήψεις.

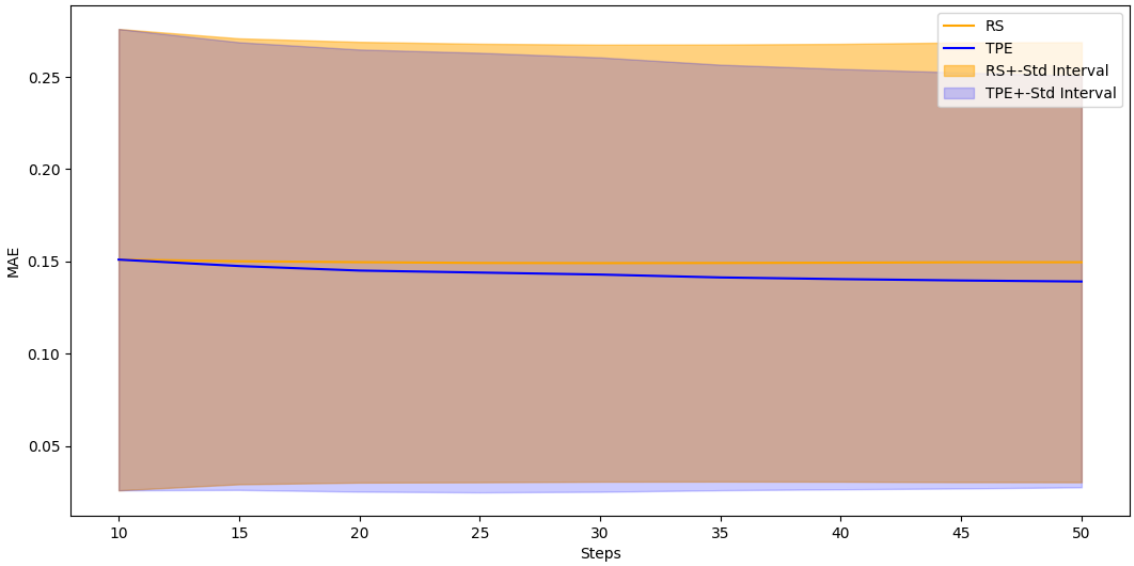
Παρακάτω για διάφορες το πλήθος επαναλήψεις, βρίσκουμε τη μέση τιμή και την τυπική απόκλιση των μέτρων $\min_{step}$ και MAE για το Random Search και το Bayesian optimization με TPE επί του συνόλου των testing συνόλων δεδομένων, χρησιμοποιώντας πάλι 3-nn query με βάση τα UD2V-M=0.1 μετα-χαρακτηριστικά

Steps	RS - Mean $\min_{steps}$±StD	RS - Mean MAE±StD	TPE - Mean $\min_{steps}$±StD	TPE - Mean MAE±StD
10	5.05±1.76	0.15099±0.1250	5.05±1.76	0.15099±0.1250
15	6.75±3.78	0.15009±0.1208	6.42±3.55	0.14751±0.1213
20	7.31±4.07	0.14960±0.1194	7.6±4.85	0.14506±0.1198
25	7.72±4.48	0.14918±0.1188	8.33±5.54	0.14397±0.1191
30	8.73±5.93	0.14910±0.1184	9.53±7.03	0.14289±0.1177
35	9.4±6.64	0.14917±0.1184	10.74±8.3	0.14133±0.1153
40	10.64±8.22	0.14931±0.1186	11.83±4.49	0.14043±0.1139
45	11.60±9.21	0.14956±0.1191	12.89±10.51	0.13969±0.1127
50	12.78±10.56	0.14959±0.1192	14±11.58	0.13912±0.1115

Πίνακας 4.20: Στον παραπάνω πίνακα βλάπουμε το μέσο όρο±StD για το σύνολο των αλγορίθμων πειραματισμού χρησιμοποιώντας την RS και την TPE προσέγγιση, για τις κλιμακωτά διάφορες το πλήθος επαναλήψεις που έγιναν.



Σχήμα 4.9: Στο παραπάνω γράφημα βλέπουμε το μέσο min_steps ± τη τυπική απόκλισή του για διάφορα το πλήθος επιλεχθέντα επαναληπτικά βήματα. Στον οριζόντιο άξονα προβάλλεται το πλήθος των βημάτων, ενώ στον κατακόρυφο το μέσο min_steps. Με μπλε χρώμα δίνονται οι μετρήσεις για χρήση της Random Search προσέγγισης, ενώ με πορτοκαλί για χρήση Bayesian optimization με TPE. Φαίνεται ότι το min_steps αυξάνει γραμμικά και για τις 2 προσεγγίσεις, με πολύ μικρή διαφορά στις τυπικές αποκλίσεις.



Σχήμα 4.10: Στο παραπάνω γράφημα βλέπουμε το μέσο MAE ± τη τυπική απόκλισή του για διάφορα το πλήθος επιλεχθέντα επαναληπτικά βήματα. Στον οριζόντιο άξονα προβάλλεται το πλήθος των βημάτων, ενώ στον κατακόρυφο το μέσο MAE. Με μπλε χρώμα δίνονται οι μετρήσεις για χρήση της Random Search προσέγγισης, ενώ με πορτοκαλί για χρήση Bayesian optimization με TPE.Φαίνεται ότι με χρήση TPE το μέσο MAE φθίνει (μαζί με την αντίστοιχη τυπική απόκλιση) καθώς μεγαλώνουμε το πλήθος των επαναλήψεων, ενώ με χρήση RS το μέσο MAE διατηρρείται σταθερό (μαζί με την αντίστοιχη τυπική απόκλιση).

Παρατηρούμε ότι αφήνοντας το budget_threshold να αυξάνεται απαιτούνται περισσότερες επαναλήψεις για τον TPE σε σχέση με τον RS, αλλά ταυτόχρονα το μέσο MAE του TPE φαίνεται να φθίνει γρηγορότερα σε σχέση με αυτό του RS.

Κεφάλαιο 5

Συμπεράσματα

Στην παρούσα διπλωματική εργασία ασχοληθήκαμε με το πρόβλημα της αυτοματοποιημένης μηχανικής μάθησης για αλγορίθμους μη-επιβλεπτόμενης μηχανικής μάθησης. Σχεδιάσαμε ένα σύστημα το οποίο προσπαθεί να λύσει το πρόβλημα CASH, δηλαδή αυτό της επιλογής βέλτιστου αλγορίθμου και της εύρεσης βέλτιστων υπερπαραμέτρων για αυτόν. Για το λόγο αυτό φτιάξαμε έναν meta-feature extractor ο οποίος αποτελεί μια unsupervised προσέγγιση του meta-feature extractor Dataset2Vec [5] για ομοιότητα συνόλων δεδομένων με χρήση κριτηρίου βάσει εκτέλεσης βέλτιστου αλγορίθμου, με στόχο να αντιμετωπίσουμε το πρόβλημα της σύστασης βέλτιστου αλγορίθμου.

Χρησιμοποιώντας μια συγκερκιμένη αρχιτεκτονική των νευρωνικών δικτύων τα οποία απαρτίζουν τον meta-feature extractor μας κι εκπαιδεύοντάς τον για 2 διαφορετικές συναρτήσεις κόστους, είδαμε ότι βάσει των μετρικών που χρησιμοποιούμε για σύγκριση, μπορούμε να πάρουμε ισοδύναμα ή και καλύτερα αποτελέσματα σε σχέση με ανταγωνιστικές τεχνικές παραγωγής μέτα-χαρακτηριστικών.

Αντίστοιχα για το πρόβλημα της εύρεσης βέλτιστου συνδυασμού υπερπαραμέτρων για τους αλγορίθμους στο σύστημά μας, χρησιμοποιήσαμε την προσέγγιση του AutoClust [1], δημιουργώντας νευρωνικά δίκτυα για την εκτίμηση του ARI βάσει κάποιων internal CVIs. Πιο συγκεκριμένα μέσω της παραπάνω προσέγγισης συγκρίναμε έναν Bayesian Optimizer με την Random Search προσέγγιση, βλέποντας ότι με κριτήριο διασποράς πρόβλεψης το Μέσο Απόλυτο Σφάλμα, η χρήση τεχνικών Bayesian optimization είναι καλύτερη.

Δεδομένου ότι με χρήση μιας συγκεκριμένης αρχιτεκτονικής για τον meta-feature extractor παίρνουμε συγκεκριμένα αποτελέσματα, μπορούμε σε μελλοντική έρευνα, να εφαρμόσουμε μέτα-βελτιστοποίηση σε επίπεδο αρχιτεκτονικών των δικτύων που απαρτίζουν τον meta-feature extractor, προκειμένου να βελτιστοποιήσουμε τον meta-feature extractor ως προς μια συγκεκριμένη μετρική, μέσω Bayesian Optimization.

Σε επίπεδο μελλοντικής έρευνας μπορούν επίσης να μελετηθούν κι άλλες unsupervised προσεγγίσεις πέραν της Column-wise Pooling δομής που χρησιμοποιήσαμε, οι οποίες θα μπορούν πέραν της συσχέτισης μεταξύ των instances να βρουν τις συσχετίσεις και μεταξύ των features. Τέτοιες προσεγγίσεις μπορούν εκτός του προβλήματος της σύστασης βέλτιστου αλγορίθμου, που αποτέλεσε και τον πυρήνα της έρευνας μας σε αυτή την εργασία, να χρησιμοποιηθούν και σε μια πληθώρα άλλων εφαρμογών της ροής εργασιών μηχανικής μάθησης, όπως είναι η επιλογή χαρακτηριστικών (feature engineering) κα, προκειμένου να εισαχθούν σε ένα πλαίσιο αυτοματισμού σε ένα AutoML σύστημα. Επιπροσθέτως, θα ήταν χρήσιμη η επέκταση του meta-feature extractor προκειμένου να μην παράγει μετα-χαρακτηριστικά μόνο για tabular σύνολα δεδομένων, αλλά να μπορεί να επεκταθεί και να κατανοήσει συσχετίσεις κι άλλων συνόλων δεδομένων όπως είναι τα σύνολα δεδομένων εικόνων, χρονοσειρών κα.

Τέλος, εξαιτίας της χρήσης νευρωνικών δικτύων στον meta-feature extractor μας, θεωρούμε ότι σημαντικό ρόλο παίζει κι η ερμηνευσιμότητα των αποτελεσμάτων (explainability). Η ερμηνευσιμότητα στον τομέα της βαθιάς μηχανικής μάθησης (Explainable AI, XAI) είναι ένα ερευνητικό πεδίο το οποίο συγκεντρώνει ολοκένα και περισσότερο ενδιαφέρον, καθότι επικεντρώνεται στο ερώτημα "γιατί το σύστημά μας παράγει το αποτέλεσμα που

μας δίνει" και σκοπό έχει να προσεγγίσει τη βαθιά μάθηση πέραν του πρίσματος μιας black box εφαρογής. Για το λόγο αυτό, κρίνεται σκόπιμη η μελλοντική προσπάθεια ερμηνείας της συσχέτισης των instances που βρίσκει ο meta-feature extractor μας.

Κεφάλαιο 6

Βιβλιογραφία

1. Y. Poulakis, C. Doulkeridis and D. Kyriazis, "AutoClust: A Framework for Automated Clustering Based on Cluster Validity Indices," 2020 IEEE International Conference on Data Mining (ICDM), 2020, pp. 1220-1225, doi: 10.1109/ICDM50108.2020.00153.

2. D. G. Ferrari and L. N. de Castro, "Clustering algorithm selection by meta-learning systems: A new distance-based problem characterization and ranking combination methods", Inf. Sci., vol. 301, pp. 181-194, 2015.

3. Dennis Tschechlov, Manuel Fritz, Holger Schwarz, AutoML4Clust: Efficient AutoML for Clustering Analyses, EDBT, Pages 343-348, OpenProceedings.org, 2021

4. Noy Cohen-Shapira, Lior Rokach, Automatic selection of clustering algorithms using supervised graph embedding, Information Sciences, Volume 577, 2021, Pages 824-851, ISSN 0020-0255

5. Jomaa, H.S., Schmidt-Thieme, L. and Grabocka, J. Dataset2Vec: learning dataset meta-features. Data Min Knowl Disc 35, 964–985 (2021). https://doi.org/10.1007/s10618-021-00737-9

6. J. Vanschoren, "Meta-Learning: A survey," 2018

7. R. Hadsell, S. Chopra and Y. LeCun, "Dimensionality Reduction by Learning an Invariant Mapping," 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), 2006, pp. 1735-1742, doi: 10.1109/CVPR.2006.100.

8. Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. 2019. Automated Machine Learning: Methods, Systems, Challenges (1st. ed.). Springer Publishing Company, Incorporated.

9. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. Journal of Machine Learning Research 13, 281–305 (2012)

10. James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for hyper-parameter optimization. In Proceedings of the 24th International Conference on Neural Information Processing

Systems (NIPS'11). Curran Associates Inc., Red Hook, NY, USA, 2546–2554.

11. Olatz Arbelaitz, Ibai Gurrutxaga, Javier Muguerza, Jesús M. Pérez, Iñigo Perona, An extensive comparative study of cluster validity indices, Pattern Recognition, Volume 46, Issue 1, 2013, Pages 243-256, ISSN 0031-3203, https://doi.org/10.1016/j.patcog.2012.07.021.

12. M. V. Maria Halkidi, A density-based Cluster Validity Approach using Multi-representatives, 2008.

13. M. Charrad, N. Ghazzali, V. Boiteau, and A. Niknafs. An examination of indices for determining the number of clusters : NbClust Package. 09 2013.

14. Ian J. Goodfellow and Yoshua Bengio and Aaron Courville, Deep Learning, MIT Press, 2016

15. B. Shahriari, K. Swersky, Z. Wang, R. P. Adams and N. de Freitas, "Taking the Human Out of the Loop: A Review of Bayesian Optimization," in Proceedings of the IEEE, vol. 104, no. 1, pp. 148-175, Jan. 2016, doi: 10.1109/JPROC.2015.2494218.