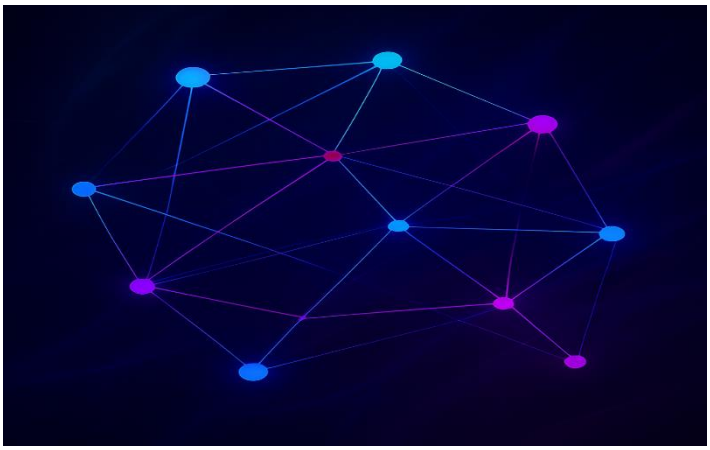
ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Multi-Label Embeddings and Explorations of Label Relations: A Literature Review and Comparative Study»



Του φοιτητή
Λοκοβίτη Αθανασίου
Αρ. Μητρώου: 093527

Επιβλέπων
Ουγιάρογλου Στέφανος
Βαθμίδα Επίκουρος καθηγητής

Ημερομηνία 16/6/2025

Τίτλος Π.Ε. Multi-Label Embeddings and Explorations of Label Relations: A Literature Review and Comparative Study

Κωδικός Π.Ε. 24268

Ονοματεπώνυμο φοιτητή/τών Λοκοβίτης Αθανάσιος

Ονοματεπώνυμο εισηγητή Ουγιάρογλου Στέφανος

Ημερομηνία ανάληψης Π.Ε. 22-10-2024

Ημερομηνία περάτωσης Π.Ε. 16-06-2025

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Πρόλογος

Η επιλογή του θέματος της παρούσας διπλωματικής εργασίας έγινε με σκοπό την εμβάθυνση σε έναν σύγχρονο και πολυσύνθετο τομέα της μηχανικής μάθησης, την κατηγοριοποίηση πολλαπλών ετικετών. Η ανάγκη για αξιόπιστες και αποδοτικές μεθόδους ταξινόμησης σε προβλήματα όπου κάθε δείγμα μπορεί να αντιστοιχεί σε περισσότερες από μία ετικέτες είναι ολοένα και πιο έντονη σε πολλούς επιστημονικούς και εφαρμοσμένους τομείς, όπως η ιατρική διάγνωση, η ανάλυση κειμένου και η αναγνώριση εικόνας.

Κατά τη διάρκεια της εκπόνησης αυτής της εργασίας, αποκτήθηκε ουσιαστική γνώση τόσο στη θεωρία των αλγορίθμων πολλαπλών ετικετών όσο και στην πρακτική υλοποίησή τους με σύγχρονα εργαλεία προγραμματισμού, όπως η γλώσσα προγραμματισμού Python. Η μελέτη των διαφορετικών τεχνικών μετασχηματισμού προβλήματος και η σύγκριση των επιδόσεων τους συνέβαλαν στην κατανόηση των πλεονεκτημάτων και περιορισμών των μεθόδων αυτών.

Τέλος, η εμπειρία αυτή ενίσχυσε το ενδιαφέρον για περαιτέρω έρευνα και ανάπτυξη στον τομέα της μηχανικής μάθησης, προσφέροντας παράλληλα σημαντική πρακτική κατάρτιση και επιστημονική εξειδίκευση.

Περίληψη

Η παρούσα πτυχιακή εργασία πραγματεύεται την κατηγοριοποίηση δεδομένων στο πλαίσιο προβλημάτων πολλαπλών ετικετών (multi-label classification), τα οποία εμφανίζονται σε πλήθος εφαρμογών της μηχανικής μάθησης, όπως η ανάλυση κειμένου, η μουσική κατηγοριοποίηση και η διάγνωση ασθενειών. Σε αντίθεση με την παραδοσιακή ταξινόμηση μονής ετικέτας, όπου κάθε δείγμα ανήκει αποκλειστικά σε μία κατηγορία, η ταξινόμηση πολλαπλών ετικετών επιτρέπει την ταυτόχρονη απόδοση πολλαπλών κατηγοριών σε κάθε παράδειγμα, γεγονός που καθιστά το πρόβλημα υπολογιστικά και εννοιολογικά πιο σύνθετο.

Στο πρώτο μέρος της εργασίας παρουσιάζονται οι βασικές έννοιες και τεχνικές της ταξινόμησης πολλαπλών ετικετών, με έμφαση στη χρήση εξειδικευμένων αλγορίθμων όπως οι BRkNN, MLkNN, MLARAM και MLTSVM. Εξετάζεται επίσης η εφαρμογή του Random Forest τόσο σε απλές όσο και σε παραλλαγμένες μορφές για την επίλυση τέτοιου τύπου προβλημάτων σε συνδυασμό με μεθόδους μετασχηματισμού προβλήματος όπως οι Binary Relevance, Classifier Chains και Label Powerset.

Ακολούθως, αναλύεται η διαδικασία υλοποίησης των παραπάνω αλγορίθμων σε περιβάλλον Python, με αξιοποίηση βιβλιοθηκών όπως η scikit-multilearn, και περιγράφονται οι παρεμβάσεις που απαιτήθηκαν για την προσαρμογή τους (π.χ. αριθμητικές σταθεροποιήσεις στον MLTSVM). Γίνεται εκτενής αναφορά στη φόρτωση και επεξεργασία των δεδομένων από οκτώ σύνολα δεδομένων πολλαπλών ετικετών, καθώς και στην ανάπτυξη των απαραίτητων λειτουργικοτήτων για την αξιολόγηση των μοντέλων.

Στο τέταρτο κεφάλαιο, παρουσιάζεται η πειραματική μελέτη, όπου εξετάζεται η απόδοση των υλοποιημένων αλγορίθμων σε όρους ακρίβειας, πληρότητας και άλλων μετρικών αξιολόγησης. Πραγματοποιείται τόσο ανάλυση ανά σύνολο δεδομένων όσο και συγκριτική αξιολόγηση συνολικά, με ιδιαίτερη έμφαση στις διαφορετικές στρατηγικές χρήσης του Random Forest. Επιπλέον, περιγράφονται προκλήσεις στην εφαρμογή συγκεκριμένων μεθόδων, όπως η αριθμητική αστάθεια στον MLTSVM.

Τέλος, η εργασία ολοκληρώνεται με τα συμπεράσματα και προτάσεις για μελλοντική έρευνα, υποδεικνύοντας πιθανούς τρόπους βελτίωσης της ακρίβειας και της αποδοτικότητας των μοντέλων ταξινόμησης πολλαπλών ετικετών, καθώς και την ανάγκη για καλύτερη αξιοποίηση ενσωμάτωσης πληροφορίας μεταξύ των ετικετών.

«Multi-Label Embeddings and Explorations of Label Relations: A Literature Review and Comparative Study»

Lokovitis Athanasios

Abstract

This undergraduate thesis explores the classification of data within the framework of multi-label classification problems, which frequently arise in various machine learning applications such as text analysis, music annotation, and medical diagnosis. In contrast to traditional single-label classification—where each instance is assigned to only one category—multi-label classification allows multiple labels to be simultaneously assigned to each instance, thereby increasing both the computational and conceptual complexity of the task.

The first part of the thesis presents the fundamental concepts and techniques of multi-label classification, with a focus on specialized algorithms including BRkNN, MLkNN, MLARAM, and MLTSVM. The study also examines the use of Random Forest, both in its basic form and in conjunction with transformation methods such as Binary Relevance, Classifier Chains, and Label Powerset, for addressing multi-label problems.

The implementation process of these algorithms in the Python programming environment is subsequently analyzed, utilizing libraries such as scikit-multilearn. Special attention is given to necessary modifications and adaptations, such as numerical stabilizations in MLTSVM, to ensure correct functionality. The thesis provides a detailed account of the data loading and preprocessing procedures for eight benchmark multi-label datasets, alongside the development of essential functionalities for model evaluation.

The fourth chapter presents the experimental study, assessing the performance of the implemented algorithms in terms of accuracy, recall, and other evaluation metrics. Both dataset-specific analysis and overall comparative evaluation are conducted, with a particular focus on the various strategies for employing Random Forest. Challenges encountered during implementation—such as numerical instability in the case of MLTSVM—are also discussed.

The thesis concludes with a summary of findings and suggestions for future research, pointing towards potential improvements in the accuracy and efficiency of multi-label classification models, and highlighting the importance of more effective utilization of label dependency and embedding techniques.

Περιεχόμενα

Κατάλογος Εικόνων

Κατάλογος Πινάκων

Συντομογραφίες

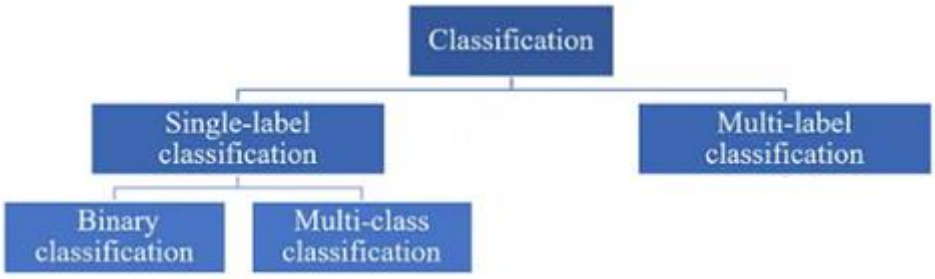
Δ.Ε.	Διπλωματική Εργασία
ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
Π.Ε.	Πτυχιακή Εργασία

Κεφάλαιο 1ο: Εισαγωγή

1.1 Κατηγοριοποίηση δεδομένων

Με τον όρο κατηγοριοποίηση δεδομένων (data classification) εννοούμε τη διαδικασία κατά την οποία ένα σύστημα μηχανικής μάθησης ή τεχνητής νοημοσύνης μαθαίνει να αναθέτει δείγματα/αντικείμενα σε μία ή περισσότερες προκαθορισμένες κλάσεις (κατηγορίες) με βάση τα χαρακτηριστικά που έχουν εξαχθεί από δεδομένα εισόδου. Πρόκειται για πρόβλημα εποπτευόμενης μάθησης (supervised learning), όπου το μοντέλο εκπαιδεύεται πάνω σε ένα σύνολο δεδομένων με γνωστές ετικέτες (labels) και στη συνέχεια χρησιμοποιείται για την πρόβλεψη των ετικετών σε νέα, άγνωστα δεδομένα. Αποτελεί μία από τις βασικότερες εργασίες στην εποπτευόμενη μηχανική μάθηση (supervised machine learning), με ευρύ φάσμα εφαρμογών όπως η ανίχνευση ανεπιθύμητης αλληλογραφίας, η ανάλυση συναισθήματος, η αναγνώριση εικόνας και η διάγνωση ασθενειών. Στην πιο γενική της μορφή, η κατηγοριοποίηση αποσκοπεί στην εκπαίδευση ενός αλγορίθμου ώστε να έχει τη δυνατότητα να προβλέπει την κατηγορία (ή τις κατηγορίες) ενός αντικειμένου βάσει ενός συνόλου χαρακτηριστικών

Όπως φαίνεται και στο σχήμα της Εικόνας 1 που ακολουθεί, η κατηγοριοποίηση δεδομένων κατατάσσεται σε δύο βασικές κατηγορίες, στην κατηγοριοποίηση με μοναδική ετικέτα (single-label classification) και στην κατηγοριοποίηση με πολλαπλές ετικέτες (multi-label classification), ανάλογα με τη φύση των ετικετών εξόδου και τη δομή του προβλήματος [1].



Εικόνα 1: Κατηγοριοποίηση Δεδομένων

Στην κατηγοριοποίηση μοναδικής ετικέτας, κάθε δείγμα δεδομένων ταξινομείται σε μία μόνο κατηγορία. Η κατηγορία αυτή διακρίνεται περαιτέρω σε δυαδική κατηγοριοποίηση (binary classification), όπου υπάρχουν δύο δυνατές κατηγορίες (0 ή 1), και σε κατηγοριοποίηση πολλαπλών κατηγοριών (multi-class classification), όπου τα δεδομένα μπορούν να ταξινομηθούν σε περισσότερες από δύο κατηγορίες [2]. Αντιθέτως, στην κατηγοριοποίηση πολλαπλών ετικετών, κάθε δείγμα μπορεί να συσχετιστεί ταυτόχρονα με περισσότερες από μία ετικέτες.

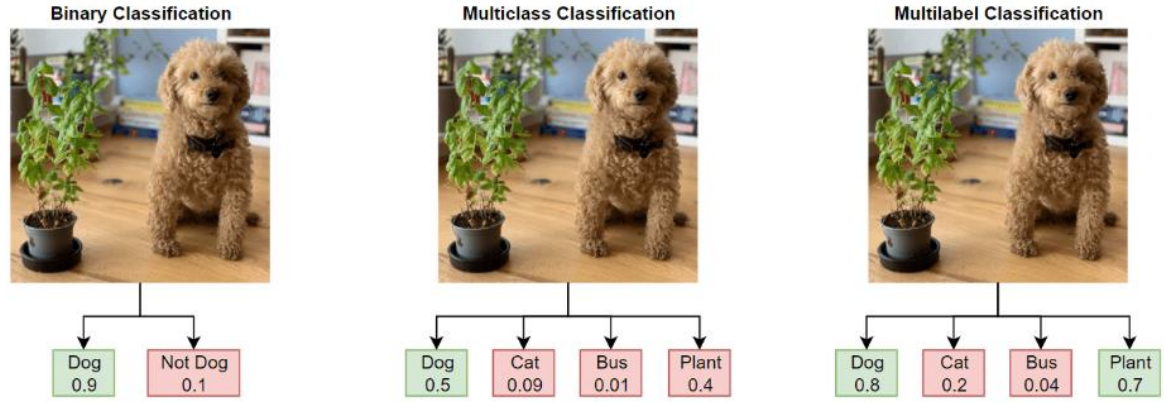
Πιο αναλυτικά:

- **Δυαδική κατηγοριοποίηση (binary classification):**
  Κάθε δείγμα αντιστοιχίζεται σε μία από δύο διακριτές κατηγορίες (0,1). Πρόκειται για την απλούστερη μορφή κατηγοριοποίησης, με εφαρμογές όπως η ανίχνευση ανεπιθύμητης αλληλογραφίας (spam vs. ham) ή η διάγνωση παθολογικών καταστάσεων (ασθενής vs. υγιής) [3].
- **Κατηγοριοποίηση πολλαπλών κλάσεων (multiclass classification):**

Το μοντέλο της δυαδικής κατηγοριοποίησης επεκτείνεται επιτρέποντας την ανάθεση κάθε παρατήρησης σε μία κατηγορία από ένα σύνολο περισσοτέρων των δύο. Ενδεικτικά παραδείγματα αποτελούν τα συστήματα αναγνώρισης χαρακτήρων (OCR – Optical Character Recognition) ή τα συστήματα αναγνώρισης αντικειμένων σε εικόνες [4]. Σε τέτοιες περιπτώσεις, εφαρμόζονται συχνά στρατηγικές όπως η One-vs-Rest (OvR) ή One-vs-One (OvO) για την προσαρμογή δυαδικών αλγορίθμων σε προβλήματα πολλαπλών κλάσεων.

- **Κατηγοριοποίηση πολλαπλών ετικετών (multi-label classification):**
  Το μοντέλο αυτό διαφοροποιείται ουσιαστικά, καθώς επιτρέπει την ταυτόχρονη ανάθεση ενός δείγματος σε περισσότερες από μία κατηγορίες. Απαντάται συχνά σε εφαρμογές όπως η θεματική ανάλυση κειμένων, η ανάκτηση περιεχομένου πολυμέσων και η βιοϊατρική πληροφορική [5]. Σε αντίθεση με τα προηγούμενα σχήματα, τα προβλήματα πολλαπλών ετικετών ενδέχεται να εμφανίζουν συσχετίσεις μεταξύ των ετικετών, γεγονός που καθιστά αναγκαία την ανάπτυξη εξειδικευμένων μεθόδων ικανών να εκμεταλλεύονται αυτές τις συσχετίσεις για τη βελτίωση της απόδοσης των μοντέλων [6].

Στην εικόνα που ακολουθεί (Εικόνα 2) απεικονίζονται τα αποτελέσματα απόδοσης κατηγορίας σε τρία διαφορετικά είδη κατηγοριοποίησης. Στις πρώτες δύο περιπτώσεις, binary και multiclass classification, έχουμε ως αποτέλεσμα την απόδοση μίας κατηγορίας στην δεδομένη εικόνα (Dog). Αντίθετα, στην περίπτωση του multilabel classification, αποδόθηκαν δύο κατηγορίες (Dog, Plant).



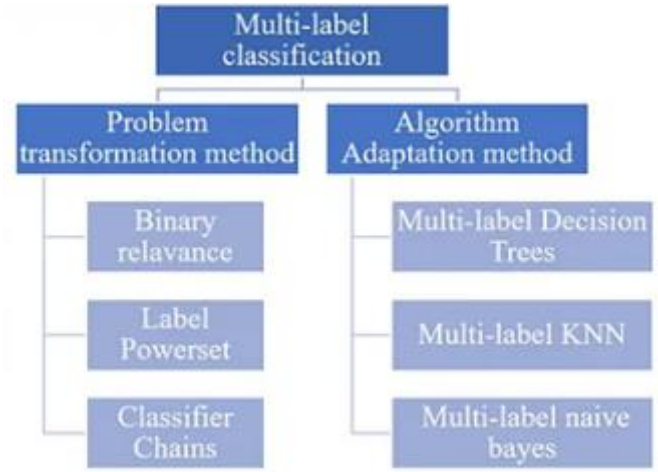
Εικόνα 2: Παράδειγμα των διαφορετικών ειδών κατηγοριοποίησης

[Πηγή: https://www.mathworks.com]

Η επιλογή της κατάλληλης προσέγγισης εξαρτάται από τη φύση των δεδομένων, τη συσχέτιση μεταξύ των ετικετών, την επιθυμητή ακρίβεια και την υπολογιστική πολυπλοκότητα. Η κατηγοριοποίηση δεδομένων σε πλαίσια, είτε σε μοναδικής ετικέτας είτε σε πολλαπλών ετικετών, αποτελεί κρίσιμο εργαλείο για πληθώρα εφαρμογών, όπως η αναγνώριση εικόνας, η επεξεργασία φυσικής γλώσσας, και η ανάλυση συναισθήματος.

1.2 Κατηγοριοποίηση Πολλαπλών Ετικετών

Η κατηγοριοποίηση πολλαπλών ετικετών (multi-label classification) αποτελεί ένα εξειδικευμένο πρόβλημα εποπτευόμενης μάθησης, όπου κάθε δείγμα μπορεί να αντιστοιχίζεται σε περισσότερες από μία ετικέτες. Σε αντίθεση με την δυαδική ταξινόμηση ή την ταξινόμηση πολλαπλών κλάσεων, όπου η έξοδος είναι μία και μοναδική κλάση ανά δείγμα, στην ταξινόμηση πολλαπλών ετικετών επιτρέπεται η ταυτόχρονη απόδοση πολλών κατηγοριών [7]. Τέτοιου είδους προβλήματα εμφανίζονται σε εφαρμογές όπως η αυτόματη κατηγοριοποίηση εγγράφων, η ανάλυση συναισθήματος, η αναγνώριση αντικειμένων σε εικόνες και η βιοπληροφορική.



Εικόνα 3: Προσεγγίσεις στην επίλυση προβλημάτων για ταξινόμηση πολλαπλών ετικετών

Οι μέθοδοι μετασχηματισμού προβλήματος (Problem transformation methods) αποτελούν προσεγγίσεις οι οποίες χρησιμοποιούνται για να μετατρέψουν ένα πρόβλημα πολλαπλών ετικετών - δηλαδή ένα πρόβλημα όπου κάθε παρατήρηση μπορεί να ανήκει σε περισσότερες από μία κατηγορίες ταυτόχρονα - σε ένα ή περισσότερα κλασικά προβλήματα ταξινόμησης. Ο στόχος είναι να καταστήσουμε ένα multi-label πρόβλημα περισσότερο εύχρηστο για παραδοσιακούς αλγορίθμους ταξινόμησης οι οποίοι χειρίζονται συνήθως μόνο μία κατηγορία ανά παρατήρηση.

Από την άλλη πλευρά, οι μέθοδοι προσαρμογής αλγορίθμων (Algorithm Adaptation methods) επεκτείνουν υπάρχοντες αλγορίθμους προκειμένου να διαχειριστούν απευθείας δεδομένα με πολλαπλές ετικέτες. Ενδεικτικά παραδείγματα τέτοιων αλγορίθμων αποτελούν ο Multi-label Decision Trees, ο Multi-label k-Nearest Neighbors (KNN) και ο Multi-label Naive Bayes [8].

Στην κατηγορία των μεθόδων προσαρμογής αλγορίθμων εντάσσονται παραλλαγές των κλασικών αλγορίθμων, όπως ο Multi-label k-Nearest Neighbors (MLkNN), ο BRkNN (Binary Relevance kNN), ο MLARAM (Multi-label Learning with Adaptive Resonance Associative Map), καθώς και ο MLTSVM (Multi-label Twin Support Vector Machine). Αντίθετα, αλγόριθμοι όπως το Random Forest συνήθως χρησιμοποιούνται στο πλαίσιο των μεθόδων μετασχηματισμού προβλήματος (Problem Transformation methods), όπως για παράδειγμα στη μέθοδο Binary Relevance ή Classifier Chains, όπου κάθε ετικέτα μετατρέπεται σε ξεχωριστό δυαδικό πρόβλημα και μοντελοποιείται με έναν ανεξάρτητο ταξινομητή τυχαίου δάσους.

Επιπρόσθετα, πέρα από τις μεθόδους μετασχηματισμού προβλήματος και τις προσαρμογές αλγορίθμων, έχει αναπτυχθεί μια τρίτη κατηγορία προσεγγίσεων που βασίζεται σε τεχνικές ενσωμάτωσης ετικετών (label embeddings).

Η ενσωμάτωση ετικετών αποτελεί μία μέθοδο κατά την οποία οι ετικέτες ενός προβλήματος μετατρέπονται σε διανύσματα (vectors) εντός ενός συνεχούς, μαθηματικού χώρου (embedding space). Στόχος αυτής της τεχνικής είναι να αποτυπώσει τις σχέσεις μεταξύ των ετικετών, όπως για παράδειγμα να αποτυπώσει ποιες ετικέτες εμφανίζονται συχνά μαζί, ποιες είναι πιο σχετικές μεταξύ τους, κ.λπ. Όταν αυτό αποδοθεί σωστά στον διανυσματικό χώρο, το μοντέλο μπορεί να γενικεύει καλύτερα, ειδικά σε περιπτώσεις όπου ορισμένες ετικέτες είναι σπάνιες ή συνδέονται με άλλες.

Στα πλαίσια της παρούσας εργασίας θα χρησιμοποιήσουμε την τεχνική Node2Vec. Η τεχνική αυτή εφαρμόζεται σε γράφους (graphs), στους οποίους οι ετικέτες αναπαρίστανται ως κόμβοι και οι σχέσεις συνεμφάνισης ως ακμές. Μέσω στοχαστικών διερευνήσεων (random walks), η τεχνική Node2Vec μαθαίνει διανυσματικές αναπαραστάσεις για κάθε κόμβο/ετικέτα, οι οποίες μπορούν να ενσωματωθούν στο κύριο μοντέλο μάθησης [9]. Αντίστοιχη προσέγγιση υπάρχει και στην επεξεργασία φυσικής γλώσσας με τις ενσωματώσεις λέξεων (word embeddings) και την τεχνική Word2Vec.

Οι τεχνικές αυτές επιδιώκουν να ενσωματώσουν τις ετικέτες σε έναν συνεχές διανυσματικό χώρο, αποτυπώνοντας τις συσχετίσεις και τις εξαρτήσεις μεταξύ τους. Με τον τρόπο αυτό, η διαδικασία ταξινόμησης δεν περιορίζεται σε δυαδικές ή ανεξάρτητες προβλέψεις ανά ετικέτα, αλλά αξιοποιεί τη γεωμετρική και σημασιολογική πληροφορία του ενσωματωμένου χώρου. Η χρήση ενσωματώσεων επιτρέπει την καλύτερη κατανόηση των σχέσεων μεταξύ ετικετών και μπορεί να ενισχύσει την απόδοση σε datasets με πολύπλοκες ή σπάνιες ετικέτες.

1.3 Κίνητρο

Η κατηγοριοποίηση πολλαπλών ετικετών έχει προσελκύσει σημαντικό ερευνητικό ενδιαφέρον τα τελευταία χρόνια, οδηγώντας στην ανάπτυξη πλήθους αλγορίθμων και προσεγγίσεων που ανταποκρίνονται σε διαφορετικά είδη δεδομένων και εφαρμογές [3,4]. Ωστόσο, παρά την πληθώρα των διαθέσιμων μεθόδων, δεν υπάρχει ακόμη ευρεία και συστηματική σύγκριση των αλγορίθμων σε πολλαπλά πραγματικά σύνολα δεδομένων, υπό ενιαίο πειραματικό πλαίσιο και με κοινές μετρικές αξιολόγησης.

Οι περισσότεροι αλγόριθμοι που έχουν προταθεί βασίζονται είτε σε μετασχηματισμούς του προβλήματος, όπως οι BRkNN και MLkNN, είτε σε πιο σύνθετες μεθόδους βασισμένες σε θεωρίες μνήμης και όγκου πληροφορίας, όπως οι MLARAM και MLTSVM. Επιπλέον, η εισαγωγή τεχνικών ενσωμάτωσης ετικετών με τη βοήθεια αλγορίθμων όπως ο Node2Vec έχει δημιουργήσει νέα ερευνητικά ερωτήματα σχετικά με την αποτελεσματικότητά τους ως προς την ακρίβεια και την αποδοτικότητα [9].

Η παρούσα εργασία στοχεύει να καλύψει αυτό το κενό, πραγματοποιώντας μια πειραματική σύγκριση πέντε δημοφιλών αλγορίθμων κατηγοριοποίησης πολλαπλών ετικετών (BRkNN, MLkNN, Random Forest , MLARAM και MLTSVM) σε οκτώ διαφορετικά πραγματικά σύνολα δεδομένων. Για κάθε αλγόριθμο υπολογίζονται οι μετρικές Hamming Loss, Accuracy και CPU Time, τόσο δίχως τη χρήση τεχνικών ενσωμάτωσης, όσο και μετά την ενσωμάτωση ετικετών με τον αλγόριθμο Node2Vec.

Ο στόχος της εργασίας είναι διττός. Αφενός, να καταγράψει τις συγκριτικές επιδόσεις των αλγορίθμων σε ετερογενή δεδομένα και αφετέρου να αξιολογήσει την επίδραση της ενσωμάτωσης ετικετών στην ακρίβεια και στην υπολογιστική αποδοτικότητα των ταξινομητών.

1.4 Οργάνωση της εργασίας

Η παρούσα εργασία οργανώνεται σε πέντε κεφάλαια, συνολικά. Στο Κεφάλαιο 2 παρουσιάζονται οι βασικές θεωρητικές έννοιες και μέθοδοι που σχετίζονται με την κατηγοριοποίηση πολλαπλών ετικετών. Αρχικά περιγράφονται οι τεχνικές μετασχηματισμού προβλήματος (Binary Relevance, Classifier Chain, Label Powerset), ενώ στη συνέχεια αναλύονται εξειδικευμένοι αλγόριθμοι κατηγοριοποίησης (όπως οι BRkNN, MLkNN, MLARAM, MLTSVM). Ακολουθεί ενότητα αφιερωμένη στη χρήση του αλγορίθμου Random Forest για προβλήματα πολλαπλών ετικετών, καθώς και παρουσίαση της προσέγγισης της ενσωμάτωσης ετικετών με χρήση του Node2Vec. Το κεφάλαιο ολοκληρώνεται με την αποτίμηση της απόδοσης των μοντέλων μέσω μετρικών όπως το Hamming Loss, η ακρίβεια (Accuracy) και ο υπολογιστικός χρόνος (CPU time).

Το Κεφάλαιο 3 επικεντρώνεται στην υλοποίηση των επιλεγμένων αλγορίθμων χρησιμοποιώντας τη βιβλιοθήκη scikit - multilearn, με παρουσίαση χαρακτηριστικών τμημάτων του κώδικα και των βασικών ρυθμίσεων.

Στο Κεφάλαιο 4 περιγράφεται η πειραματική διαδικασία. Αρχικά παρουσιάζονται τα σύνολα δεδομένων που χρησιμοποιήθηκαν και τα βασικά χαρακτηριστικά τους ,όπως η πυκνότητα και η καρδινάλιοτητα (cardinality) των ετικετών. Ακολουθούν τα πειραματικά αποτελέσματα, με σχολιασμό της απόδοσης κάθε αλγορίθμου ανά dataset, καθώς και συγκρίσεις με βάση τους μέσους όρους των μετρικών. Τέλος, πραγματοποιείται μια συνολική συζήτηση πάνω στα ευρήματα της μελέτης.

Το Κεφάλαιο 5 περιλαμβάνει τα τελικά συμπεράσματα της εργασίας, με αναφορές στα πειραματικά αποτελέσματα και τη γενική αποδοτικότητα των μεθόδων. Το κεφάλαιο ολοκληρώνεται με προτάσεις για μελλοντική έρευνα, εστιάζοντας σε πιθανές βελτιώσεις και επεκτάσεις των μεθόδων που μελετήθηκαν.

Κεφάλαιο 2ο: Αλγόριθμοι και Τεχνικές Κατηγοριοποίησης

2.1 Μέθοδοι Μετασχηματισμού Προβλήματος

Οι μέθοδοι μετασχηματισμού προβλήματος (problem transformation methods) αποτελούν μια σημαντική προσέγγιση στην κατηγοριοποίηση πολλαπλών ετικετών. Η βασική ιδέα είναι η μετατροπή του αρχικού προβλήματος πολλαπλών ετικετών σε ένα ή περισσότερα προβλήματα απλής κατηγοριοποίησης (single-label classification), τα οποία είναι ευκολότερα στη διαχείριση και ανάλυση με χρήση συμβατικών αλγορίθμων μηχανικής μάθησης.

Μία από τις απλούστερες και πλέον διαδεδομένες μεθόδους είναι η Binary Relevance (BR), κατά την οποία κάθε ετικέτα αντιμετωπίζεται ως ανεξάρτητο πρόβλημα δυαδικής κατηγοριοποίησης [8]. Η μέθοδος αυτή έχει χαμηλό υπολογιστικό κόστος και μπορεί να εφαρμοστεί εύκολα με οποιονδήποτε αλγόριθμο δυαδικής ταξινόμησης. Ωστόσο, παρουσιάζει σημαντικό μειονέκτημα την αγνόηση της συσχέτισης μεταξύ των ετικετών, γεγονός που μπορεί να μειώσει την ακρίβεια των αποτελεσμάτων σε περιπτώσεις όπου οι ετικέτες παρουσιάζουν εξάρτηση μεταξύ τους [6].

Για την αντιμετώπιση αυτής της αδυναμίας προτάθηκε η μέθοδος Classifier Chains (CC), η οποία επεκτείνει την BR με την εισαγωγή αλληλεξαρτήσεων μεταξύ των ετικετών. Στην CC, κάθε δυαδικός ταξινομητής εκπαιδεύεται με βάση όχι μόνο τα χαρακτηριστικά εισόδου, αλλά και τις προβλέψεις των προηγούμενων ταξινομητών στην αλυσίδα [10]. Η σειρά των ετικετών παίζει κρίσιμο ρόλο στην απόδοση της μεθόδου, και συχνά χρησιμοποιείται τυχαία πολλαπλή αλυσίδα (ensemble of random chains) για τη βελτίωση της γενίκευσης [11].

Η τρίτη κλασική μέθοδος μετασχηματισμού προβλήματος είναι η Label Powerset (LP), η οποία μετασχηματίζει ένα πρόβλημα πολλαπλών ετικετών σε ένα πρόβλημα ταξινόμησης πολλαπλών κλάσεων (multi-class classification) όπου κάθε μοναδικός συνδυασμός ετικετών αντιμετωπίζεται ως ξεχωριστή κλάση [7]. Αν και η μέθοδος LP διατηρεί πλήρως τις εξαρτήσεις μεταξύ των ετικετών, παρουσιάζει προβλήματα κλιμάκωσης, καθώς ο αριθμός των πιθανών συνδυασμών αυξάνεται εκθετικά με τον αριθμό των ετικετών. Ως εκ τούτου, η μέθοδος LP παρουσιάζει τάση υπερπροσαρμογής (overfitting) σε περιπτώσεις που το σύνολο των δεδομένων περιλαμβάνει σπάνιους ή μοναδικούς συνδυασμούς ετικετών, γεγονός που ενδέχεται να περιορίσει την ικανότητά της να γενικεύει αποτελεσματικά σε νέα δεδομένων [12].

Πίνακας 1: Συγκριτικός Πίνακας Μεθόδων Μετασχηματισμού Προβλήματος

Κριτήριο	Binary Relevance (BR)	Label Powerset (LP)	Classifier Chains (CC)
Αρχή λειτουργίας	Μετατρέπει το πρόβλημα σε *L ανεξάρτητα δυαδικά προβλήματα*, ένα για κάθε ετικέτα.	Αντιμετωπίζει κάθε *μοναδικό συνδυασμό ετικετών* ως ξεχωριστή κλάση ενός προβλήματος ταξινόμησης Πολλαπλών κλάσεων.	Χρησιμοποιεί *αλυσίδα δυαδικών ταξινομητών*, όπου κάθε ταξινομητής λαμβάνει ως είσοδο και τις προβλέψεις των προηγούμενων.
Συσχετίσεις μεταξύ ετικετών	Αγνοούνται (θεωρούνται ανεξάρτητες).	Λαμβάνονται πλήρως υπόψη, αφού κάθε συνδυασμός αποτελεί νέα ετικέτα.	Λαμβάνονται μερικώς υπόψη, μέσω της χρήσης των προβλέψεων προηγούμενων ετικετών.
Πολυπλοκότητα	Χαμηλή – απλή υλοποίηση, εύκολη παραλληλοποίηση.	Υψηλή – εκθετική αύξηση του αριθμού κλάσεων με τον αριθμό ετικετών.	Μέτρια – εξαρτάται από το μήκος της αλυσίδας και τη σειρά των ετικετών.

Κριτήριο	Binary Relevance (BR)	Label Powerset (LP)	Classifier Chains (CC)
Κίνδυνος υπερπροσαρμογής (overfitting)	Μικρός, αλλά περιορισμένη ακρίβεια λόγω απώλειας συσχετίσεων.	**Μεγάλος**, ειδικά όταν υπάρχουν σπάνιοι συνδυασμοί ετικετών.	Μέτριος – σχετίζεται με τη σειρά ταξινομητών και την ποιότητα των ενδιάμεσων προβλέψεων.
Απόδοση σε δεδομένα με εξαρτώμενες ετικέτες	Χαμηλή έως μέτρια.	Υψηλή, αλλά εξαρτάται από την πυκνότητα των ετικετών.	Συχνά υψηλότερη από BR, χωρίς τα προβλημάτα της LP.
Ευκολία υλοποίησης	Πολύ εύκολη.	Πιο περίπλοκη, λόγω ανάγκης διαχείρισης πλήθους κλάσεων.	Απαιτεί περισσότερη φροντίδα στην υλοποίηση και πειραματισμό με τη σειρά των ετικετών.

Η επιλογή της κατάλληλης μεθόδου μετασχηματισμού εξαρτάται από τα χαρακτηριστικά του συνόλου δεδομένων και τις απαιτήσεις της εφαρμογής. Συνήθως, η BR είναι κατάλληλη για μεγάλα σύνολα με σχετικά ανεξάρτητες ετικέτες, ενώ η CC και η LP προτιμώνται όταν υπάρχουν εμφανείς εξαρτήσεις μεταξύ των ετικετών που πρέπει να ληφθούν υπόψη για την επίτευξη αποτελεσμάτων υψηλής ακρίβειας.

2.1.1 Binary Relevance

Η μέθοδος Binary Relevance (BR) αποτελεί μία από τις πιο θεμελιώδεις και ευρέως χρησιμοποιούμενες τεχνικές μετασχηματισμού προβλήματος στην πολυετικετική κατηγοριοποίηση. Η βασική ιδέα πίσω από τη μέθοδο αυτή είναι η αντιμετώπιση κάθε ετικέτας ως ανεξάρτητου δυαδικού προβλήματος ταξινόμησης. Συγκεκριμένα, αν υποθέσουμε ότι υπάρχουν L διαφορετικές ετικέτες στο σύνολο δεδομένων, η BR κατασκευάζει L ανεξάρτητους ταξινομητές, καθένας εκ των οποίων εκπαιδεύεται για να προβλέψει την παρουσία ή απουσία μίας μόνο ετικέτας σε κάθε παράδειγμα εισόδου [8]. Με άλλα λόγια, κάθε ταξινομητής μαθαίνει να επιλύει ένα πρόβλημα δυαδικής κατηγοριοποίησης, χωρίς να λαμβάνει υπόψη του την πιθανή συσχέτιση με τις υπόλοιπες ετικέτες.

Η μέθοδος παρουσιάζει αρκετά πρακτικά πλεονεκτήματα. Πρωτίστως, είναι απλή στην υλοποίηση, γεγονός που την καθιστά προσιτή σε πληθώρα εφαρμογών. Επίσης, λόγω της ανεξαρτησίας των ταξινομητών, δίνεται η δυνατότητα παραλληλοποίησης της διαδικασίας εκπαίδευσης και πρόβλεψης, γεγονός που μπορεί να επιταχύνει σημαντικά την υπολογιστική διαδικασία, ειδικά σε μεγάλα σύνολα δεδομένων. Επιπλέον, η BR είναι ευέλικτη, καθώς μπορεί να αξιοποιήσει οποιονδήποτε συμβατικό αλγόριθμο δυαδικής ταξινόμησης, καθιστώντας τη μέθοδο εύκολα προσαρμόσιμη σε διαφορετικά είδη προβλημάτων.

Ωστόσο, η Binary Relevance συνοδεύεται από έναν σημαντικό περιορισμό, ο οποίος πηγάζει από τη βασική της υπόθεση: την ανεξαρτησία των ετικετών μεταξύ τους. Στην πράξη, οι ετικέτες σε προβλήματα κατηγοριοποίησης πολλαπλών ετικετών παρουσιάζουν συχνά συσχετίσεις ή αλληλεξαρτήσεις (π.χ. σε ένα ιατρικό σύστημα διάγνωσης, η παρουσία μιας ασθένειας μπορεί να υποδηλώνει αυξημένη πιθανότητα ύπαρξης άλλης). Η μη ενσωμάτωση αυτών των εξαρτήσεων στο μοντέλο μάθησης ενδεχομένως να οδηγήσει σε υποβάθμιση της συνολικής απόδοσης, καθώς η μέθοδος αδυνατεί να αξιοποιήσει σημαντική πληροφορία από τη συνεμφάνιση ετικετών [6,13].

2.1.2 Classifier Chain

Η μέθοδος Classifier Chain (CC) προτάθηκε ως βελτιωμένη επέκταση της προσέγγισης της μεθόδου Binary Relevance (BR), με στόχο την ενσωμάτωση της αλληλεξάρτησης μεταξύ των ετικετών σε προβλήματα κατηγοριοποίησης πολλαπλών ετικετών [10]. Σε αντίθεση με τη BR, όπου κάθε ετικέτα αντιμετωπίζεται εντελώς ανεξάρτητα, η CC επιδιώκει να μοντελοποιήσει τις πιθανές συσχετίσεις μεταξύ τους.

Στην πράξη, η CC οργανώνει τους δυαδικούς ταξινομητές σε μια αλυσίδα, με τέτοιο τρόπο ώστε κάθε ταξινομητής να λαμβάνει ως είσοδο όχι μόνο το αρχικό σύνολο χαρακτηριστικών του δείγματος, αλλά και τις προβλέψεις των προηγούμενων ταξινομητών στην αλυσίδα. Με τον τρόπο αυτό, κάθε μοντέλο έχει τη δυνατότητα να ενσωματώσει πληροφορία από τις προηγούμενες ετικέτες στη διαδικασία λήψης απόφασης για την τρέχουσα. Αυτό οδηγεί σε μεγαλύτερη ικανότητα έκφρασης συσχετισμών μεταξύ των ετικετών, η οποία συχνά μεταφράζεται σε βελτιωμένη ακρίβεια ταξινόμησης, ιδίως σε περιπτώσεις όπου οι ετικέτες εμφανίζουν εξαρτήσεις.

Ωστόσο, ένα βασικό μειονέκτημα της μεθόδου έγκειται στο γεγονός ότι τα αποτελέσματά της εξαρτώνται από τη σειρά με την οποία οργανώνονται οι ετικέτες στην αλυσίδα. Διαφορετικές σειρές μπορεί να οδηγήσουν σε διαφορετικές επιδόσεις, καθώς η πληροφορία από κρίσιμες ετικέτες ενδέχεται να μην είναι διαθέσιμη εγκαίρως κατά την πρόβλεψη. Για να μετριαστεί αυτό το πρόβλημα, έχουν προταθεί παραλλαγές όπως ο Random Classifier Chains (RCC) ή ο Ensemble of Classifier Chains (ECC), όπου κατασκευάζονται πολλαπλές αλυσίδες με τυχαίες ή διαφορετικές σειρές ετικετών και τα αποτελέσματά τους συνδυάζονται, συνήθως μέσω ψηφοφορίας ή μέσου όρου [14]. Αυτές οι τεχνικές ενισχύουν τη σταθερότητα και τη γενίκευση του τελικού μοντέλου.

2.1.3 Label Powerset

Η μέθοδος Label Powerset (LP) αντιμετωπίζει το πρόβλημα πολλαπλών ετικετών ως πρόβλημα πολυκατηγορικής ταξινόμησης, όπου κάθε διαφορετικός συνδυασμός ετικετών θεωρείται ως μία ξεχωριστή κλάση [7]. Αυτή η προσέγγιση επιτρέπει στο μοντέλο να λάβει υπόψη όλες τις εξαρτήσεις μεταξύ των ετικετών.

Αν και η LP θεωρητικά παρέχει υψηλότερη ικανότητα μοντελοποίησης, παρουσιάζει πρακτικά προβλήματα όταν ο αριθμός των μοναδικών συνδυασμών ετικετών είναι μεγάλος, με αποτέλεσμα την εκθετική αύξηση των κλάσεων και την πρόκληση σπάνιων περιπτώσεων (class imbalance) [15]. Για την αντιμετώπιση αυτού του ζητήματος έχουν προταθεί παραλλαγές όπως η RAkEL (Random k-Labelsets), η οποία χωρίζει το σύνολο των ετικετών σε μικρότερα υποσύνολα και εφαρμόζει την LP σε καθένα από αυτά [12].

Η μέθοδος Label Powerset (LP) προσεγγίζει το πρόβλημα της ταξινόμησης πολλαπλών ετικετών μετασχηματίζοντάς το σε πρόβλημα ταξινόμησης πολλαπλών κλάσεων (multi-class classification) [7]. Συγκεκριμένα, κάθε μοναδικός συνδυασμός ετικετών που εμφανίζεται στο σύνολο δεδομένων θεωρείται ως μία διακριτή κλάση. Για παράδειγμα, εάν ένα δείγμα φέρει τις ετικέτες {A, B}, και ένα άλλο τις {B, C}, τότε οι συνδυασμοί αυτοί αντιμετωπίζονται ως διαφορετικές κλάσεις, ανεξαρτήτως του αν περιλαμβάνουν κοινές ετικέτες.

Η κύρια ισχύς της LP έγκειται στη ρητή μοντελοποίηση των εξαρτήσεων μεταξύ των ετικετών. Σε αντίθεση με μεθόδους όπως η Binary Relevance (BR), όπου οι ετικέτες θεωρούνται ανεξάρτητες, η LP λαμβάνει υπόψη πλήρως τις συσχετίσεις και συγχρονισμένες εμφανίσεις των ετικετών σε ένα ενιαίο

πλαίσιο πρόβλεψης. Αυτό επιτρέπει στο μοντέλο να μάθει πολύπλοκες σχέσεις και μοτίβα μεταξύ των ετικετών, οδηγώντας δυνητικά σε υψηλότερη ακρίβεια σε περιβάλλοντα όπου τέτοιες εξαρτήσεις είναι κρίσιμες.

Ωστόσο, παρά τα θεωρητικά πλεονεκτήματα, η μέθοδος παρουσιάζει πρακτικά προβλήματα κλιμάκωσης. Ο αριθμός των κλάσεων αυξάνεται εκθετικά με τον αριθμό των ετικετών, καθώς κάθε δυνατός συνδυασμός μπορεί να αποτελέσει νέα κλάση. Αυτό οδηγεί όχι μόνο σε μεγάλο αριθμό κλάσεων, αλλά και σε φαινόμενα ανομοιόμορφης κατανομής των δεδομένων (class imbalance), αφού πολλοί συνδυασμοί εμφανίζονται σπάνια ή ακόμη και μία μόνο φορά στο σύνολο εκπαίδευσης. Το γεγονός αυτό καθιστά τη μέθοδο επιρρεπή σε υπερπροσαρμογή (overfitting), καθώς το μοντέλο ενδέχεται να προσαρμοστεί σε σπάνιες ή μοναδικές περιπτώσεις που δεν γενικεύονται καλά σε νέα δεδομένα.

Για την αντιμετώπιση αυτών των περιορισμών, έχουν αναπτυχθεί βελτιωμένες παραλλαγές, με πιο γνωστή τη μέθοδο RAkEL (Random k-Labelsets). Η RAkEL επιδιώκει να μειώσει την πολυπλοκότητα της LP διαιρώντας το αρχικό σύνολο ετικετών σε πολλαπλά τυχαία επιλεγμένα υποσύνολα (labelsets) μεγέθους k. Η LP εφαρμόζεται ανεξάρτητα σε κάθε τέτοιο υποσύνολο, και τα αποτελέσματα από όλες τις υπο-προβλέψεις συνδυάζονται ώστε να παραχθεί η τελική εκτίμηση [12]. Με αυτόν τον τρόπο, η RAkEL ισορροπεί μεταξύ της μοντελοποίησης εξαρτήσεων και της διαχειρίσιμης πολυπλοκότητας, προσφέροντας καλύτερη γενίκευση και μειωμένο υπολογιστικό κόστος.

2.2 Αλγόριθμοι κατηγοριοποίησης πολλαπλών ετικετών

Οι αλγόριθμοι οι οποίοι εντάσσονται στην κατηγορία των μεθόδων προσαρμογής (algorithm adaptation methods) συνιστούν μία διαφορετική προσέγγιση κατηγοριοποίησης πολλαπλών ετικετών σε σχέση με τις τεχνικές μετασχηματισμού προβλήματος. Σε αντίθεση με τις τελευταίες, οι οποίες αναδιατυπώνουν ένα πρόβλημα πολλαπλών ετικετών σε μία ή περισσότερες μορφές γνωστών εποπτευόμενων προβλημάτων (όπως δυαδική ταξινόμηση ή ταξινόμηση πολλαπλών κλάσεων), οι μέθοδοι προσαρμογής επιχειρούν να επεκτείνουν άμεσα τους υπάρχοντες αλγορίθμους μάθησης ώστε να χειρίζονται άμεσα, χωρίς ενδιάμεση αναδιατύπωση του προβλήματος, τα δεδομένων με πολλαπλές ετικέτες.

Στην κατηγορία αυτή, οι αλγόριθμοι είναι δομικά σχεδιασμένοι ώστε να υποστηρίζουν την πρόβλεψη πολλαπλών ετικετών ταυτόχρονα για κάθε παρατήρηση, αξιοποιώντας τις συσχετίσεις και τις συνυπάρξεις που εντοπίζονται μεταξύ των ετικετών. Αντί να αντιμετωπίζουν τις ετικέτες ως ανεξάρτητες μεταβλητές, αυτές οι μέθοδοι επιδιώκουν να μοντελοποιήσουν τις αλληλεπιδράσεις μεταξύ τους, γεγονός που μπορεί να οδηγήσει σε βελτιωμένη απόδοση ταξινόμησης, ειδικά σε περιβάλλοντα όπου οι ετικέτες εμφανίζουν υψηλό βαθμό αλληλεξάρτησης.

Επιπλέον, οι μέθοδοι προσαρμογής προσφέρουν καλύτερη ενσωμάτωση της πολυπλοκότητας των δεδομένων, χωρίς την ανάγκη τεχνητής διάσπασης ή ανακατασκευής του προβλήματος. Χάρη σε αυτή τη δυνατότητα, οι εν λόγω αλγόριθμοι είναι σε θέση να επιτύχουν υψηλότερη ακρίβεια, καλύτερη γενίκευση, και πιο συνεπείς προβλέψεις σε σύγκριση με τις μεθόδους μετασχηματισμού, ιδίως όταν οι εξαρτήσεις μεταξύ των ετικετών παίζουν κρίσιμο ρόλο.

Στις ενότητες που ακολουθούν, παρουσιάζονται τέσσερις αντιπροσωπευτικοί αλγόριθμοι της κατηγορίας των μεθόδων προσαρμογής, οι οποίοι έχουν χρησιμοποιηθεί εκτενώς στη βιβλιογραφία και προσφέρουν διαφορετικές προσεγγίσεις στην πρόβλεψη με πολλαπλές ετικέτες.

2.2.1 BRkNN

Ο αλγόριθμος BRkNN (Binary Relevance k-Nearest Neighbors) αποτελεί μία χαρακτηριστική μέθοδο προσαρμογής αλγορίθμου για την κατηγοριοποίηση πολλαπλών ετικετών, η οποία στηρίζεται στη φιλοσοφία της ταξινόμησης βάσει γειτόνων. Η τεχνική αυτή προκύπτει από τον συνδυασμό της μεθόδου Binary Relevance (BR) και του κλασικού αλγορίθμου k-Nearest Neighbors (kNN), επεκτείνοντας έτσι τη χρήση του τελευταίου στην επίλυση προβλημάτων με πολλαπλές ετικέτες.

Στην προσέγγιση του BRkNN, για κάθε μία από τις L ετικέτες δημιουργείται ένας ανεξάρτητος δυαδικός ταξινομητής, σύμφωνα με την αρχή του Binary Relevance. Ωστόσο, αντί να χρησιμοποιείται κάποιος γενικός γραμμικός ή δενδροειδής ταξινομητής, η πρόβλεψη βασίζεται στην ανάλυση των τοπικών γειτόνων: για κάθε νέο δείγμα, προσδιορίζονται οι k κοντινότεροι γείτονες στον χώρο των χαρακτηριστικών, και για κάθε ετικέτα ελέγχεται η συχνότητα εμφάνισής της ανάμεσα σε αυτούς τους γείτονες. Εάν η συχνότητα υπερβαίνει ένα προκαθορισμένο κατώφλι, η ετικέτα θεωρείται παρούσα· διαφορετικά, απορρίπτεται [16]. Με αυτόν τον τρόπο, το μοντέλο συνυπολογίζει τα γειτονικά δείγματα προκειμένου να διαμορφώσει την τελική του απόφαση.

Η μέθοδος ξεχωρίζει για την απλότητα υλοποίησης και την ικανοποιητική απόδοση που παρουσιάζει σε πολλές περιπτώσεις, ιδίως όταν οι ετικέτες είναι σχετικά ανεξάρτητες ή όταν τα δεδομένα εμφανίζουν σαφώς διαχωρισμένες περιοχές στο χώρο χαρακτηριστικών. Επιπλέον, η μη παραμετρική φύση του kNN επιτρέπει στο μοντέλο να προσαρμόζεται με ευελιξία σε διαφορετικές κατανομές δεδομένων, χωρίς την ανάγκη εκτενούς εκπαίδευσης.

Ωστόσο, ένα βασικό μειονέκτημα του αλγορίθμου BRkNN εντοπίζεται στην αδυναμία του να συμπεριλάβει τις εξαρτήσεις μεταξύ ετικετών, καθώς κάθε δυαδικός ταξινομητής λειτουργεί εντελώς ανεξάρτητα από τους υπόλοιπους. Αυτή η παραδοχή μπορεί να οδηγήσει σε μειωμένη απόδοση, ειδικά σε εφαρμογές όπου οι ετικέτες είναι ισχυρά συσχετισμένες ή εμφανίζονται κατά ομάδες (π.χ. στη θεματική κατηγοριοποίηση κειμένων ή στην ανάλυση συναισθήματος πολλαπλών επιπέδων).

Παρά τις αδυναμίες του, ο αλγόριθμος BRkNN αποτελεί μία χρήσιμη και ερμηνεύσιμη επιλογή για προβλήματα με πολλαπλές ετικέτες, ιδίως όταν απαιτείται ταχεία εφαρμογή χωρίς εκτενή εκπαίδευση ή σε περιβάλλοντα όπου η συσχέτιση μεταξύ των ετικετών δεν είναι κρίσιμη για την τελική πρόβλεψη.

2.2.2 MLkNN

Ο αλγόριθμος MLkNN (Multi-Label k-Nearest Neighbors) αποτελεί μία από τις πιο διαδεδομένες και αξιόπιστες τεχνικές για την αντιμετώπιση προβλημάτων ταξινόμησης πολλαπλών ετικετών, επεκτείνοντας τη βασική λογική του κλασικού αλγορίθμου k-Nearest Neighbors (kNN) με τη χρήση πιθανολογικής συλλογιστικής. Σε αντίθεση με τον BRkNN, ο οποίος εφαρμόζει ξεχωριστούς δυαδικούς ταξινομητές χωρίς να λαμβάνει υπόψη τη συσχέτιση των ετικετών, ο MLkNN ενσωματώνει ένα μοντέλο πιθανοτήτων βασισμένο στο θεώρημα του Bayes, το οποίο αξιοποιεί την πληροφορία των γειτονικών παραδειγμάτων για να εκτιμήσει την πιθανότητα εμφάνισης κάθε ετικέτας.

Συγκεκριμένα, για κάθε νέο δείγμα, ο MLkNN εντοπίζει τους k πλησιέστερους γείτονες και καταγράφει για κάθε ετικέτα τον αριθμό των γειτόνων στους οποίους αυτή εμφανίζεται. Στη συνέχεια, με βάση αυτή

τη συχνότητα και χρησιμοποιώντας προηγούμενες κατανομές πιθανοτήτων που έχουν υπολογιστεί από το εκπαιδευτικό σύνολο, ο αλγόριθμος εκτιμά μέσω της συμπερασματολογίας κατά Bayes (Bayesian inference) την πιθανότητα να είναι η εκάστοτε ετικέτα παρούσα ή απούσα για το συγκεκριμένο δείγμα. Η τελική απόφαση πρόβλεψης βασίζεται στη μέγιστη εκ των δύο πιθανοτήτων [17]. Το θεώρημα του Bayes ορίζεται μαθηματικά με την παρακάτω εξίσωση:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

όπου A και B είναι γεγονότα.

- P(A) και P(B) είναι οι πιθανότητες των A και B, χωρίς να υπάρχει κάποια δεδομένη συνθήκη.
- P(A | B), η υπό συνθήκη πιθανότητα, είναι η πιθανότητα του A δεδομένου του B να είναι αληθής.
- P(B | A), είναι η πιθανότητα του B δεδομένου του A να είναι αληθής.

Η μέθοδος πλεονεκτεί σημαντικά σε περιπτώσεις όπου υπάρχουν έντονες συσχετίσεις μεταξύ των ετικετών, καθώς η πιθανολογική προσέγγιση επιτρέπει την αξιοποίηση της πληροφορίας των συνθηκών εμφάνισης των ετικετών με τρόπο που δεν είναι εφικτός σε πιο απλές μεθόδους όπως η Binary Relevance. Ως εκ τούτου, ο MLkNN θεωρείται ιδιαίτερα αποδοτικός σε εφαρμογές όπως η κατηγοριοποίηση κειμένων, η επισήμανση εικόνων και η ανάλυση συναισθήματος, όπου η συνύπαρξη ετικετών είναι συχνή και σημαντική για την ακρίβεια του μοντέλου.

Ωστόσο, η χρήση πιθανολογικών εκτιμήσεων συνεπάγεται αυξημένη υπολογιστική πολυπλοκότητα, ειδικά σε περιπτώσεις όπου ο αριθμός των δειγμάτων ή των ετικετών είναι μεγάλος. Ο υπολογισμός πιθανοτήτων για κάθε ετικέτα και για κάθε δείγμα αυξάνει σημαντικά τον χρόνο επεξεργασίας, ενώ η ανάγκη αποθήκευσης πιθανοτήτων ανά συχνότητα εμφάνισης απαιτεί σημαντικούς πόρους μνήμης. Παρ' όλα αυτά, ο MLkNN εξακολουθεί να συγκαταλέγεται στις πλέον αξιόπιστες μεθόδους πολυετικετικής μάθησης, ιδιαίτερα όταν η ακρίβεια πρόβλεψης υπερισχύει της ανάγκης για απόδοση σε πραγματικό χρόνο.

2.2.3 MLARAM

Ο αλγόριθμος MLARAM (Multi-Label Adaptive Resonance Associative Map) βασίζεται στη θεωρία Adaptive Resonance Theory (ART), η οποία αποτελεί ένα γνωστικό μοντέλο από τον χώρο των τεχνητών νευρωνικών δικτύων, σχεδιασμένο για να επιτυγχάνει σταθερή, σταδιακή μάθηση, χωρίς καταστροφική απώλεια προηγούμενης γνώσης (catastrophic forgetting). Η αρχιτεκτονική ART προάγει τη διαρκή ενσωμάτωση νέας πληροφορίας χωρίς να διαγράφει προηγούμενη γνώση, καθιστώντας την κατάλληλη για προβλήματα όπου τα δεδομένα παρουσιάζουν μεταβαλλόμενες ή επαναλαμβανόμενες δομές.

Σε προβλήματα πολλαπλών ετικετών, ο MLARAM λειτουργεί ως συσχετιστικός ταξινομητής, όπου κάθε παράδειγμα εκπαίδευσης αποθηκεύεται σε μία μνήμη προτύπων, συνοδευόμενο από το αντίστοιχο σύνολο ετικετών του. Η προσέγγιση αυτή επιτρέπει στο σύστημα να «θυμάται» συγκεκριμένα μοτίβα

εισόδου και να τα ανακαλεί κατά την πρόβλεψη, προσομοιάζοντας τη λειτουργία της ανθρώπινης μνήμης.

Κατά τη φάση πρόβλεψης, ένα νέο πρότυπο συγκρίνεται με τα αποθηκευμένα πρότυπα στη βάση γνώσης, με τη χρήση ενός κατάλληλου μέτρου απόστασης (συνήθως Ευκλείδεια απόσταση ή Manhattan). Ο MLARAM επιλέγει την πλησιέστερη εγγραφή και προβλέπει ως τελικό αποτέλεσμα το σύνολο των ετικετών που συσχετίζεται με αυτή. Η μέθοδος αυτή δίνει έμφαση στη συγγένεια της μορφής μεταξύ των δειγμάτων και βασίζεται στην υπόθεση ότι παρόμοιες είσοδοι πρέπει να αντιστοιχούν σε παρόμοιες εξόδους.

Ο MLARAM παρουσιάζει υψηλή ακρίβεια σε προβλήματα όπου υπάρχουν επαναλαμβανόμενα ή καλά ορισμένα πρότυπα, όπως σε εφαρμογές με δεδομένα που δεν είναι αριθμητικά αλλά είναι είτε συμβολικά (π.χ. ονόματα, καταστάσεις, σύμβολα) είτε κατηγορικά, δηλαδή δεδομένα με διακριτές τιμές (π.χ. Φύλο: {Άνδρας, Γυναίκα}) δεδομένα. Σε αυτές τις περιπτώσεις ο αλγόριθμος αποδίδει ιδιαίτερα καλά, ειδικότερα όταν η κατανομή των δεδομένων παραμένει σχετικά σταθερή [18]. Επιπλέον, η προσέγγισή του δεν απαιτεί εκτεταμένη εκπαίδευση, γεγονός που τον καθιστά κατάλληλο για διαδικτυακά περιβάλλοντα ή σε περιβάλλοντα σταδιακής μάθησης (incremental learning).

Ωστόσο, η απόδοση του MLARAM μπορεί να υποβαθμιστεί σε περιπτώσεις όπου το σύνολο δεδομένων είναι πολύπλοκο, εκτεταμένο ή μη γραμμικά διαχωρίσιμο. Επειδή η μέθοδος βασίζεται αποκλειστικά στην αντιστοίχιση με αποθηκευμένα δείγματα, ενδέχεται να μην γενικεύει επαρκώς σε περιπτώσεις που απαιτούν αφηρημένη μοντελοποίηση ή αντιμετώπιση θορύβου. Παρ' όλα αύτα, ο MLARAM παραμένει μία ενδιαφέρουσα και αποτελεσματική επιλογή σε ειδικές εφαρμογές όπου η αναγνώριση μορφών και η προσπέλαση παλαιότερων παραδειγμάτων είναι κρίσιμης σημασίας.

2.2.4 MLTSVM

Ο αλγόριθμος MLTSVM (Multi-Label Twin Support Vector Machine) αποτελεί μία προηγμένη παραλλαγή του προηγούμενου αλγορίθμου Twin Support Vector Machine (TWSVM), προσαρμοσμένη για την αντιμετώπιση προβλημάτων ταξινόμησης πολλαπλών ετικετών. Ο TWSVM διαφοροποιείται από τον παραδοσιακό Support Vector Machine (SVM) κατασκευάζοντας δύο υπερεπιφάνειες (hyperplanes) αντί μίας, καθεμία από τις οποίες είναι σχεδιασμένη να βρίσκεται πλησιέστερα σε ένα σύνολο δειγμάτων και ταυτόχρονα όσο το δυνατόν πιο μακριά από το αντίθετο σύνολο. Αυτή η διττή γεωμετρική προσέγγιση οδηγεί σε έναν βελτιωμένο διαχωρισμό μεταξύ των κατηγοριών και μειωμένο υπολογιστικό κόστος κατά την επίλυση του συστήματος βελτιστοποίησης.

Η έκδοση του αλγορίθμου που αφορά σε προβλήματα πολλαπλών ετικετών, MLTSVM, επεκτείνει την παραπάνω ιδέα, δημιουργώντας πολλαπλά ζεύγη υπερεπιφανειών —ένα για κάθε ετικέτα— με τρόπο που λαμβάνει υπόψη τις πολύπλοκες συσχετίσεις και αλληλεπιδράσεις μεταξύ των ετικετών. Αντί να εξετάζει την πρόβλεψη κάθε ετικέτας ως ανεξάρτητο πρόβλημα, ο MLTSVM αξιοποιεί τις κοινές δομές στον πολυδιάστατο χώρο ετικετών, γεγονός που ενισχύει την ικανότητα γενίκευσης και τη διακριτική ισχύ του μοντέλου [19].

Η διαδικασία πρόβλεψης συνίσταται στη μέτρηση της απόστασης ενός νέου δείγματος από τα κατάλληλα κατασκευασμένα υπερεπίπεδα για κάθε ετικέτα. Η ετικέτα αποδίδεται στο δείγμα εάν το σημείο βρίσκεται «πιο κοντά» στην αντίστοιχη υπερεπιφάνεια σε σχέση με την αρνητική της. Η δομή δύο υπερεπιφανειών ανά ετικέτα επιτρέπει στον MLTSVM να χειρίζεται πολύπλοκα και μη γραμμικά

διαχωρίσιμα δεδομένα, γεγονός που καθιστά τη μέθοδο κατάλληλη για απαιτητικά σύνολα δεδομένων με πολυδιάστατη και επικαλυπτόμενη σήμανση.

Ωστόσο, παρά τα πλεονεκτήματα σε ακρίβεια και μοντελοποίηση της ετικετικής αλληλεπίδρασης, ο MLTSVM συνοδεύεται από αυξημένο υπολογιστικό κόστος, κυρίως λόγω της ανάγκης εκπαίδευσης πολλαπλών μοντέλων και της σύνθετης κατασκευής υπερεπιφανειών. Επομένως, η εφαρμογή του είναι περισσότερο ενδεδειγμένη σε περιβάλλοντα όπου η υπολογιστική επάρκεια είναι δεδομένη και η ακρίβεια πρόβλεψης αποτελεί καθοριστική απαίτηση.

2.3 Κατηγοριοποίηση με Random Forest

Ο αλγόριθμος Random Forest (RF) αποτελεί μια ευρέως χρησιμοποιούμενη τεχνική μηχανικής μάθησης για προβλήματα ταξινόμησης και παλινδρόμησης. Βασίζεται στη δημιουργία ενός συνόλου από ανεξάρτητα δέντρα απόφασης (decision trees), καθένα από τα οποία εκπαιδεύεται σε διαφορετικό υποσύνολο των δεδομένων εκπαίδευσης με χρήση της τεχνικής bootstrap sampling. Η τελική απόφαση λαμβάνεται με πλειοψηφική ψήφο (majority voting) των προβλέψεων των επιμέρους δέντρων, αυξάνοντας έτσι τη σταθερότητα και την ακρίβεια του συστήματος [20].

Στην κατηγοριοποίηση πολλαπλών ετικετών, μια συνήθης προσέγγιση είναι η εφαρμογή του Random Forest μέσω της τεχνικής της Binary Relevance (BR), όπου δημιουργείται ένας ξεχωριστός ταξινομητής για κάθε ετικέτα. Έτσι, το αρχικό πρόβλημα πολλαπλών ετικετών μετασχηματίζεται σε πολλαπλά δυαδικά προβλήματα, καθένα εκ των οποίων επιλύεται ανεξάρτητα [5]. Αν και αυτή η προσέγγιση είναι απλή και αποτελεσματική, παραβλέπει τις ενδεχόμενες αλληλεξαρτήσεις μεταξύ των ετικετών, κάτι που ενδέχεται να επηρεάσει αρνητικά την τελική απόδοση [6].

Ο Random Forest ενδείκνυται ιδιαίτερα για εργασίες κατηγοριοποίησης πολλαπλών ετικετών λόγω της ικανότητάς του να χειρίζεται δεδομένα υψηλής διάστασης, καθώς και της ανθεκτικότητάς του στον θόρυβο και την υπερεκπαίδευση [20,21].

Η χρήση του RF σε σενάρια πολλαπλών ετικετών προσφέρει σημαντικά πλεονεκτήματα, όπως:

- Ανθεκτικότητα στην υπερπροσαρμογή (overfitting) που εμφανίζουν μεμονωμένα δέντρα [20].
- Αποτελεσματική διαχείριση πολυδιάστατων και ασαφών δεδομένων [21].
- Υπολογιστικά αποδοτική εκπαίδευση και πρόβλεψη, ιδίως με τη χρήση παραλληλισμού.
- Εκτίμηση της σημασίας των χαρακτηριστικών (feature importance), κάτι που ενισχύει τη διαφάνεια και την ερμηνευσιμότητα του μοντέλου [22].

Προκειμένου να ξεπεραστεί ο περιορισμός της ανεξάρτητης επεξεργασίας των ετικετών που επιβάλλει το BR, έχουν προταθεί πιο εξελιγμένα σχήματα, όπως η μέθοδος Classifier Chains, όπου κάθε μοντέλο λαμβάνει υπόψη του τις προβλέψεις των προηγούμενων ετικετών [10]. Όταν συνδυάζονται με τον Random Forest, τα σχήματα αυτά μπορούν να ενσωματώσουν τις εξαρτήσεις μεταξύ των ετικετών χωρίς να θυσιάσουν τη γενική απόδοση.

Η βιβλιοθήκη scikit-multilearn στην Python αποτελεί ένα χαρακτηριστικό παράδειγμα εργαλείου που επιτρέπει τον συνδυασμό μοντέλων όπως το RF με σύγχρονες τεχνικές πολυετικετικής μάθησης, προσφέροντας ευελιξία στην επιλογή μετασχηματισμών και αξιολόγησης [23].

Παρά τα πλεονεκτήματά του, ο Random Forest παρουσιάζει και ορισμένους περιορισμούς:

- Υψηλή υπολογιστική πολυπλοκότητα όταν εφαρμόζεται σε προβλήματα με μεγάλο πλήθος ετικετών.
- Δυσκολία στην επεξήγηση των τελικών αποφάσεων, ιδίως όταν το μοντέλο αποτελείται από μεγάλο αριθμό δέντρων.
- Περιορισμένη αξιοποίηση συσχετίσεων μεταξύ ετικετών στο βασικό σχήμα BR.

Ωστόσο, η ευελιξία του και η δυνατότητα ενσωμάτωσης σε πιο σύνθετα πλαίσια, καθιστούν τον RF ένα από τα πιο αξιόπιστα εργαλεία στην κατηγοριοποίηση πολλαπλών ετικετών.

2.4 Τεχνικές Ενσωμάτωσης Ετικετών

Η παραδοσιακή προσέγγιση στην κατηγοριοποίηση πολλαπλών ετικετών συχνά βασίζεται σε μεθόδους που αγνοούν τη συσχέτιση μεταξύ των ετικετών. Ωστόσο, σε πλήθος εφαρμογών —όπως η ανάκτηση πληροφοριών, η βιοπληροφορική και η επεξεργασία φυσικής γλώσσας— οι ετικέτες εμφανίζουν σημαντικές συσχετίσεις και δομικές σχέσεις. Οι τεχνικές ενσωμάτωσης ετικετών (label embeddings) επιχειρούν να αξιοποιήσουν τις σχέσεις αυτές μέσω της αναπαράστασης των ετικετών σε ένα χαμηλότερης διάστασης, συνεχές διανυσματικό χώρο. Σκοπός είναι η αποτύπωση της συνθετότητας και των συσχετίσεων μεταξύ των ετικετών, ώστε η τελική πρόβλεψη να είναι πιο συνεπής και ακριβής [6].

Μία κοινή προσέγγιση στη δημιουργία των συσχετίσεων για τις ετικέτες είναι η αναπαράσταση του χώρου των πολλαπλών ετικετών ως γράφου. Σε αυτό το πλαίσιο, οι ετικέτες απεικονίζονται ως κόμβοι (nodes) και οι συσχετίσεις μεταξύ τους ως ακμές (edges). Οι ακμές μπορούν να βασίζονται σε συν-εμφανίσεις (co-occurrence) ετικετών στο εκπαιδευτικό σύνολο, ή σε πιο σύνθετες συναρτήσεις ομοιότητας. Ο στόχος είναι η αποτύπωση της τοπολογίας του γράφου σε έναν συνεχές διανυσματικό χώρο, όπου ετικέτες με παρόμοια συμφραζόμενα τοποθετούνται πλησιέστερα μεταξύ τους [24].

Ο Node2Vec αποτελεί έναν από τους πλέον διαδεδομένους αλγορίθμους για τη δημιουργία συσχετίσεων σε γράφους. Εισήχθη από τους Grover και Leskovec [9] και βασίζεται σε μια τροποποιημένη στρατηγική τυχαίων περιπάτων (biased random walks). Σε αντίθεση με προηγούμενες μεθόδους όπως το DeepWalk, ο Node2Vec επιτρέπει την ευέλικτη εξερεύνηση της τοπικής και της ευρύτερης δομής του γράφου, μέσω δύο υπερπαραμέτρων (p, q) που ελέγχουν το πώς κατευθύνονται οι περίπατοι.

Οι παραγόμενες ακολουθίες κόμβων θεωρούνται αναλογικά ως «προτάσεις» σε ένα γλωσσικό μοντέλο, και τροφοδοτούνται σε έναν αλγόριθμο τύπου Skip-Gram [25], ο οποίος εκπαιδεύεται ώστε να προβλέπει «γειτονικούς κόμβους» σε κάθε ακολουθία. Το αποτέλεσμα είναι μια αναπαράσταση ετικετών σε συνεχή διανυσματική μορφή, που διατηρεί τη σημασιολογική τους εγγύτητα όπως αυτή εκφράζεται στον γράφο.

Η χρήση των συσχετίσεων μέσω του Node2Vec σε προβλήματα πολυετικετικής μάθησης επιτρέπει τη μετατροπή του προβλήματος πρόβλεψης ετικετών σε πρόβλημα παλινδρόμησης σε έναν ενσωματωμένο χώρο. Αντί να προβλεφθεί απευθείας ένα σύνολο ετικετών, προβλέπεται ένα διανυσματικό σημείο στον χώρο συσχετίσεων, το οποίο στη συνέχεια συσχετίζεται με τις πλησιέστερες ετικέτες (μέσω k-NN ή cosine similarity) [26].

Η προσέγγιση αυτή έχει αποδειχθεί αποδοτική σε περιπτώσεις μεγάλου πλήθους ετικετών και υψηλής ετερογένειας, καθώς επιτρέπει την καλύτερη γενίκευση σε άγνωστους συνδυασμούς ετικετών. Επίσης, μέσω της γραφικής μοντελοποίησης, μπορεί να ενσωματώσει επιπλέον γνώση από ιεραρχικές ή σημασιολογικές δομές, βελτιώνοντας περαιτέρω την απόδοση [27].

Παρά τα σημαντικά πλεονεκτήματα, η χρήση συσχετίσεων με τον Node2Vec φέρει ορισμένες προκλήσεις:

- Υπολογιστικό κόστος: Η εκπαίδευση των συσχετίσεων σε μεγάλους γράφους είναι υπολογιστικά απαιτητική.
- Εξάρτηση από την ποιότητα του γράφου: Εσφαλμένες ή ελλιπείς συσχετίσεις μεταξύ ετικετών οδηγούν σε κακής ποιότητας συσχετίσεις.
- Μη ερμηνευσιμότητα: Οι παραγόμενες συσχετίσεις είναι δύσκολο να ερμηνευτούν ως προς την αρχική σημασιολογία των ετικετών.

Παρόλα αυτά, η ενσωμάτωση τεχνικών όπως ο Node2Vec ενισχύει τη δυνατότητα μοντελοποίησης της συνθετότητας και των εσωτερικών σχέσεων ενός προβλήματος πολλαπλών ετικετών, γεγονός που τις καθιστά ιδιαίτερα πολύτιμες σε απαιτητικά σύγχρονα συστήματα κατηγοριοποίησης.

2.5 Αποτίμηση της απόδοσης σε προβλήματα πολλαπλών ετικετών

Η αξιολόγηση της απόδοσης σε προβλήματα κατηγοριοποίησης πολλαπλών ετικετών αποτελεί μια πρόκληση που διαφέρει ουσιωδώς από την αξιολόγηση σε προβλήματα μοναδικής ετικέτας (single-label) σενάρια. Σε ένα πρόβλημα πολλαπλών ετικετών, κάθε δείγμα μπορεί να αντιστοιχεί σε πολλαπλές ταυτόχρονες ετικέτες, γεγονός που καθιστά ανεπαρκή τη χρήση κλασικών μετρικών, όπως η ακρίβεια ή η ακρίβεια ανά κλάση. Ως εκ τούτου, έχουν προταθεί εξειδικευμένες μετρικές αποτίμησης που λαμβάνουν υπόψη την πολυδιάστατη φύση των ετικετών [5].

Οι μετρικές αυτού του τύπου αξιολογούν την απόδοση του ταξινομητή για κάθε δείγμα ξεχωριστά και στη συνέχεια υπολογίζουν τον μέσο όρο επί του συνόλου του δείγματος.

- Hamming Loss: Υπολογίζει το ποσοστό των ετικετών που προβλέφθηκαν εσφαλμένα ανά δείγμα, δηλαδή τις ετικέτες που είτε λανθασμένα προβλέφθηκαν ως θετικές είτε παρέλειψαν να προβλεφθούν. Όσο χαμηλότερη είναι η τιμή αυτού του δείκτη, τόσο καλύτερη η απόδοση [6].
- Subset Accuracy (Exact Match Ratio): Πρόκειται για μία αυστηρή μετρική που θεωρεί σωστή μία πρόβλεψη μόνο όταν όλες οι προβλεφθείσες ετικέτες ταιριάζουν ακριβώς με τις πραγματικές. Παρότι χρήσιμη, τείνει να υποεκτιμά μοντέλα σε δεδομένα με υψηλό πλήθος ετικετών [11].
- Example-based Accuracy: Μετρά την ακρίβεια ανά δείγμα συγκρίνοντας το σύνολο των σωστά προβλεφθέντων ετικετών με το σύνολο όλων των ετικετών που είναι είτε πραγματικές είτε προβλεφθείσες. Ορίζεται ως το μέσο του λόγου της τομής προς την ένωση μεταξύ των προβλεπόμενων και πραγματικών ετικετών για κάθε δείγμα. Αποτελεί έναν πιο ισορροπημένο δείκτη απόδοσης, ευαίσθητο τόσο στις σωστές όσο και στις ελλιπείς προβλέψεις.
- Example-based Precision, Recall, F1-score: Υπολογίζονται ανά δείγμα, συγκρίνοντας το προβλεπόμενο σύνολο ετικετών με το πραγματικό, και στη συνέχεια λαμβάνεται ο μέσος όρος. Αυτές οι μετρικές είναι λιγότερο αυστηρές και ευαίσθητες στην αλληλεπικάλυψη των προβλέψεων με τις πραγματικές ετικέτες [28].

Επιπλέον, άλλες προσεγγίσεις βασίζονται σε μέσους όρους μετρικών δυαδικής ταξινόμησης ανά ετικέτα. Αντί να εστιάζουν σε μεμονωμένα δείγματα, οι παρακάτω μετρικές βασίζονται σε μετρικές δυαδικής ταξινόμησης για κάθε ετικέτα ξεχωριστά, και στη συνέχεια υπολογίζουν μέσους όρους.

- Macro-averaging: Υπολογίζει τις επιδόσεις κάθε ετικέτας ξεχωριστά και στη συνέχεια λαμβάνει τον απλό μέσο όρο. Χρήσιμο όταν όλες οι ετικέτες θεωρούνται εξίσου σημαντικές, ανεξαρτήτως συχνότητας [29].
- Micro-averaging: Υπολογίζει το συνολικό πλήθος αληθών/ψευδών θετικών και αρνητικών για όλες τις ετικέτες και έπειτα εφαρμόζει τις μετρικές (π.χ. precision, recall). Ευνοεί τις

συχνότερες ετικέτες και προσφέρει καλύτερη συνολική εικόνα σε σύνολα δεδομένων με ανομοιόμορφη κατανομή ετικετών [30].

Σε εφαρμογές όπου το σύστημα εξάγει βαθμολογίες καταλληλότητας ανά ετικέτα (π.χ. πιθανότητες ή scores), η αξιολόγηση μπορεί να βασιστεί σε τεχνικές κατάταξης, όπως:

- Ranking Loss: Εκφράζει το ποσοστό των περιπτώσεων στις οποίες μία άσχετη ετικέτα κατατάχθηκε υψηλότερα από μια σχετική, υποδηλώνοντας εσφαλμένη διάταξη [31].
- Average Precision: Υπολογίζει τη μέση ακρίβεια κατά την πλοήγηση σε λίστα ταξινομημένων προβλέψεων. Υψηλότερες τιμές υποδεικνύουν ότι οι σχετικές ετικέτες εντοπίζονται νωρίτερα στη λίστα [32].

Αυτού του είδους οι μετρικές είναι ιδιαιτέρως χρήσιμες σε εφαρμογές ανάκτησης πληροφοριών, σύστασης περιεχομένου και ιατρικής διάγνωσης, όπου η σειρά εμφάνισης των προτεινόμενων ετικετών φέρει σημασιολογική αξία.

Η επιλογή της κατάλληλης μετρικής εξαρτάται από τα χαρακτηριστικά του προβλήματος και τις απαιτήσεις της εφαρμογής. Για παράδειγμα, σε προβλήματα με πολλές σπάνιες ετικέτες, η χρήση micro-averaging μπορεί να οδηγήσει σε παραπλανητικά αποτελέσματα, ενώ σε εφαρμογές όπου απαιτείται πλήρης συμφωνία με το αναμενόμενο αποτέλεσμα, η subset accuracy έχει μεγαλύτερη σημασία. Συνεπώς, η χρήση πολλαπλών συμπληρωματικών μετρικών συνιστάται για την πλήρη αποτύπωση της απόδοσης ενός συστήματος πολλαπλών ετικετών [33].

Κεφάλαιο 3ο: Υλοποίηση Αλγορίθμων

3.1 Περιβάλλον Εργασίας και Εργαλεία Ανάπτυξης

Η ανάπτυξη του συστήματος για την παρούσα διπλωματική πραγματοποιήθηκε σε περιβάλλον εργασίας με τα εξής χαρακτηριστικά:

- ✓ **Σταθερός Υπολογιστής HP Elitedesk**
- ✓ **Επεξεργαστής: Intel(R) Core(TM) i7 6700 CPU @ 3.40GHz**
- ✓ **Μνήμη (RAM): 16,00GB**
- ✓ **OS: Microsoft Windows 10 Pro**
- ✓ **Οθόνη: 20''**

Η υλοποίηση των αλγορίθμων και η ανάπτυξη του συστήματος έγινε με τα εξής εργαλεία:

- ✓ **Python 3.10.4 (64-bit)**
- ✓ **Microsoft Visual Studio Code, Version 1.99.3**

Να σημειώσουμε εδώ ότι ο λόγος που προτιμήθηκε η χρήση της έκδοσης Python 3.10.4 ενώ υπάρχουν νεότερες εκδόσεις είναι ότι στις νεότερες εκδόσεις (version 3.13.4) υπήρξε πρόβλημα ασυμβατότητας με τη βιβλιοθήκη networkx, η χρήση της οποίας περιγράφεται στην επόμενη παράγραφο.

3.2 Τεχνολογικό Υπόβαθρο

Για την υλοποίηση της λειτουργικότητας και αξιολόγηση των αλγορίθμων πολυετικετικής ταξινόμησης, αξιοποιήθηκε ένα σύνολο βιβλιοθηκών της Python οι οποίες καλύπτουν ανάγκες προεπεξεργασίας, μοντελοποίησης, αποτίμησης και οπτικοποίησης. Παρακάτω παρουσιάζεται αναλυτικά ο ρόλος της κάθε βιβλιοθήκης:

- **Pandas**
  Παρέχει ευέλικτα εργαλεία για ανάγνωση, διαχείριση και επεξεργασία δεδομένων σε μορφή πινάκων (dataframes). Η βιβλιοθήκη παρέχει εργαλεία για εύκολη φόρτωση, καθαρισμό, φιλτράρισμα και ανάλυση δεδομένων, διευκολύνοντας τη διαδικασία προετοιμασίας των συνόλων δεδομένων. Χρησιμοποιείται κυρίως για τη μετατροπή δεδομένων τύπου ARFF σε κατάλληλη μορφή για την εκπαίδευση μοντέλων.

- **Numpy**
  Αποτελεί βασικό εργαλείο για αριθμητικούς υπολογισμούς, καθώς υποστηρίζει πίνακες πολλών διαστάσεων (arrays) και παρέχει γρήγορες μαθηματικές συναρτήσεις για αριθμητική επεξεργασία, χρήσιμες στην αναπαράσταση ετικετών και χαρακτηριστικών. Χρησιμοποιείται για μετασχηματισμούς δεδομένων και για τον χειρισμό των ετικετών και των χαρακτηριστικών.

- **Matplotlib.pyplot**
  Η υποβιβλιοθήκη pyplot της Matplotlib χρησιμοποιείται για τη γραφική απεικόνιση των αποτελεσμάτων. Επιτρέπει τη δημιουργία διαγραμμάτων, όπως bar plots και γραφήματα ακρίβειας, για την καλύτερη κατανόηση των επιδόσεων των αλγορίθμων.

- **Scipy**
  Αποτελεί μία ισχυρή, ανοιχτού κώδικα βιβλιοθήκη της Python που επεκτείνει τις δυνατότητες της NumPy, παρέχοντας προηγμένες μαθηματικές, επιστημονικές και τεχνικές συναρτήσεις.

Χρησιμοποιείται κυρίως για αριθμητική ολοκλήρωση και παραγώγους, γραμμική άλγεβρα, επεξεργασία σήματος και εικόνας, ανάλυση στατιστικών δεδομένων και εργασίες με αραιούς πίνακες. Χρησιμοποιήθηκαν οι υποβιβλιοθήκες:

- ✓ scipy.io: για την ανάγνωση αρχείων .arff, τα οποία περιλαμβάνουν πολυετικετικά σύνολα δεδομένων.
- ✓ scipy.sparse.csr_matrix: για τη μετατροπή των δεδομένων σε μορφή αραιών πινάκων, βελτιώνοντας την αποδοτικότητα σε υπολογισμούς.

- **Networkx**
  Βιβλιοθήκη για την κατασκευή και επεξεργασία γράφων. Χρησιμοποιήθηκε για τη δημιουργία γράφου συνεμφάνισης ετικετών, όπου οι κόμβοι είναι ετικέτες και οι ακμές υποδεικνύουν συχνές συνεμφανίσεις.

- **Skmultilearn.adapt**
  Εξειδικευμένη βιβλιοθήκη για πολυετικετική μάθηση

  - ✓ skmultilearn.adapt.MLkNN: Προσαρμογή του αλγορίθμου kNN ώστε να υποστηρίζει την πρόβλεψη πολλαπλών ετικετών με βάση τη συχνότητα των ετικετών στους γείτονες.
  - ✓ skmultilearn.adapt.BRkNNaClassifier: Επέκταση της στρατηγικής Binary Relevance με χρήση του kNN για κάθε ετικέτα ανεξάρτητα, που χρησιμοποιήθηκε ως βασικός αλγόριθμος στα πειράματα.
  - ✓ skmultilearn.adapt.MLARAM (Multi-Label Associative Rule-based Classification): Συνδυάζει τεχνικές βασισμένες στη μνήμη και στους κανόνες συσχέτισης (associative memory) για την πρόβλεψη πολλαπλών ετικετών με βάση την ενεργοποίηση προτύπων που έχουν αποθηκευτεί στο μοντέλο.
  - ✓ skmultilearn.adapt.MLTSVM (Multi-Label Twin SVM): Επέκταση του δυαδικού αλγορίθμου Twin SVM για πολυετικετικά προβλήματα, δημιουργώντας δύο μη παράλληλα υπερεπίπεδα ανά ετικέτα και εκπαιδεύοντας πολλαπλά μοντέλα με στόχο τη βελτιωμένη διαχωρισιμότητα.
  - ✓ skmultilearn.problem_transform.LabelPowerset: Τεχνική μετασχηματισμού προβλήματος που αντιμετωπίζει κάθε μοναδικό συνδυασμό ετικετών ως ξεχωριστή κατηγορία. Παρότι καταγράφει τις αλληλεπιδράσεις μεταξύ των ετικετών, δυσκολεύεται να επεκταθεί σε datasets με μεγάλο αριθμό μοναδικών συνδυασμών.

- **Scikit-learn (Sklearn)**
  Η πιο ευρέως χρησιμοποιούμενη βιβλιοθήκη για μηχανική μάθηση. Από αυτήν χρησιμοποιήθηκαν διάφορα υποπακέτα:

  - ✓ sklearn.ensemble.RandomForestClassifier: Χρησιμοποιείται για την εκπαίδευση μοντέλων Random Forest σε προβλήματα ταξινόμησης.
  - ✓ sklearn.neighbors.NearestNeighbors: Παρέχει τη βασική λειτουργικότητα για την υλοποίηση αλγορίθμων k-Nearest Neighbors, και χρησιμοποιήθηκε ως βάση σε BRkNN και MLkNN.
  - ✓ sklearn.metrics: Περιλαμβάνει μετρικές αποτίμησης όπως hamming_loss και accuracy_score, που είναι κατάλληλες για multilabel προβλήματα.
  - ✓ sklearn.cluster.KMeans: Χρησιμοποιήθηκε για την ομαδοποίηση των label embeddings και τη δημιουργία συγχωνευμένων ομάδων ετικετών.
  - ✓ sklearn.multioutput.ClassifierChain: Εφαρμογή της τεχνικής Classifier Chains, στην οποία κάθε ταξινομητής λαμβάνει υπόψη τις προβλέψεις των προηγούμενων, ενσωματώνοντας έτσι εξαρτήσεις μεταξύ ετικετών.

- ✓ sklearn.multioutput.MultiOutputClassifier: Γενικεύει οποιονδήποτε ταξινομητή για χρήση σε προβλήματα με πολλαπλές εξόδους, μετατρέποντας το multilabel πρόβλημα σε ισάριθμα ανεξάρτητα μοντέλα—παρόμοιο με τη στρατηγική Binary Relevance.

- **Node2Vec**
  Εφαρμόστηκε για την παραγωγή διανυσματικών αναπαραστάσεων (embeddings) των ετικετών από τον γράφο του NetworkX. Η τεχνική αυτή επιτρέπει την ενσωμάτωση των λανθανουσών σχέσεων μεταξύ των ετικετών στο τελικό μοντέλο μάθησης.

- **Time**
  Ενσωματωμένη βιβλιοθήκη της Python που χρησιμοποιήθηκε για τη μέτρηση του χρόνου εκτέλεσης των αλγορίθμων και των επιμέρους σταδίων, παρέχοντας συγκρίσιμες μετρήσεις υπολογιστικής αποδοτικότητας.

3.3 Σύνολα Δεδομένων (Datasets)

Για την πειραματική αξιολόγηση των αλγορίθμων κατηγοριοποίησης πολλαπλών ετικετών, επιλέχθηκαν οκτώ ευρέως χρησιμοποιούμενα και ετερογενή σύνολα δεδομένων (datasets), τα οποία είναι διαθέσιμα μέσω της πλατφόρμας MLL Resources του Πανεπιστημίου της Κόρδοβα [34].

Πιο συγκεκριμένα, τα σύνολα δεδομένων που επιλέχθηκαν είναι:

CAL500 [35]: Πρόκειται για ένα μουσικό σύνολο δεδομένων το οποίο αποτελείται από 502 τραγούδια. Κάθε τραγούδι έχει επισημανθεί από τουλάχιστον τρεις ανθρώπους (anotators), οι οποίοι χρησιμοποίησαν ένα λεξιλόγιο 174 ετικετών σχετικών με σημασιολογικές έννοιες (semantic concepts). Οι ετικέτες αυτές καλύπτουν έξι σημασιολογικές κατηγορίες: οργανολογία (instrumentation), χαρακτηριστικά φωνής, μουσικά είδη, συναισθήματα, ακουστική ποιότητα του τραγουδιού και όροι χρήσης.

Birds [36]: Πρόκειται για ένα σύνολο δεδομένων το οποίο χρησιμοποιείται για την πρόβλεψη του συνόλου των ειδών πτηνών τα οποία είναι παρόντα, βάσει ενός ηχητικού αποσπάσματος διάρκειας δέκα δευτερολέπτων.

Water Quality [37]: Το συγκεκριμένο σύνολο δεδομένων χρησιμοποιείται για την πρόβλεψη της ποιότητας του νερού των ποταμών της Σλοβενίας, βάσει 16 χαρακτηριστικών, μεταξύ των οποίων είναι η θερμοκρασία, το pH, η σκληρότητα του νερού, τα επίπεδα NO_2 ή CO_2.

Yeast [38]: Το συγκεκριμένο σύνολο δεδομένων περιλαμβάνει εκφράσεις μικροσυστοιχιών (micro-array expressions) και φυλογενετικά προφίλ για 2417 γονίδια της ζύμης. Κάθε γονίδιο συνοδεύεται από ένα υποσύνολο 14 λειτουργικών κατηγοριών (π.χ. μεταβολισμός, ενέργεια κ.ά.) οι οποίες ανήκουν στο ανώτερο επίπεδο του καταλόγου λειτουργιών (functional catalogue).

Mediamill [39]: Πρόκειται για ένα σύνολο δεδομένων multimedia το οποίο χρησιμοποιείται για γενική δεικτοδότηση βίντεο (generic video indexing) και προέρχεται από το benchmark TRECVID 2005/2006. Το σύνολο περιλαμβάνει 85 ώρες διεθνών ειδησεογραφικών μεταδόσεων, κατηγοριοποιημένες σε 100 ετικέτες, ενώ κάθε δείγμα βίντεο αναπαρίσταται ως διανυσματικό χαρακτηριστικό 120 διαστάσεων, αποτελούμενο από αριθμητικά γνωρίσματα.

Scene [40]: Πρόκειται για ένα σύνολο δεδομένων το οποίο περιλαμβάνει 2407 εικόνες, επισημασμένες με κατηγορίες πλήθους έως και 6 στο σύνολο. Αυτές είναι: παραλία, ηλιοβασίλεμα, φθινοπωρινό

φύλλωμα, αγρός, βουνό και αστικό τοπίο. Κάθε εικόνα περιγράφεται μέσω 294 αριθμητικών οπτικών χαρακτηριστικών, τα οποία αντιστοιχούν σε χωρικές χρωματικές ροπές στον χρωματικό χώρο LUV.

Emotions [41]: Το συγκεκριμένο σύνολο δεδομένων είναι επίσης γνωστό ως *Music*. Πρόκειται για ένα μικρό σύνολο δεδομένων που χρησιμοποιείται για την ταξινόμηση μουσικών κομματιών με βάση τα συναισθήματα που προκαλούν, σύμφωνα με το μοντέλο διάθεσης Tellegen-Watson-Clark: έκπληξη-θαυμασμός, χαρά-ευχαρίστηση, χαλάρωση-ηρεμία, ηρεμία-ακινησία, λύπη-μοναξιά και θυμός-επιθετικότητα. Το σύνολο αποτελείται από 593 τραγούδια και περιλαμβάνει 6 κατηγορίες.

Flags [42]: Το σύνολο αυτό περιλαμβάνει πληροφορίες για ορισμένες χώρες και τις σημαίες τους, με στόχο την πρόβλεψη ορισμένων χαρακτηριστικών. Χρησιμοποιήθηκε για πρώτη φορά σε πρόβλημα ταξινόμησης πολλαπλών ετικετών (Multi-label classification) στη μελέτη των Gonçalves et al.[42], ενώ το αρχικό σύνολο δεδομένων είναι διαθέσιμο στο αποθετήριο του UCI.

Τα εν λόγω σύνολα δεδομένων επιλέχθηκαν με τρόπο ώστε να καλύπτουν ποικιλία ως προς τον αριθμό των παρατηρήσεων, τα αριθμητικά χαρακτηριστικά, τον αριθμό ετικετών και τη δομή της ετικετοποίησης.

Ο Πίνακας 3.1 συνοψίζει τα βασικά χαρακτηριστικά των συνόλων δεδομένων που χρησιμοποιήθηκαν:

Πίνακας 2: Χαρακτηριστικά των συνόλων δεδομένων

Σύνολο Δεδομένων	**Συντ/ση**	**Τομέας**	**Παρατηρήσεις**	**Αριθμ. Χαρακτ/κά**	**Ετικέτες**	**Μειωμένες ετικέτες**
CAL500	CAL	Music	502	68	174	*30%
BIRDS	BRD	Audio	645	258	19	10
WATER QUALITY	WQT	Chemistry	1.060	16	14	10
YEAST	YST	Biology	2.417	103	14	10
MEDIAMILL	MDM	Video	43.907	120	101	*30%
SCENE	SCN	Image	2.407	294	6	*50%
EMOTIONS	EMT	Music	593	72	6	*50%
FLAGS	FLG	Image	194	10	7	*50%

Τα παραπάνω σύνολα δεδομένων καλύπτουν εφαρμογές από διαφορετικούς τομείς, όπως η μουσική (CAL500, EMOTIONS), τα πολυμέσα (MEDIAMILL), η βιολογία (YEAST), το περιβάλλον (WATER QUALITY), τα γεωγραφικά δεδομένα (FLAGS), και η αναγνώριση σκηνών (SCENE). Στη στήλη Παρατηρήσεις σημειώνεται το πλήθος των εγγραφών του κάθε συνόλου δεδομένων. Κάθε παρατήρηση (εγγραφή) απαρτίζεται από ένα σύνολο αριθμητικών χαρακτηριστικών κι ένα σύνολο ετικετών.

Το πλήθος των παρατηρήσεων στο σύνολο των δεδομένων κυμαίνεται από 194 (Flags) έως 43.907 (Mediamill), ενώ το πλήθος των ετικετών κυμαίνεται από 6 (Emotions, Scene) έως 174 (CAL500). Το πλήθος των αριθμητικών χαρακτηριστικών παρουσιάζει επίσης σημαντικές διακυμάνσεις, με πλήθος που κυμαίνεται από 10 (Flags) έως 294 (Scene) αριθμητικά γνωρίσματα.

Η τελευταία στήλη του πίνακα αναφέρεται στο ποσοστό των επιστρεφόμενων ετικετών μετά τη μείωση κατά την εφαρμογή των ενσωματώσεων μέσω του Node2Vec, είτε ως απόλυτη τιμή (π.χ., 10 ετικέτες),

είτε ως ποσοστό επί του αρχικού πλήθους ετικετών (π.χ., *30% ή *50%). Στις περιπτώσεις όπου έχουμε λίγες ετικέτες (6 ή 7) στο αρχικό δείγμα, η μείωση των ετικετών εφαρμόζεται στο 50%. Για πλήθος ετικετών έως 20 εφαρμόζουμε μείωση τέτοια ώστε να έχουμε 10 ετικέτες κατά απόλυτη τιμή. Τέλος, στην περίπτωση όπου το πλήθος των ετικετών υπερβαίνει τις 20 έχουμε μείωση ετικετών κατά 70%. Η διαδικασία αυτή στοχεύει στη συμπύκνωση πληροφορίας και στη μείωση της υπολογιστικής πολυπλοκότητας των ταξινομητών.

Η ποικιλομορφία των επιλεγμένων συνόλων δεδομένων προσφέρει ισχυρό πειραματικό πλαίσιο για την εξαγωγή γενικευμένων συμπερασμάτων όσον αφορά την αποτελεσματικότητα των συγκρινόμενων αλγορίθμων, τόσο πριν όσο και μετά την ενσωμάτωση χαρακτηριστικών.

3.4 Υλοποίηση Φόρτωσης Δεδομένων

Για κάθε ένα από τα οκτώ σύνολα δεδομένων, πραγματοποιήθηκε διαδικασία φόρτωσης των δεδομένων από αρχεία τύπου .arff με χρήση της βιβλιοθήκης scipy.io.arff.

3.4.1 Αρχεία Τύπου ARFF

Τα αρχεία τύπου ARFF (Attribute-Relation File Format) είναι ένα ειδικός τύπος αρχείων ο οποίος χρησιμοποιείται ευρέως στη μηχανική μάθηση. Ιδιαίτερα, σε συνδυασμό με το εργαλείο WEKA, ένα δημοφιλές λογισμικό ανοιχτού κώδικα για εξόρυξη δεδομένων (data mining) και ανάλυση δεδομένων. Ο τύπος αυτός είναι απλός και βασίζεται στην κωδικοποίηση ASCII (plain text), συνδυάζοντας οριστικά δεδομένα και μεταδεδομένα σε ένα ενιαίο αρχείο.

Ένα ARFF αρχείο αποτελείται από δύο βασικά μέρη:

1. **Header (Κεφαλίδα): Περιγραφή των μεταδεδομένων (metadata)**
   Η κεφαλίδα περιγράφει τα χαρακτηριστικά (attributes) των δεδομένων και περιλαμβάνει:
   - Το όνομα του dataset (@RELATION)
   - Τη λίστα των χαρακτηριστικών (@ATTRIBUTE)
   - Το είδος των δεδομένων για κάθε χαρακτηριστικό (π.χ. NUMERIC, STRING, NOMINAL)
2. **Data (Δεδομένα):**
   Η ενότητα των δεδομένων ξεκινά με τη λέξη-κλειδί @DATA, την οποία ακολουθεί μια λίστα εγγραφών. Κάθε εγγραφή (γραμμή) αποτελεί ένα δείγμα (instance), το οποίο περιλαμβάνει τις τιμές των χαρακτηριστικών χωρισμένες με κόμμα.

Στην εικόνα που ακολουθεί παρουσιάζεται η δομή ενός τέτοιου αρχείου. Το αρχείο αποτελείται από πέντε (5) χαρακτηριστικά (attributes). Οι τιμές τους μπορεί να είναι αριθμητικές (NUMERIC), ΝΑΙ/ΟΧΙ ή λογικές (TRUE/FALSE). Τα δεδομένα καταγράφονται σε γραμμές.

sample.arff

```
@RELATION weather

@ATTRIBUTE outlook {sunny, overcast, rainy}
@ATTRIBUTE temperature NUMERIC
@ATTRIBUTE humidity NUMERIC
@ATTRIBUTE windy {TRUE, FALSE}
@ATTRIBUTE play {yes, no}

@DATA
sunny, 85, 85, FALSE, no
sunny, 80, 90, TRUE, no
overcast, 83, 78, FALSE, yes
rainy, 70, 96, FALSE, yes
```

3.4.2 Φόρτωση αρχείων ARFF στην Python

Προκειμένου να φορτωθούν τα στοιχεία που περιέχονται σε ένα αρχείο με κατάληξη .arff, στη γλώσσα προγραμματισμού Python, χρησιμοποιούμε τη βιβλιοθήκη scipy.io. Στο παράδειγμα που ακολουθεί αρχικά, εισάγουμε τις βιβλιοθήκες scipy.io και pandas (import). Ύστερα, φορτώνουμε ένα αρχείο arff με όνομα '*sample.arff*'. Στην μεταβλητή data ανατίθενται τα δεδομένα ως πίνακας (numpy array) και στη μεταβλητή meta ανατίθενται πληροφορίες για τα χαρακτηριστικά (τύποι, ονόματα, κ.λπ.).

sample.py > ...

```
from scipy.io import arff
import pandas as pd

data, meta = arff.loadarff('sample.arff')
df = pd.DataFrame(data)
```

Τα αρχεία τύπου ARFF παρουσιάζουν αρκετά πλεονεκτήματα. Είναι εύκολα στην δημιουργία και την κατανόησή τους καθώς στην ουσία είναι απλό κείμενο. Περιέχουν μεταπληροφορίες, όπως ονόματα των χαρακτηριστικών, στο ίδιο αρχείο με τα δεδομένα και υποστηρίζονται από πολλά εργαλεία μηχανικής μάθησης.

Ωστόσο, υπάρχουν και περιορισμοί στη χρήση τους, όπως το γεγονός ότι δεν είναι ιδιαίτερα αποδοτικά σε περίπτωση μεγάλων συνόλων δεδομένων, ότι δεν υποστηρίζουν εμφωλευμένες δομές ή πολύπλοκους τύπους δεδομένων και οι κατηγορικές τιμές, όπως yes/no, θα πρέπει να είναι προκαθορισμένες (enumerated).

3.4.3 Φόρτωση των οκτώ (8) συνόλων δεδομένων

Η διαδικασία φόρτωσης και προεπεξεργασίας των δεδομένων αποτελεί το πρώτο κρίσιμο στάδιο της υλοποίησης του αλγορίθμου. Τα δεδομένα τα οποία χρησιμοποιήθηκαν φορτώθηκαν από αρχεία τύπου `.arff` με χρήση της βιβλιοθήκης `scipy.io.arff`, και μετατράπηκαν σε αντικείμενο τύπου

`DataFrame` μέσω της βιβλιοθήκης `pandas`. Όπως φαίνεται και στο δείγμα κώδικα που ακολουθεί, τα πρώτα χαρακτηριστικά αναπαριστούν το διάνυσμα εισόδου (X), ενώ οι επόμενες στήλες αντιστοιχούν σε ετικέτες πολλαπλής κατηγοριοποίησης (multi-label). Οι ετικέτες μετατράπηκαν σε δυαδική μορφή, ώστε να μπορούν να αξιοποιηθούν από τους ταξινομητές κατηγορίας πολλαπλών ετικετών.

Για παράδειγμα, η διαδικασία φόρτωσης των δεδομένων για το σύνολο δεδομένων Mediamill έχει ως εξής:

```
# ===== Φόρτωση δεδομένων ARFF =====
data, meta = arff.loadarff('mediamill.arff')
df = pd.DataFrame(data)

X = df.iloc[:, :120].values.astype(float)
y = df.iloc[:, -101:].apply(lambda x: x == b'1').astype(int).values  # Binary μετατροπή
numOfLabels = 101 #μεταβλητή που κρατάει το πλήθος των labels
```

Στη μεταβλητή X ανατίθενται οι πρώτες 120 αριθμητικές τιμές και οι τελευταίες 101 τιμές αφορούν τις ετικέτες του συνόλου δεδομένων.

Κατά τον ίδιο τρόπο πραγματοποιήθηκε η φόρτωση των δεδομένων και για τα υπόλοιπα επτά (7) σύνολα.

3.5 Ανάπτυξη Κώδικα για BRkNN

3.5.1 Θεωρητικό Υπόβαθρο

Η μέθοδος ταξινόμησης BRkNN (Binary Relevance k-Nearest Neighbors) είναι μια επέκταση του αλγορίθμου k-NN για ταξινόμηση σε προβλήματα πολλαπλών ετικετών. Στηρίζεται στην προσέγγιση Binary Relevance, η οποία μετατρέπει ένα πρόβλημα σε πολλαπλά δυαδικά προβλήματα ταξινόμησης, ένα για κάθε ετικέτα. Η μέθοδος BRkNN βασίζεται στο γεγονός ότι κάθε δείγμα μπορεί να έχει περισσότερες από μία ετικέτες και χρησιμοποιεί k-πλησιέστερους γείτονες για να υλοποιήσει μία πρόβλεψη για κάθε ετικέτα ξεχωριστά.

Ωστόσο, η ανεξάρτητη αντιμετώπιση των ετικετών ενδεχομένως να αγνοεί σημαντικές συσχετίσεις μεταξύ τους. Για την αντιμετώπιση αυτού του προβλήματος, χρησιμοποιούνται τεχνικές όπως:

- Κατασκευή γράφου συνεμφάνισης ετικετών, κατά την οποία δημιουργείται ένας γράφος όπου οι ετικέτες αποτελούν τους κόμβους και οι ακμές απεικονίζουν τη σύνδεση των ετικετών που εμφανίζονται μαζί.
- Ενσωμάτωση ετικετών με Node2Vec, έτσι ώστε οι συσχετίσεις να κωδικοποιούνται ως αριθμητικά διανύσματα.
- Clustering στις ενσωματωμένες ετικέτες προκειμένου να γίνεται ομαδοποίηση παρόμοιων ετικετών και μείωση της διαστασιμότητας.
- Εμπλουτισμός χαρακτηριστικών (feature enrichment), κατά την οποία οι αρχικές τιμές χαρακτηριστικών των δειγμάτων ενισχύονται με τις ενσωματωμένες πληροφορίες των ετικετών.

3.5.2 Ανάπτυξη Λειτουργικότητας σε Python

Η ανάπτυξη της λειτουργικότητας με τη βοήθεια της μεθόδου BRkNN υλοποιήθηκε εξ ολοκλήρου σε γλώσσα Python, με χρήση δημοφιλών βιβλιοθηκών όπως:

- pandas, numpy για την επεξεργασία των δεδομένων
- networkx για τη δημιουργία των γράφων
- node2vec για την ενσωμάτωση ετικετών
- sklearn και scikit-multilearn για την ταξινόμηση και την αξιολόγηση

Κατά την ανάπτυξη του κώδικα, χρειάστηκε να αντιμετωπίσουμε διάφορα προβλήματα ασυμβατότητας με βιβλιοθήκες, τα οποία αναλύονται στην ενότητα 3.5.2.1. Η ανάπτυξη της απαιτούμενης λειτουργικότητας χωρίζονται σε επιμέρους ενότητες, οι οποίες περιγράφονται αναλυτικά στις παραγράφου 3.5.2.2 έως 3.5.2.10:

3.5.2.1 Αντιμετώπιση Συμβατότητας Βιβλιοθηκών με Monkey Patching

Κατά την ενσωμάτωση του ταξινομητή BRkNNaClassifier της βιβλιοθήκης `scikit-multilearn`, παρουσιάστηκαν προβλήματα συμβατότητας τα οποία σχετίζονται με τη χρήση συγκεκριμένων εκδόσεων της βιβλιοθήκης `scikit-learn`. Πιο συγκεκριμένα, ο ταξινομητής BRkNNaClassifier βασίζεται στην κλάση MLkNN, η οποία με τη σειρά της χρησιμοποιεί λειτουργίες της scikit-learn, όπως η μέθοδος get_params(). Αυτή η μέθοδος εξαρτάται εσωτερικά από τη λειτουργία _get_param_names(), η οποία στη χρησιμοποιούμενη έκδοση της scikit-learn είτε είχε αφαιρεθεί είτε είχε τροποποιηθεί με τέτοιο τρόπο ώστε να μην είναι πλέον συμβατή.
Επιπλέον, παρουσιάστηκε πρόβλημα και με τη χρήση της κλάσης NearestNeighbors της scikit-learn. Συγκεκριμένα, όταν γινόταν προσπάθεια να δοθεί ως είσοδος μόνο μία ακέραιη τιμή (π.χ. ο αριθμός γειτόνων), η μέθοδος δεν τη διαχειριζόταν όπως αναμενόταν, πιθανόν λόγω αλλαγών στη διαχείριση παραμέτρων στη νέα έκδοση.
Για την επίλυση των παραπάνω ασυμβατοτήτων εφαρμόστηκε η τεχνική του Monkey Patching. Αυτή η τεχνική περιλαμβάνει την άμεση, δυναμική τροποποίηση των μεθόδων ή των κλάσεων των βιβλιοθηκών κατά τη διάρκεια εκτέλεσης του κώδικα. Στην προκειμένη περίπτωση:

- Ορίστηκε χειροκίνητα η `_get_param_names()` στην κλάση `BaseEstimator` ώστε να επιστρέφει κατάλληλα τις παραμέτρους αρχικοποίησης της κλάσης, εξαιρώντας το `self`.
- Τροποποιήθηκε η μέθοδος `__init__()` της κλάσης `NearestNeighbors` ώστε να μπορεί να δέχεται απευθείας έναν ακέραιο αριθμό ως τιμή για την παράμετρο `k` (παράμετρος που ορίζει τον αριθμών των πλησιέστερων γειτόνων), επιτρέποντας έτσι τη συμβατή χρήση του ταξινομητή BRkNNaClassifier χωρίς την πρόκληση σφαλμάτων.

```
# ===== Monkey Patch για MLkNN, BRkNN =====
from sklearn.base import BaseEstimator
def _get_param_names(cls):
    return [key for key in cls.__init__.__code__.co_varnames if key != 'self']
BaseEstimator._get_param_names = classmethod(_get_param_names)

# Monkey Patch για NearestNeighbors
_original_init = NearestNeighbors.__init__
def patched_init(self, *args, **kwargs):
    if len(args) == 1 and not kwargs.get('n_neighbors'):
        kwargs['n_neighbors'] = args[0]
        args = ()
    _original_init(self, *args, **kwargs)
NearestNeighbors.__init__ = patched_init
```

Η εφαρμογή της παραπάνω μεθόδου επέτρεψε τη συνέχιση της υλοποίησης χωρίς την ανάγκη παρέμβασης στο υπάρχον περιβάλλον των βιβλιοθηκών, αποφεύγοντας έτσι ενδεχόμενα προβλήματα που θα μπορούσαν να προκύψουν από αναβαθμίσεις ή τροποποιήσεις των σχετικών εξαρτήσεων.

3.5.2.2 Φόρτωση και Προεπεξεργασία Δεδομένων

Για την παρουσίαση της υλοποίησης της μεθόδου χρησιμοποιείται το σύνολο δεδομένων CAL 500, ωστόσο η ίδια διαδικασία ακολουθήθηκε για όλα τα σύνολα (8).
Το σύνολο δεδομένων CAL500 αρχικά διατίθεται σε μορφή αρχείου .arff, η οποία είναι ευρέως χρησιμοποιούμενη για την αναπαράσταση συνόλων δεδομένων στον χώρο της μηχανικής μάθησης. Το αρχείο αυτό μετατρέπεται σε αντικείμενο τύπου DataFrame, χρησιμοποιώντας κατάλληλες βιβλιοθήκες επεξεργασίας δεδομένων, όπως η pandas.
Ένα DataFrame αποτελεί βασική δομή δεδομένων στη βιβλιοθήκη pandas και μπορεί να θεωρηθεί ως ένας πίνακας δύο διαστάσεων (παρόμοιος με ένα υπολογιστικό φύλλο ή έναν πίνακα σε μια σχεσιακή βάση δεδομένων), στον οποίο οι γραμμές αντιστοιχούν σε δείγματα (παρατηρήσεις) και οι στήλες σε χαρακτηριστικά ή ετικέτες. Κάθε στήλη μπορεί να περιέχει δεδομένα διαφορετικού τύπου (αριθμητικά, κατηγορικά κ.ά.), και η δομή αυτή επιτρέπει την αποδοτική ανάλυση και επεξεργασία των δεδομένων.
Στο συγκεκριμένο σύνολο δεδομένων, το DataFrame διαχωρίζεται σε δύο κύρια μέρη:

- 68 χαρακτηριστικά (features), τα οποία αντιπροσωπεύουν τις εισόδους του συστήματος ταξινόμησης (όπως π.χ. ακουστικά χαρακτηριστικά τραγουδιών),
- 174 δυαδικές ετικέτες (target labels), οι οποίες αντιστοιχούν σε κατηγορίες ή περιγραφές των τραγουδιών. Οι ετικέτες αυτές κωδικοποιούνται με δυαδικό τρόπο (0/1), ώστε να είναι κατάλληλες για χρήση σε αλγορίθμους ταξινόμησης πολλαπλών ετικετών (multi-label classification).

Ακολούθως, το σύνολο δεδομένων διαχωρίζεται με τυχαίο τρόπο σε υποσύνολο εκπαίδευσης (training set) και υποσύνολο δοκιμής (test set), με αναλογία 70%–30% αντίστοιχα. Ο διαχωρισμός αυτός επιτρέπει την εκπαίδευση των αλγορίθμων στο πρώτο υποσύνολο και την αξιολόγηση της απόδοσής τους στο δεύτερο, εξασφαλίζοντας έτσι αντικειμενική εκτίμηση των αποτελεσμάτων.

```
# ===== Φόρτωση δεδομένων ARFF =====
data, meta = arff.loadarff('CAL500.arff')
df = pd.DataFrame(data)

X = df.iloc[:, :68].values.astype(float)
y = df.iloc[:, -174:].apply(lambda x: x == b'1').astype(int).values  #Binary μετατροπή
numOfLabels = 174 #metavliti pou krataei to plithos twn labels

# ===== Διαχωρισμός σε Training και Test Set =====
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

3.5.2.3 Αρχική Εκπαίδευση με BRkNN

Εκπαιδεύεται ο ταξινομητής BRkNNaClassifier με όλα τα χαρακτηριστικά και όλες τις ετικέτες, χωρίς καμία προεπεξεργασία ή μείωση. Για την τιμή του k έγιναν δοκιμές για την τιμή k=1 και k= nrOfLabels. Παρατηρείται ότι για τιμή k ίση με τον αριθμό των ετικετών (π.χ. k = 174 για σύνολο CAL500), προκύπτει η χαμηλότερη τιμή για τον δείκτη Hamming Loss.
Ο ταξινομητής BRkNNaClassifier εκπαιδεύεται χρησιμοποιώντας το πλήρες σύνολο των διαθέσιμων χαρακτηριστικών καθώς και το σύνολο όλων των ετικετών, χωρίς να εφαρμόζεται καμία επιπλέον προεπεξεργασία ή μείωση στα δεδομένα εισόδου. Για τον καθορισμό της βέλτιστης τιμής της παραμέτρου k, η οποία αντιστοιχεί στον αριθμό των γειτονικών δειγμάτων που λαμβάνονται υπόψη κατά την ταξινόμηση, πραγματοποιήθηκαν πειραματικές δοκιμές με δύο βασικές τιμές: k=1 και k ίση με τον συνολικό αριθμό των ετικετών (δηλαδή k = nrOfLabels) για κάθε σύνολο δεδομένων. Στην περίπτωση για το σύνολο δεδομένων CAL500, όπως φαίνεται και στο δείγμα του κώδικα που ακολουθεί, ο αριθμός των ετικετών είναι 174. Συνεπώς, εξετάστηκε η συμπεριφορά του ταξινομητή τόσο για k=1 όσο και για k=174.

Από τα πειραματικά αποτελέσματα προκύπτει ότι η επιλογή της παραμέτρου k ίσης με τον συνολικό αριθμό των ετικετών οδηγεί σε χαμηλότερη τιμή του δείκτη Hamming Loss.
Ο δείκτης Hamming Loss αποτελεί μέτρο της απόδοσης σε προβλήματα ταξινόμησης πολλαπλών ετικετών και υπολογίζει το μέσο ποσοστό λανθασμένων ετικετών ανά δείγμα. Συνεπώς, η μείωση της τιμής αυτού του δείκτη υποδηλώνει βελτιωμένη ακρίβεια στην πρόβλεψη των σωστών ετικετών από τον ταξινομητή.

```
#Arxikos Yplogismos prin ti sygxwneusi
##classifierBrKnn = BRkNNaClassifier(k=1)
classifierBrKnn = BRkNNaClassifier(k=174) #gia k=174 dinei to xamilotero hamming loss
classifierBrKnn.fit(X_train, y_train)  # Ekpaidefsi tou taksinomiti

#PREDICTION on the test set
y_pred = classifierBrKnn.predict(X_test) # Provlepw ta labels gia to X_test
```

Οι τιμές των δεικτών Hamming Loss και Accuracy, μαζί με το συνολικό χρόνο επεξεργασίας καταγράφονται για σκοπούς σύγκρισης με τη βελτιωμένη εκδοχή που ακολουθεί.

```
#EVALUATE performance
# Μέτρηση Χρόνου πριν τα embeddings
print("Hamming Loss BRkNN Before Embeddings:", hamming_loss(y_test, y_pred.astype(int)))
end_time = time.time()
print("Accuracy BRkNN Before Embeddings:", accuracy_score(y_test, y_pred.astype(int)))
print("Process Time Before Embeddings:", end_time - start_time, "seconds")
```

3.5.2.4 Δημιουργία Γράφου Συνεμφάνισης Ετικετών

Με βάση το υποσύνολο εκπαίδευσης (training set), δημιουργείται ένας γράφος ο οποίος απεικονίζει τη συνεμφάνιση των ετικετών. Στον γράφο αυτό, κάθε κόμβος αντιστοιχεί σε μία ετικέτα του συνόλου δεδομένων. Μία ακμή εισάγεται μεταξύ δύο κόμβων όταν οι αντίστοιχες ετικέτες εμφανίζονται ταυτόχρονα στο ίδιο δείγμα. Επιπλέον, το βάρος κάθε ακμής καθορίζεται και αυξάνεται αναλογικά με βάση τη συχνότητα εμφάνισης των δύο ετικετών σε κοινά δείγματα, παρέχοντας έτσι μία μέτρηση της σχέσης και της συσχέτισης μεταξύ τους.

```
# ===== Δημιουργία Γράφου Συνεμφάνισης Ετικετών =====
start_time = time.time()
def build_label_cooccurrence_graph(y):
    num_labels = y.shape[1] #plithos twn labels
    G = nx.Graph()

    for label in range(num_labels):
        G.add_node(label)

    for sample in y:
        labels = np.where(sample == 1)[0]
        for i in range(len(labels)):
            for j in range(i + 1, len(labels)):
                if G.has_edge(labels[i], labels[j]):
                    G[labels[i]][labels[j]]['weight'] += 1
                else:
                    G.add_edge(labels[i], labels[j], weight=1)
    return G

G = build_label_cooccurrence_graph(y_train)
```

3.5.2.5 Ενσωμάτωση Ετικετών με Node2Vec

Ο παραπάνω γράφος εισάγεται ως παράμετρος στον αλγόριθμο **Node2Vec**, ο οποίος μέσω προσομοιωμένων τυχαίων περιπάτων και μοντέλων τύπου Word2Vec παράγει **χαμηλοδιάστατα διανύσματα (embeddings)** για κάθε ετικέτα. Τα διανύσματα αυτά κωδικοποιούν τις σχέσεις συσχέτισης μεταξύ ετικετών.
Ο προαναφερθείς γράφος χρησιμοποιείται ως είσοδος στον αλγόριθμο **Node2Vec**, ο οποίος εφαρμόζει προσομοιωμένους τυχαίους περιπάτους (simulated random walks) στον γράφο για τη διερεύνηση τοπικών και καθολικών δομών.
Τα μοτίβα που προκύπτουν από αυτούς τους περιπάτους αξιοποιούνται σε μοντέλα τύπου **Word2Vec**, με σκοπό την παραγωγή χαμηλοδιάστατων διανυσμάτων χαρακτηριστικών (embeddings) για κάθε κόμβο, δηλαδή για κάθε ετικέτα του γράφου. Τα διανύσματα αυτά κωδικοποιούν πληροφορίες σχετικά με τη συσχέτιση και τη δομική σχέση μεταξύ των ετικετών, επιτρέποντας την περαιτέρω ανάλυση και την αξιοποίηση τους σε διαδικασίες μηχανικής μάθησης.

```
# ===== Ενσωμάτωση Ετικετών με Node2Vec =====
##node2vec = Node2Vec(G, dimensions=64, walk_length=10, num_walks=100, workers=1)
##node2vec = Node2Vec(G, dimensions=64, walk_length=10, num_walks=100, workers=2)
node2vec = Node2Vec(G, dimensions=128, walk_length=20, num_walks=200, workers=4)
model = node2vec.fit(window=5, min_count=1, batch_words=4)

label_embeddings = np.array([model.wv[str(label)] for label in range(y_train.shape[1])])
```

3.5.2.6 Ομαδοποίηση Ετικετών με KMeans (Clustering)

Οι παραγόμενες ενσωματώσεις των ετικετών (label embeddings), οι οποίες κωδικοποιούν τη σημασιολογική συσχέτιση μεταξύ τους με βάση τον γράφο συνεμφάνισης, χρησιμοποιούνται ως είσοδος σε έναν αλγόριθμο ομαδοποίησης τύπου KMeans. Ο αλγόριθμος αυτός ομαδοποιεί τις ετικέτες σε διακριτές ομάδες (clusters), όπου κάθε ομάδα περιλαμβάνει τις ετικέτες που παρουσιάζουν μεταξύ τους υψηλή ομοιότητα στη δομή και το περιεχόμενο, όπως αυτό έχει προκύψει από την ανάλυση της συνεμφάνισης.

Η διαδικασία αυτή επιτρέπει τη μείωση της διάστασης του προβλήματος ταξινόμησης, καθώς το αρχικό πλήθος ετικετών (π.χ. 174 για το dataset CAL500) συμπυκνώνονται σε ένα μικρότερο σύνολο ομάδων. Το ποσοστό της μείωσης προκύπτει κάθε φορά συναρτήσει του αρχικού πλήθους των ετικετών. Έχουμε ορίσει τα εξής κριτήρια:

- Για labels<10, το πλήθος των clusters να είναι στο 50% του αρχικού πλήθους των ετικετών.
- Για 10<labels<20, το πλήθος των clusters να είναι ακριβώς 10.
- Για labels>20, το πλήθος των clusters να είναι στο 30% του αρχικού πλήθους των ετικετών.

Στην συγκεκριμένη περίπτωση, όπως απεικονίζεται στο τμήμα κώδικα που ακολουθεί, το τελικό πλήθος των ομάδων ισοδυναμεί στο 30% του αρχικού πλήθους των ετικετών (~52 clusters). Με αυτόν τον τρόπο, διευκολύνεται η επεξεργασία και βελτιώνεται η αποδοτικότητα των επόμενων βημάτων του κώδικα, ενώ παράλληλα διατηρείται η βασική πληροφορία που περιέχεται στις αρχικές ετικέτες.

```
# ===== Clustering για Μείωση Ετικετών =====
num_new_labels = int(numOfLabels*0.3) #για labels>20
##num_new_labels = int(numOfLabels*0.5) #για labels<10
##num_new_labels = 10 #για labels μεταξύ 10-20

kmeans = KMeans(n_clusters=num_new_labels, random_state=42)
label_clusters = kmeans.fit_predict(label_embeddings)
```

3.5.2.7 Συγχώνευση Ετικετών

Οι αρχικές ετικέτες κάθε εγγραφής μετατρέπονται σε ένα νέο δυαδικό διάνυσμα όπου κάθε θέση δηλώνει αν η εγγραφή ανήκει στο αντίστοιχο cluster. Αν τουλάχιστον μία από τις αρχικές ετικέτες μιας εγγραφής ανήκει σε ένα cluster, τότε η εγγραφή λαμβάνει τη νέα συνδυαστική ετικέτα.
Σε αυτό το στάδιο, οι αρχικές ετικέτες που αντιστοιχούν σε κάθε εγγραφή του συνόλου δεδομένων μετασχηματίζονται σε ένα νέο δυαδικό διάνυσμα χαρακτηριστικών, το οποίο αναπαριστά την αντιστοίχιση της εγγραφής στα clusters ετικετών, όπως έχουν προκύψει. Συγκεκριμένα, το μήκος του νέου διανύσματος ισούται με τον αριθμό των clusters που έχουν δημιουργηθεί προηγουμένως μέσω της διαδικασίας ομαδοποίησης (clustering). Κάθε θέση στο διάνυσμα αυτό υποδεικνύει αν η εγγραφή σχετίζεται ή όχι με το αντίστοιχο cluster.
Η εκχώρηση της νέας αυτής συνδυαστικής ετικέτας γίνεται με τον ακόλουθο τρόπο: εάν τουλάχιστον μία από τις αρχικές ετικέτες που ανήκουν στην εγγραφή περιλαμβάνεται σε ένα συγκεκριμένο cluster, τότε το αντίστοιχο στοιχείο του νέου δυαδικού διανύσματος παίρνει την τιμή 1, δηλώνοντας την παρουσία της εγγραφής σε αυτό το cluster. Αντίθετα, εάν καμία από τις αρχικές ετικέτες δεν ανήκει στο cluster αυτό, το στοιχείο παίρνει την τιμή 0. Με αυτόν τον τρόπο, επιτυγχάνεται η σύνοψη και συγχώνευση πολλών ετικετών σε λιγότερα και πιο γενικευμένα σύνολα, μειώνοντας έτσι την πολυπλοκότητα του προβλήματος και διευκολύνοντας τις επόμενες φάσεις της ανάλυσης και της ταξινόμησης.

```
# ===== Συγχώνευση Labels σύμφωνα με τα Clusters =====
def merge_labels(y, label_clusters):
    new_y = np.zeros((y.shape[0], num_new_labels), dtype=int)
    for i, sample_labels in enumerate(y):
        active_labels = np.where(sample_labels == 1)[0]
        for label in active_labels:
            new_cluster = label_clusters[label]
            new_y[i, new_cluster] = 1
    return new_y

# Εφαρμογή στους πίνακες train και test
y_train_new = merge_labels(y_train, label_clusters)
y_test_new = merge_labels(y_test, label_clusters)
```

3.5.2.8 Εμπλουτισμός Χαρακτηριστικών με Embeddings

Για κάθε εγγραφή τόσο στο υποσύνολο εκπαίδευσης (training set) όσο και στο υποσύνολο δοκιμής (test set), υπολογίζεται το μέσο διάνυσμα ενσωμάτωσης (mean embedding) των ετικετών οι οποίες σχετίζονται με αυτήν. Συγκεκριμένα, λαμβάνονται υπόψη οι ενσωματώσεις των ετικετών που αντιστοιχούν στην εγγραφή και υπολογίζεται ο μέσος όρος τους ως διανυσματική παράσταση.
Το παραγόμενο μέσο διάνυσμα ενσωμάτωσης προστίθεται στα αρχικά χαρακτηριστικά εισόδου της εγγραφής, με σκοπό να εμπλουτίσει το αρχικό σύνολο δεδομένων με πρόσθετη πληροφορία που απορρέει από τις σχέσεις και τις συσχετίσεις μεταξύ των ετικετών. Με αυτόν τον τρόπο, η είσοδος των δεδομένων (input) αποκτά ένα πιο πλούσιο και αντιπροσωπευτικό περιεχόμενο, το οποίο μπορεί να βελτιώσει την ικανότητα των αλγορίθμων μηχανικής μάθησης να εντοπίζουν μοτίβα και να πραγματοποιούν πιο ακριβείς προβλέψεις.

```
# ===== Εμπλουτισμός Χαρακτηριστικών με Embeddings =====
def transform_with_embeddings(X, y, label_embeddings):
    transformed_features = []
    for i in range(X.shape[0]):
        labels = np.where(y[i] == 1)[0]
        if len(labels) > 0:
            embedded_vector = np.mean(label_embeddings[labels], axis=0)
        else:
            embedded_vector = np.zeros(label_embeddings.shape[1])
        combined_features = np.hstack((X[i].flatten(), embedded_vector))
        transformed_features.append(combined_features)
    return np.array(transformed_features)

X_train_embedded = transform_with_embeddings(X_train, y_train, label_embeddings)
X_test_embedded = transform_with_embeddings(X_test, y_test, label_embeddings)
```

3.5.2.9 Τελική Εκπαίδευση και Αξιολόγηση

Ο ταξινομητής **BRkNNaClassifier** επανεκπαιδεύεται χρησιμοποιώντας το ενημερωμένο σύνολο δεδομένων, το οποίο περιλαμβάνει τόσο τα αρχικά χαρακτηριστικά εισόδου όσο και τα εμπλουτισμένα χαρακτηριστικά που προκύπτουν από τις ενσωματώσεις των ετικετών. Παράλληλα, οι αρχικές ετικέτες έχουν αντικατασταθεί από τις συγχωνευμένες ετικέτες, όπως προέκυψαν από τη διαδικασία ομαδοποίησης και συγχώνευσης.

```
# ===== Εκπαίδευση Ταξινομητών =====
clfBrKnn = BRkNNaClassifier(k=num_new_labels)
clfBrKnn.fit(X_train_embedded, y_train_new)  # Ekpaidefsi tou taksinomiti
BRkNN_predictions = clfBrKnn.predict(csr_matrix(X_test_embedded))
```

Η απόδοση του ταξινομητή αξιολογείται εκ νέου με τη χρήση των κατάλληλων μετρικών. Τα αποτελέσματα δείχνουν σημαντική μείωση του δείκτη Hamming Loss, γεγονός που υποδηλώνει βελτίωση της ακρίβειας των προβλέψεων. Η παρατηρούμενη αυτή βελτίωση αποδίδεται αφενός στην ενσωμάτωση πληροφοριών σχετικά με τις συσχετίσεις μεταξύ των ετικετών μέσω των ενσωματώσεων και αφετέρου στη μείωση της διάστασης του προβλήματος μέσω της συγχώνευσης των ετικετών.

```
# ===== Αξιολόγηση Απόδοσης =====
print("Hamming Loss BRkNN(με νέα labels):", hamming_loss(y_test_new, BRkNN_predictions.astype(int)))
end_time = time.time()
print("Accuracy BRkNN(με νέα labels):", accuracy_score(y_test_new, BRkNN_predictions.astype(int)))
print("Process Time After Embeddings:", end_time - start_time, "seconds")
```

3.5.2.10 Οπτικοποίηση

Για την καλύτερη κατανόηση και οπτικοποίηση της πληροφορίας που περιέχεται στον γράφο συνεμφάνισης ετικετών, πραγματοποιούνται οι εξής ενέργειες:

- Αρχικά, απεικονίζεται ο αρχικός γράφος συνεμφάνισης, όπου κάθε κόμβος αντιστοιχεί σε μία ετικέτα και οι ακμές αναπαριστούν τις συσχετίσεις μεταξύ των ετικετών βάσει της συχνότητας της συνεμφάνισής τους στα δεδομένα.
- Στη συνέχεια, δημιουργείται και απεικονίζεται ένας νέος γράφος, στον οποίο κάθε κόμβος αντιστοιχεί σε ένα cluster ετικετών ο οποίος προκύπτει από τη διαδικασία ομαδοποίησης. Οι ακμές αυτού του γράφου αναπαριστούν τις σχέσεις και τις αλληλεπιδράσεις μεταξύ των clusters, όπως αυτές προέκυψαν από τις αρχικές συσχετίσεις μεταξύ των επιμέρους ετικετών.

Με αυτήν τη διπλή απεικόνιση, διευκολύνεται η ανάλυση της δομής και των σχέσεων σε διαφορετικά επίπεδα ομαδοποίησης, προσφέροντας μια πιο ολοκληρωμένη εικόνα της δομικής πληροφορίας που περιέχει ο γράφος.

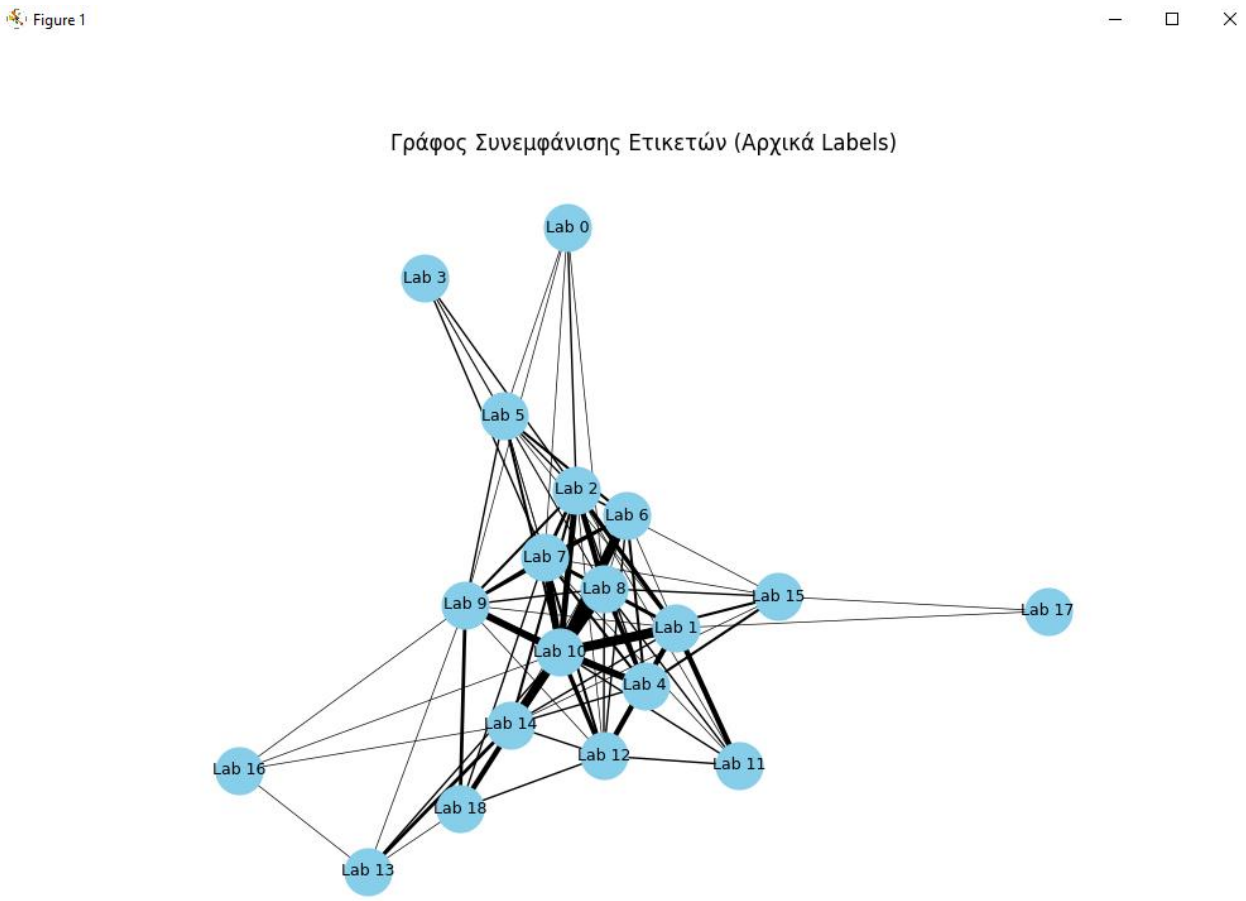
```
# ===== Οπτικοποίηση =====
# 1. Αρχικός Γράφος
plt.figure(figsize=(10, 8))
pos = nx.spring_layout(G, seed=42)

nx.draw_networkx_nodes(G, pos, node_size=700, node_color='skyblue')
edges = G.edges(data=True)
weights = [edge[2]['weight'] for edge in edges]
nx.draw_networkx_edges(G, pos, edgelist=G.edges(), width=[w * 0.5 for w in weights])

# Ετικέτες των αρχικών labels
initial_labels = {i: f'Label {i}' for i in G.nodes()}
nx.draw_networkx_labels(G, pos, labels=initial_labels, font_size=9, font_color='yellow')

plt.title("Γράφος Συνεμφάνισης Ετικετών (Αρχικά Labels)")
plt.axis('off')
plt.show()
```

Για να δώσουμε ένα παράδειγμα οπτικού αποτελέσματος του αρχικού γράφου, χρησιμοποιούμε το σύνολο δεδομένων Birds το οποίο περιλαμβάνει 19 αρχικά labels.



Εικόνα 4: Οπτικοποίηση του γράφου πριν τη συγχώνευση

Ακολουθεί το τμήμα κώδικα που αφορά στην δημιουργία του νέου γράφου μετά τις συγχωνεύσεις

```
# 2. Γράφος μετά τη Συγχώνευση Labels
# Δημιουργία Νέου Γράφου για τα Clusters
G_new = nx.Graph()

# Προσθήκη κόμβων (clusters)
for cluster_id in range(num_new_labels):
    G_new.add_node(cluster_id)

# Δημιουργία ακμών μεταξύ clusters αν υπήρχαν σχέσεις στα αρχικά δεδομένα
for (label1, label2, data) in G.edges(data=True):
    cluster1 = label_clusters[label1]
    cluster2 = label_clusters[label2]
    if cluster1 != cluster2:  # Ενώνουμε διαφορετικά clusters
        if G_new.has_edge(cluster1, cluster2):
            G_new[cluster1][cluster2]['weight'] += data['weight']
        else:
            G_new.add_edge(cluster1, cluster2, weight=data['weight'])
```

Και το τμήμα του κώδικα που αφορά στην οπτικοποίηση του νέου γράφου.

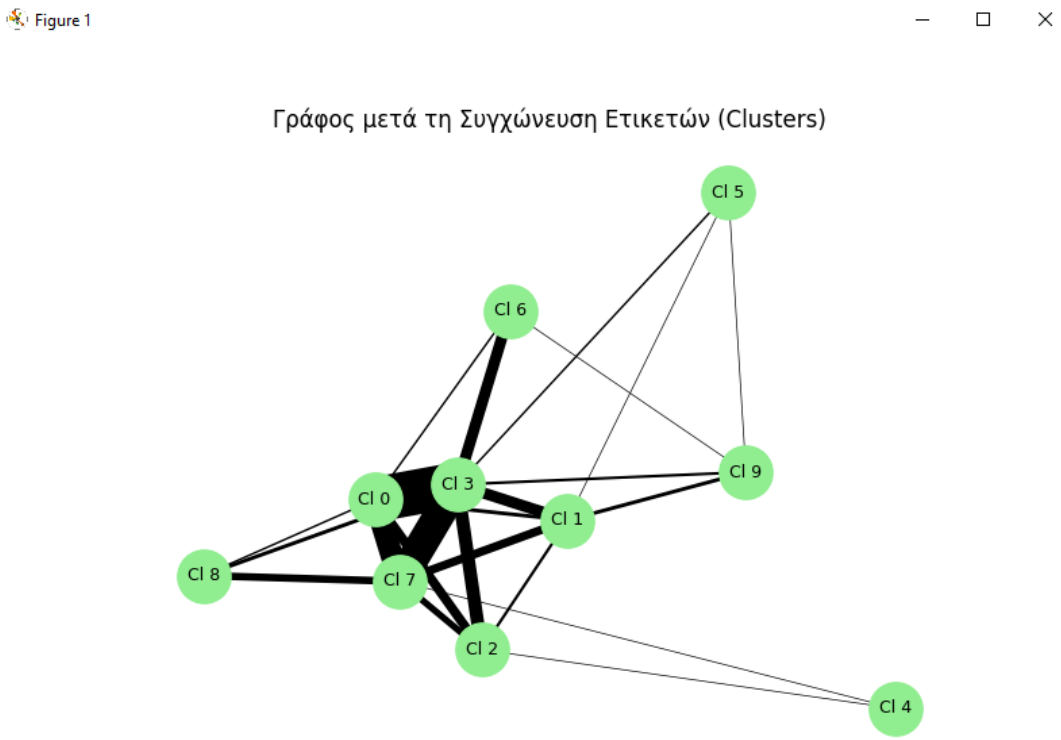
```
# Οπτικοποίηση Νέου Γράφου
plt.figure(figsize=(8, 6))
pos_new = nx.spring_layout(G_new, seed=42)

nx.draw_networkx_nodes(G_new, pos_new, node_size=800, node_color='lightgreen')
new_weights = [edge[2]['weight'] for edge in G_new.edges(data=True)]
nx.draw_networkx_edges(G_new, pos_new, width=[w * 0.5 for w in new_weights])

# Ετικέτες των νέων clusters
new_labels = {i: f'Cluster {i}' for i in G_new.nodes()}
nx.draw_networkx_labels(G_new, pos_new, labels=new_labels, font_size=9, font_color='red')

plt.title("Γράφος μετά τη Συγχώνευση Ετικετών (Clusters)")
plt.axis('off')
plt.show()
```

Τέλος, παρουσιάζεται το οπτικό αποτέλεσμα του νέου γράφου, όπου φαίνονται το πλήθος των clusters και οι συσχετίσεις τους.



Εικόνα 5: Οπτικοποίηση του γράφου μετά τη συγχώνευση

3.6 Ανάπτυξη Κώδικα για MLkNN

3.6.1 Θεωρητικό Υπόβαθρο

Ο αλγόριθμος MLkNN (Multi-Label k-Nearest Neighbors) αποτελεί μία από τις πιο διαδεδομένες και αποτελεσματικές προσεγγίσεις για την αντιμετώπιση προβλημάτων στην ταξινόμηση πολλαπλών ετικετών. Πρόκειται για επέκταση του κλασικού αλγορίθμου k-Nearest Neighbors (kNN) σε ένα πλαίσιο, όπου κάθε δείγμα μπορεί να σχετίζεται με περισσότερες από μία ετικέτες.

Η βασική λειτουργία του MLkNN συνίσταται στον εντοπισμό των k πλησιέστερων γειτόνων ενός δείγματος, βάσει κάποιας μετρικής απόστασης, συνήθως της ευκλείδειας απόστασης. Στη συνέχεια, για κάθε ετικέτα, εκτιμάται η πιθανότητα εμφάνισής της στο υπό πρόβλεψη δείγμα, λαμβάνοντας υπόψη τη συχνότητα με την οποία εμφανίζεται στους γείτονες αυτούς. Η απόφαση για το αν μια ετικέτα θα ανατεθεί στο δείγμα βασίζεται σε μια πιθανοθεωρητική διαδικασία η οποία συγκρίνει την πιθανότητα εμφάνισης και μη εμφάνισης της ετικέτας υπό την προϋπόθεση ότι η ετικέτα εμφανίζεται j φορές στους k γείτονες. Με άλλα λόγια, ο αλγόριθμος υιοθετεί μια Bayesιανή προσέγγιση, αξιοποιώντας τόσο τις εκ των προτέρων (prior) όσο και τις υπό συνθήκη (conditional) πιθανότητες, προκειμένου να εκτιμήσει την καταλληλότερη ταξινόμηση.

Ο MLkNN παρουσιάζει πλεονεκτήματα σε σχέση με άλλες τεχνικές, κυρίως λόγω της απλότητας στην υλοποίηση, της αποδοτικότητας στην εκπαίδευση και της ικανότητάς του να χειρίζεται προβλήματα με μεγάλο αριθμό ετικετών. Επιπλέον, χαρακτηρίζεται ως μη παραμετρικός αλγόριθμος, αφού οι μόνοι παράμετροι που χρειάζεται να οριστούν εκ των προτέρων είναι η τιμή του k (ο αριθμός των γειτόνων) και ένας παράγοντας smoothing (s), που χρησιμοποιείται για τη ρύθμιση των εκτιμώμενων πιθανοτήτων ώστε να αποφεύγονται προβλήματα μηδενικής συχνότητας.

Παρόλα αυτά, ο MLkNN αντιμετωπίζει σημαντικές προκλήσεις όταν το πλήθος των ετικετών είναι πολύ μεγάλο, καθώς η πολυπλοκότητα και ο υπολογιστικός φόρτος αυξάνονται σημαντικά. Για την αντιμετώπιση αυτών των περιορισμών, η χρήση μεθόδων μείωσης της πολυπλοκότητας και ενσωμάτωσης συμπληρωματικής πληροφορίας, όπως η δημιουργία ενσωματώσεων και η ομαδοποίηση (clustering) των ετικετών, μπορεί να συμβάλλει στην αύξηση της αποδοτικότητας και της ακρίβειας του μοντέλου.

3.6.2 Ανάπτυξη Λειτουργικότητας σε Python

Η ανάπτυξη της λειτουργικότητας με χρήση του ταξινομητή MLkNN (Multi-Label k-Nearest Neighbors) ακολουθεί συγκεκριμένα βήματα, τα οποία περιλαμβάνουν την προεπεξεργασία των δεδομένων, την εκπαίδευση του μοντέλου, την πρόβλεψη των ετικετών για το υποσύνολο δοκιμών και την αξιολόγηση των αποτελεσμάτων βάσει μετρικών για την ταξινόμηση πολλαπλών ετικετών.

Όπως και στην περίπτωση ανάπτυξης της λειτουργικότητα με χρήση του αλγορίθμου BRkNN, κατά την ενσωμάτωση του ταξινομητή MLkNN της βιβλιοθήκης scikit-multilearn, παρουσιάστηκαν προβλήματα συμβατότητας με εκδόσεις της scikit-learn. Συγκεκριμένα:

- Η κλάση MLkNN χρησιμοποιεί τη μέθοδο get_params()της scikit-learn, η οποία βασίζεται στη μέθοδο _get_param_names(). Στην έκδοση της scikit-learn που χρησιμοποιήθηκε, η μέθοδος αυτή δεν ήταν διαθέσιμη ή είχε αλλάξει υπογραφή.
- Επιπλέον, η NearestNeighbors απαιτούσε παράμετρο σε μορφή tuple ή dict, και όχι σε μορφή ακεραίου.

Για την επίλυση αυτών των ασυμβατοτήτων εφαρμόστηκε Monkey Patching, δηλαδή δυναμική τροποποίηση των μεθόδων των βιβλιοθηκών κατά το χρόνο εκτέλεσης. Συγκεκριμένα:

- Ορίστηκε νέα μέθοδος _get_param_names() στην BaseEstimator ώστε να επιστρέφει τις μεταβλητές αρχικοποίησης εκτός του self.
- Τροποποιήθηκε η μέθοδος __init__() της NearestNeighbors ώστε να μπορεί να δέχεται ακέραια τιμή για το k.

```
# ===== Monkey Patch για MLkNN, BRkNN =====
from sklearn.base import BaseEstimator
def _get_param_names(cls):
    return [key for key in cls.__init__.__code__.co_varnames if key != 'self']
BaseEstimator._get_param_names = classmethod(_get_param_names)

# Monkey Patch για NearestNeighbors
_original_init = NearestNeighbors.__init__
def patched_init(self, *args, **kwargs):
    if len(args) == 1 and not kwargs.get('n_neighbors'):
        kwargs['n_neighbors'] = args[0]
        args = ()
    _original_init(self, *args, **kwargs)
NearestNeighbors.__init__ = patched_init
```

Με αυτόν τον τρόπο κατέστη δυνατή η συνέχιση της υλοποίησης χωρίς ανάγκη για αναβάθμιση ή τροποποίηση της εγκατάστασης των βιβλιοθηκών.

Η διαδικασία σε πολλά σημεία παρουσιάζει ομοιότητες με αυτήν που εφαρμόσαμε στον ταξινομητή BRkNN (Binary Relevance k-Nearest Neighbors), ως προς την υποδομή και την αξιοποίηση του kNN πυρήνα, αλλά και σημαντικές διαφορές στη μοντελοποίηση των ετικετών.

Αρχικά, το σύνολο δεδομένων μετατράπηκε από ARFF μορφή σε DataFrame, όπου τα χαρακτηριστικά και οι ετικέτες διαχωρίστηκαν κατάλληλα. Τα χαρακτηριστικά μετατράπηκαν σε αριθμητικούς πίνακες, ενώ οι ετικέτες μετατράπηκαν σε δυαδική μορφή (binary), όπως απαιτείται για τους περισσότερους πολυετικετικούς ταξινομητές.

Στη συνέχεια, το σύνολο δεδομένων χωρίστηκε σε δύο υποσύνολα εκπαίδευσης και ελέγχου σε αναλογία70/30, και ακολούθησε η βασική εκπαίδευση του μοντέλου MLkNN με τιμή k=1. Παρόμοια διαδικασία ακολουθείται και για μία δεύτερη μέτρηση με k= 174, δηλαδή για αριθμό γειτόνων ίσο με τον αριθμό των αρχικών ετικετών, καθώς η τιμή αυτή παρατηρείται αποδίδει το χαμηλότερο Hamming Loss. Τόσο ο BRkNN που χρησιμοποιήσαμε στην προηγούμενη παράγραφο όσο και ο MLkNN βασίζονται στην αρχή της εύρεσης των k πλησιέστερων γειτόνων, με τη διαφορά ότι ο MLkNN προσεγγίζει την πρόβλεψη μέσω πιθανολογικού μοντέλου για κάθε ετικέτα, ενώ ο BRkNN υλοποιεί ξεχωριστό kNN ταξινομητή για κάθε ετικέτα βάσει της μεθόδου Binary Relevance.

```
# ===== Διαχωρισμός σε Training και Test Set =====
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
start_time = time.time()

#Ypologismos Before
##classifierMLkNN = MLkNN(k=1)  # Εκτέλεση του MLkNN με k=1, αρχικα
classifierMLkNN = MLkNN(k=174) #gia k=174 dinei to xamilotero hamming loss
classifierMLkNN.fit(X_train, y_train)  # Fit the classifier

#PREDICTION on the test set
y_pred = classifierMLkNN.predict(X_test)  # Predict the labels for the test set
```

Μετά την αρχική εκπαίδευση και αξιολόγηση, εφαρμόστηκε επιπλέον επεξεργασία που αφορά στη δομική ενσωμάτωση πληροφορίας συσχέτισης μεταξύ ετικετών, μέσω της δημιουργίας γράφου συνεμφάνισης ετικετών (label co-occurrence graph). Ο γράφος αυτός αξιοποιεί τις συχνότητες συνεμφάνισης των ετικετών στο training set και ενσωματώνει τη συσχετισμική πληροφορία μέσω του Node2Vec για την παραγωγή ενσωματώσεων (embeddings) των ετικετών. Η ίδια στρατηγική εφαρμόστηκε τόσο στον MLkNN όσο και στον BRkNN, επιτρέποντας άμεση σύγκριση της απόδοσης των δύο ταξινομητών πριν και μετά τη χρήση embedding-based enrichment.

Ακολούθησε ομαδοποίηση των ετικετών (clustering) με τη χρήση του αλγορίθμου KMeans, όπου οι ενσωματώσεις χρησιμοποιούνται ως είσοδος για τον ομαδοποιητή. Η διαδικασία αυτή στοχεύει στη μείωση του πλήθους των ετικετών μέσω συγχώνευσης, ώστε να αντιμετωπιστεί η πολυπλοκότητα και η αραιότητα των δεδομένων. Οι νέες, συγχωνευμένες ετικέτες προκύπτουν από την ένωση των αρχικών ετικετών που ανήκουν στο ίδιο cluster, γεγονός που οδηγεί σε νέους δυαδικούς πίνακες (binary matrices) για τα υποσύνολα εκπαίδευσης και δοκιμής. Αυτή η μορφή μετατροπής των ετικετών εφαρμόζεται πανομοιότυπα και στους δύο ταξινομητές, MLkNN και BRkNN.

Επιπλέον, δημιουργήθηκαν νέα χαρακτηριστικά μέσω εμπλουτισμού των αρχικών χαρακτηριστικών εισόδου με τις μέσες ενσωματώσεις (mean label embeddings) των ετικετών κάθε δείγματος. Η διαδικασία αυτή αποτελεί ένα είδος επαύξησης των χαρακτηριστικών που ενσωματώνει τη γνώση της σχέσης μεταξύ χαρακτηριστικών και ετικετών στο χώρο εισόδου. Το βήμα αυτό είναι κρίσιμο για την ενίσχυση της εκφραστικότητας των δεδομένων εισόδου και εφαρμόζεται κατά τα ίδια πρότυπα τόσο στον MLkNN όσο και στον BRkNN.

```
def transform_with_embeddings(X, y, label_embeddings):
    transformed_features = []
    for i in range(X.shape[0]):
        labels = np.where(y[i] == 1)[0]
        if len(labels) > 0:
            embedded_vector = np.mean(label_embeddings[labels], axis=0)
        else:
            embedded_vector = np.zeros(label_embeddings.shape[1])
        combined_features = np.hstack((X[i].flatten(), embedded_vector))
        transformed_features.append(combined_features)
    return np.array(transformed_features)

X_train_embedded = transform_with_embeddings(X_train, y_train, label_embeddings)
X_test_embedded = transform_with_embeddings(X_test, y_test, label_embeddings)
```

Τέλος, εκπαιδεύτηκε εκ νέου ο ταξινομητής MLkNN, χρησιμοποιώντας τα εμπλουτισμένα χαρακτηριστικά και τις νέες ετικέτες.

```
# ===== Εκ νέου Εκπαίδευση Ταξινομητών =====
clf_mlknn = MLkNN(k=num_new_labels, s=1.0) # νέα μειωμένη τιμή labels
clf_mlknn.fit(csr_matrix(X_train_embedded), y_train_new)
mlknn_predictions = clf_mlknn.predict(csr_matrix(X_test_embedded))
```

Η αξιολόγηση απόδοσης πραγματοποιήθηκε με χρήση των μετρικών Hamming Loss και Accuracy, καθώς και μέτρηση χρόνου επεξεργασίας πριν και μετά την ενσωμάτωση embeddings, καθιστώντας εφικτή τη σύγκριση των ταξινομητών τόσο σε όρους ακρίβειας όσο και αποδοτικότητας.

```
# ===== Αξιολόγηση Απόδοσης =====
print("Hamming Loss MLkNN(με νέα labels):", hamming_loss(y_test_new, mlknn_predictions.astype(int)))
end_time = time.time()
print("Accuracy MLkNN(με νέα labels):", accuracy_score(y_test_new, mlknn_predictions.astype(int)))
print("Process Time After Embeddings:", end_time - start_time, "seconds")
```

Συνοψίζοντας, η πειραματική μεθοδολογία για τους ταξινομητές MLkNN και BRkNN παρουσιάζει υψηλό βαθμό δομικής και υπολογιστικής ομοιότητας, ενώ οι βασικές διαφορές εντοπίζονται στον τρόπο χειρισμού των ετικετών (πιθανολογικό vs binary relevance μοντέλο). Η υιοθέτηση κοινών τεχνικών ενσωμάτωσης γνώσης ετικετών μέσω γραφημάτων, ενσωματώσεων και ομαδοποίησης εξασφαλίζει μια συνεκτική πειραματική βάση για αξιόπιστη συγκριτική αξιολόγηση.

3.7 Ανάπτυξη Κώδικα για MLARAM

3.7.1 Θεωρητικό Υπόβαθρο

Ο αλγόριθμος MLARAM (Multi-Label Adaptive Resonance Associative Map) συνιστά μια προηγμένη προσέγγιση ταξινόμησης πολλαπλών ετικετών, η οποία βασίζεται στις αρχές της Adaptive Resonance Theory (ART), ενός θεωρητικού πλαισίου το οποίο προέρχεται από τον χώρο των τεχνητών νευρωνικών δικτύων. Η ART αναπτύχθηκε με σκοπό να εξισορροπεί τη σταθερότητα και την πλαστικότητα της μάθησης, επιτρέποντας την απόκτηση νέας πληροφορίας χωρίς την απώλεια της ήδη αποκτηθείσας γνώσης. Στο πλαίσιο αυτό, ο MLARAM υιοθετεί μια μη-εποπτική και επεκτάσιμη αρχιτεκτονική, η οποία δύναται να προσαρμόζεται δυναμικά σε νέες εισροές δεδομένων, χωρίς την ανάγκη πλήρους επανεκπαίδευσης του μοντέλου. Η ιδιότητα αυτή καθιστά τον MLARAM ιδιαίτερα κατάλληλο για εφαρμογές με ροή δεδομένων σε πραγματικό χρόνο (data streams) ή σε σενάρια όπου η διαρκής επικαιροποίηση του μοντέλου είναι κρίσιμη.

Ο MLARAM διαφοροποιείται από άλλες μεθόδους ταξινόμησης πολλαπλών ετικετών ως προς το γεγονός ότι ενσωματώνει εννοιολογικά πρότυπα (conceptual prototypes) κατά τη διαδικασία εκπαίδευσης, διατηρώντας για κάθε κατηγορία μια χωρική αναπαράσταση στο χώρο των χαρακτηριστικών. Κάθε νέο δείγμα συγκρίνεται με τα υφιστάμενα πρότυπα και είτε εντάσσεται σε κάποιο από αυτά (σε περίπτωση ικανοποιητικής ομοιότητας), είτε προκαλεί τη δημιουργία ενός νέου προτύπου. Αυτή η αναδραστική διαδικασία προσαρμογής επιτρέπει στο μοντέλο να παραμένει ευέλικτο ως προς τη διαφοροποίηση της πληροφορίας που συναντά στα δεδομένα, ενώ παράλληλα διατηρεί υψηλή δυνατότητα ερμηνείας, καθώς τα πρωτότυπα μπορούν να αναλυθούν και να συσχετιστούν με συγκεκριμένες περιοχές του χώρου των χαρακτηριστικών.

Η εφαρμογή του MLARAM σε προβλήματα ταξινόμησης πολλαπλών ετικετών περιλαμβάνει την απευθείας συσχέτιση εισόδου-εξόδου μέσω του εννοιολογικού χάρτη (associative map), επιτρέποντας την ταυτόχρονη ανάθεση πολλαπλών ετικετών σε ένα δείγμα. Η προσέγγιση αυτή έχει το πλεονέκτημα ότι λαμβάνει υπόψη ενδογενώς τις πιθανές συσχετίσεις μεταξύ ετικετών, κάτι που απουσιάζει από μεθόδους τύπου Binary Relevance, όπου κάθε ετικέτα αντιμετωπίζεται ανεξάρτητα. Παράλληλα, η ευελιξία του μοντέλου το καθιστά κατάλληλο για προβλήματα όπου οι ετικέτες παρουσιάζουν επικαλυπτόμενες περιοχές στο χώρο των χαρακτηριστικών.

Ωστόσο, σε περιβάλλοντα υψηλών διαστάσεων και μεγάλης πολυπλοκότητας —χαρακτηριστικά που είναι συνηθισμένα σε σύνολα δεδομένων πολλαπλών ετικετών— κρίνεται αναγκαία η εφαρμογή

προσεγγίσεων μείωσης διαστάσεων και ενισχυμένης αναπαράστασης. Συγκεκριμένα, τεχνικές όπως η χρήση ενσωματωμένων αναπαραστάσεων (embeddings) και η ομαδοποίηση των ετικετών μέσω clustering μπορούν να βελτιώσουν σημαντικά τη λειτουργικότητα και την αποδοτικότητα του MLARAM. Οι ενσωματώσεις επιτρέπουν τη μεταφορά της πληροφορίας των συσχετίσεων μεταξύ ετικετών ή δειγμάτων σε έναν πυκνότερο και εκφραστικότερο χώρο χαρακτηριστικών, ενώ η ομαδοποίηση (clustering) συμβάλλει στη μείωση της πολυπλοκότητας της εξόδου, συνενώνοντας ετικέτες με παρόμοια σημασιολογική ή στατιστική συμπεριφορά.

Συνεπώς, ο MLARAM αποτελεί μια εύρωστη και προσαρμοστική λύση για προβλήματα Πολλαπλών ετικετών, συνδυάζοντας την υπολογιστική αποτελεσματικότητα της ART με την ικανότητα διατήρησης εννοιολογικής δομής, καθιστώντας τον κατάλληλο για ένα ευρύ φάσμα εφαρμογών, από δυναμικά συστήματα πληροφορίας έως ανάλυση κειμένου, εικόνας ή βιοϊατρικών δεδομένων.

3.7.2 Ανάπτυξη Λειτουργικότητας σε Python

Κατά την εκτέλεση του πειραματικού μέρους με τον ταξινομητή MLARAM, παρουσιάστηκε πρόβλημα συμβατότητας που σχετίζεται με τη χρήση παρωχημένης συνάρτησης από τη βιβλιοθήκη Scipy. Πιο συγκεκριμένα, το αρχείο mlaram.py, το οποίο βρίσκεται στο path skmultilearn/adapt/mlaram.py, καλεί τη συνάρτηση scipy.ones(), η οποία δεν υποστηρίζεται πλέον στις νεότερες εκδόσεις της βιβλιοθήκης Scipy. Το σφάλμα αυτό οφείλεται στο γεγονός ότι η scipy.ones() έχει καταργηθεί και η χρήση της έχει αντικατασταθεί από την αντίστοιχη συνάρτηση της βιβλιοθήκης NumPy, δηλαδή την numpy.ones().

Για την αποκατάσταση της λειτουργικότητας του ταξινομητή, πραγματοποιήθηκε τροποποίηση στον πηγαίο κώδικα του πακέτου scikit-multilearn. Η διόρθωση αυτή επέτρεψε τη σωστή εκτέλεση του ταξινομητή MLARAM και την ομαλή συνέχιση της πειραματικής διαδικασίας.

Η παρουσίαση της ανάπτυξης του κώδικα η οποία αξιοποιεί το σύνολο δεδομένων CAL500, το οποίο φορτώθηκε από αρχική μορφή ARFF. Ο ίδιος κώδικας εφαρμόστηκε και στα υπόλοιπα σύνολα δεδομένων. Τα χαρακτηριστικά και οι ετικέτες διαχωρίστηκαν καταλλήλως και τα δεδομένα αναλύθηκαν σε εκπαιδευτικό και δοκιμαστικό υποσύνολο σε αναλογία 70%-30%.

Αρχικά, εφαρμόστηκε ο MLARAM χωρίς καμία προεπεξεργασία ή εμπλουτισμό των δεδομένων.

```
#Ypologismos MLARAM Before
classifierMLARAM = MLARAM() #!
classifierMLARAM.fit(X_train, y_train) #!

# PREDICTION on the test set
y_pred = classifierMLARAM.predict(X_test) #!

# EVALUATE performance
print("Hamming Loss MLARAM Before Embeddings:", hamming_loss(y_test, y_pred.astype(int)))
end_time = time.time()
print("Accuracy MLARAM Before Embeddings:", accuracy_score(y_test, y_pred.astype(int)))
print("Process Time Before Embeddings:", end_time - start_time, "seconds")
```

Στη συνέχεια, ακολούθησε ο εμπλουτισμός των δεδομένων με τεχνικές ενσωμάτωσης και ομαδοποίησης των δεδομένων με στόχο τη βελτίωση της απόδοσης του ταξινομητή. Αρχικά, υλοποιήθηκε η μέθοδος build_label_cooccurrence_graph(y) η οποία δέχεται ως είσοδο το υποσύνολο δεδομένων εκπαίδευσης και δημιουργεί έναν γράφο συνεμφάνισης ετικετών (G) όπου οι κόμβοι αντιστοιχούν σε ετικέτες και οι ακμές εκφράζουν την συνεμφάνιση των ετικετών στις ίδιες παρατηρήσεις. Έπειτα εφαρμόζεται η μέθοδος Node2Vec με τη βοήθεια της οποίας οι ετικέτες

χαρτογραφούνται σε έναν συνεχή χώρο χαρακτηριστικών, βάσει της δομής του γράφου συνεμφάνισης ετικετών (G).

Ακολουθεί η ομαδοποίηση των ετικετών με τη βοήθεια της μεθόδου K-Means όπου οι ενσωματώσεις των ετικετών ομαδοποιούνται σε νέα σύνολα με σκοπό της μείωσης της διάστασης του πίνακα των ετικετών.

```
kmeans = KMeans(n_clusters=num_new_labels, random_state=42)
label_clusters = kmeans.fit_predict(label_embeddings)

# ===== Συγχώνευση Labels σύμφωνα με τα Clusters =====
def merge_labels(y, label_clusters):
    new_y = np.zeros((y.shape[0], num_new_labels), dtype=int)
    for i, sample_labels in enumerate(y):
        active_labels = np.where(sample_labels == 1)[0]
        for label in active_labels:
            new_cluster = label_clusters[label]
            new_y[i, new_cluster] = 1
    return new_y
```

Στη συνέχεια, πραγματοποιείται μετασχηματισμός των δεδομένων και εμπλουτισμός των χαρακτηριστικών, όπου τα αρχικά διανύσματα χαρακτηριστικών εμπλουτίζονται με την ενσωματωμένη αναπαράσταση των ενεργών ετικετών κάθε δείγματος. Με αυτόν τον τρόπο παράγονται συνδυασμένα διανύσματα εισόδου που αντικατοπτρίζουν πληροφορία τόσο για τα χαρακτηριστικά όσο και για τη σχετική δομή των ετικετών.

Μετά την ενσωμάτωση της πληροφορίας των ετικετών στα χαρακτηριστικά και τη μείωση του πλήθους των ετικετών, ο ταξινομητής MLARAM επανεκπαιδεύεται και αξιολογείται εκ νέου, βάσει των μετρικών Hamming Loss και της Accuracy. Επιπλέον, καταγράφεται ο χρόνος εκτέλεσης παρέχοντας πλήρη εικόνα για την αποδοτικότητα της μεθόδου

```
# ===== Εκ νέου Εκπαίδευση Ταξινομητών ===== #!
clfMLARAM = MLARAM()
clfMLARAM.fit(X_train_embedded, y_train_new)
MLARAM_predictions = clfMLARAM.predict(X_test_embedded)

print("Hamming Loss MLARAM (με νέα labels):", hamming_loss(y_test_new, MLARAM_predictions.astype(int)))
end_time = time.time()
print("Accuracy MLARAM (με νέα labels):", accuracy_score(y_test_new, MLARAM_predictions.astype(int)))
print("Process Time After Embeddings:", end_time - start_time, "seconds")
```

Τέλος, πραγματοποιείται οπτικοποίηση τόσο του αρχικού γράφου ετικετών όσο και του γραφήματος μετά τη συγχώνευση, αποκαλύπτοντας τη δομή των συσχετίσεων και των clusters που δημιουργήθηκαν. Οι αναπαραστάσεις αυτές συμβάλουν στην καλύτερη κατανόηση της διαδικασίας μετασχηματισμού των ετικετών και της επίδρασής της στα αποτελέσματα ταξινόμησης.

3.8 Ανάπτυξη Κώδικα για MLTSVM

3.8.1 Θεωρητικό Υπόβαθρο

Ο αλγόριθμος MLTSVM (Multi-Label Twin Support Vector Machine) συνιστά μια προηγμένη παραλλαγή του κλασικού Support Vector Machine (SVM), ειδικά σχεδιασμένη για την αντιμετώπιση προβλημάτων ταξινόμησης πολλαπλών ετικετών. Σε αντίθεση με τον παραδοσιακό SVM, ο οποίος επιδιώκει την κατασκευή ενός και μόνο υπερεπιπέδου (hyperplane) διαχωρισμού μεταξύ των κλάσεων, ο MLTSVM υιοθετεί την προσέγγιση των δυαδικών υπερεπιπέδων (twin hyperplanes). Πιο συγκεκριμένα, υπολογίζονται δύο παράλληλα υπερεπίπεδα, καθένα από τα οποία είναι βελτιστοποιημένο με τέτοιο τρόπο ώστε να βρίσκεται πλησιέστερα στο ένα υποσύνολο των δεδομένων και ταυτόχρονα μακριά από το άλλο.

Η γενίκευση αυτής της ιδέας στο πλαίσιο των πολλαπλών ετικετών επιτρέπει στον MLTSVM να διαχειρίζεται αποτελεσματικά περιπτώσεις όπου κάθε δείγμα εισόδου μπορεί να σχετίζεται με πολλαπλές ετικέτες. Για κάθε ετικέτα κατασκευάζεται ένα ανεξάρτητο δυαδικό μοντέλο βασισμένο στη μεθοδολογία twin SVM, και τα αποτελέσματα συνδυάζονται για την τελική ετικετοποίηση του δείγματος. Αυτή η αρχιτεκτονική επιτρέπει στον MLTSVM να μειώνει την πολυπλοκότητα εκμάθησης συγκριτικά με τις συμβατικές μεθόδους SVM, ενώ παράλληλα διατηρεί υψηλά επίπεδα ακρίβειας και υπολογιστικής αποδοτικότητας.

Η συγκεκριμένη μεθοδολογία αποδεικνύεται ιδιαιτέρως χρήσιμη σε προβλήματα όπου οι διαστάσεις των χαρακτηριστικών είναι πολλαπλές και οι συσχετίσεις μεταξύ των ετικετών είναι σύνθετες, όπως συμβαίνει στο σύνολο δεδομένων CAL500. Στην περίπτωση αυτή, κάθε μουσικό κομμάτι μπορεί να περιγράφεται με πληθώρα θεματικών ή συναισθηματικών ετικετών, γεγονός που απαιτεί έναν αλγόριθμο ικανό να συλλάβει και να μοντελοποιήσει τέτοιου είδους πολυπλοκότητα.

3.8.2 Ανάπτυξη Λειτουργικότητας σε Python

Κατά την ενσωμάτωση του αλγορίθμου MLTSVM (Multi-Label Twin Support Vector Machine) στη διαδικασία της ανάπτυξης του κώδικα, προέκυψαν προειδοποιητικά μηνύματα χρόνου εκτέλεσης τα οποία σχετίζονται με αριθμητικά σφάλματα σε διαίρεση. Συγκεκριμένα, κατά την εκτέλεση του κώδικα της βιβλιοθήκης mltsvm.py, εντοπίστηκαν τα εξής μηνύματα:

```
Runtime Warning: divide by zero encountered in scalar divide
```

και

```
Runtime Warning: invalid value encountered in divide.
```

Τα παραπάνω σφάλματα υποδεικνύουν ότι η μεταβλητή self.wk_norms, η οποία χρησιμοποιείται για την κανονικοποίηση των βαρών του ταξινομητή, μπορεί περιέχει μηδενικές τιμές, οδηγώντας σε διαίρεση με το μηδέν και δημιουργία μη αριθμητικών τιμών (NaN). Αυτή η κατάσταση δύναται να επηρεάσει την αξιοπιστία των αποτελεσμάτων, αλλά και να παρεμποδίσει τη σταθερή εκτέλεση του αλγορίθμου.

Για την αντιμετώπιση του προβλήματος, κρίθηκε απαραίτητη η άμεση τροποποίηση του πηγαίου κώδικα της βιβλιοθήκης, και συγκεκριμένα του αρχείου mltsvm.py, στις γραμμές 125 και 133. Η τροποποίηση περιλαμβάνει την εισαγωγή ενός πολύ μικρού θετικού αριθμού (γνωστού ως epsilon, π.χ. 1e-10) στον παρονομαστή των σχετικών πράξεων, ώστε να αποτραπεί η διαίρεση με μηδέν. Με αυτόν

τον τρόπο εξασφαλίζεται η αριθμητική σταθερότητα και αποφεύγονται πιθανά σφάλματα κατά την εκτίμηση των αποστάσεων και του ορίου απόφασης.

Η διόρθωση αυτή επέτρεψε τη σωστή εκτέλεση του ταξινομητή MLTSVM και την ομαλή συνέχιση της πειραματικής διαδικασίας.

Ο αλγόριθμος MLTSVM εφαρμόστηκε αρχικά στο σύνολο δεδομένων CAL500, το οποίο περιλαμβάνει 68 χαρακτηριστικά και 174 διαφορετικές ετικέτες. Τα δεδομένα φορτώθηκαν από αρχείο τύπου .arff και διαχωρίστηκαν σε σύνολα εκπαίδευσης και δοκιμής σε αναλογία 70%-30%. Η υλοποίηση ακολούθησε δύο βασικές φάσεις.

Κατά την πρώτη φάση πραγματοποιήθηκε η βασική εκπαίδευση του αλγορίθμου. Το μοντέλο MLTSVM εκπαιδεύτηκε απευθείας στα αρχικά δεδομένα και αξιολογήθηκε μετρώντας το Hamming Loss και την Accuracy και καταγράφοντας το χρόνο εκτέλεσης.

```
# ===== Εκπαίδευση MLTSVM Ταξινομητή =====
clfMLTSVM = MLTSVM()
clfMLTSVM.fit(csr_matrix(X_train), csr_matrix(y_train))
y_pred = clfMLTSVM.predict(X_test)

# Αξιολόγηση Απόδοσης
print("Hamming Loss MLTSVM Before Embeddings:", hamming_loss(y_test, y_pred.astype(int)))
end_time = time.time()
print("Accuracy MLTSVM Before Embeddings:", accuracy_score(y_test, y_pred.astype(int)))
print("Process Time Before Embeddings:", end_time - start_time, "seconds")
```

Κατά τη δεύτερη φάση η οποία αφορά στην ενσωμάτωση χαρακτηριστικών και τη μείωση ετικετών με ομαδοποίηση, ακολουθήσαμε τα ίδια επιμέρους βήματα, όπως και στις προηγούμενες τρεις υλοποιήσεις. Πιο συγκεκριμένα:

- **Κατασκευή Γράφου Συνεμφάνισης Ετικετών**
  Υλοποιήθηκε γράφος (χρήση NetworkX), όπου οι κόμβοι αντιστοιχούν σε ετικέτες και οι ακμές βασίζονται στη συνεμφάνιση τους στο ίδιο δείγμα.
- **Ενσωμάτωση (Embeddings) με Node2Vec**
  Εφαρμόστηκε η μέθοδος Node2Vec για την παραγωγή αριθμητικών αναπαραστάσεων (διαστάσεων 128) των ετικετών, λαμβάνοντας υπόψη τη δομή του γράφου.
- **Ομαδοποίηση Ετικετών με K-Means**
  Οι παραγόμενες αναπαραστάσεις ομαδοποιήθηκαν σε νέο, μειωμένο σύνολο ετικετών (30% του αρχικού πλήθους). Η διαδικασία συγχώνευσε τις αρχικές ετικέτες σε clusters.
- **Μετασχηματισμός των Label Vectors:**
  Τα δεδομένα εξόδου επανακωδικοποιήθηκαν σύμφωνα με τις ομάδες των ετικετών (clusters).
- **Εμπλουτισμός των Εισόδων με Ενσωματωμένες Πληροφορίες:**
  Τα χαρακτηριστικά κάθε δείγματος εμπλουτίστηκαν με τον μέσο όρο των embeddings των ετικετών του, δημιουργώντας ένα συνδυασμένο διάνυσμα που περιλαμβάνει πληροφορία για τη συσχέτιση χαρακτηριστικών–ετικετών.

Τέλος, το μοντέλο εκπαιδεύτηκε εκ νέου με τα εμπλουτισμένα δεδομένα και με τις συγχωνευμένες ετικέτες.

```
# ===== Εκ νέου Εκπαίδευση MLTSVM Ταξινομητή =====
clfMLTSVM = MLTSVM()
clfMLTSVM.fit(csr_matrix(X_train_embedded), csr_matrix(y_train_new))
MLTSVM_predictions = clfMLTSVM.predict(csr_matrix(X_test_embedded))
```

Για την αξιολόγηση της απόδοσης, μετά τη δεύτερη φάση, υπολογίστηκαν ξανά τόσο οι δείκτες Hamming Loss και Accuracy όσο και ο χρόνος εκτέλεσης, ώστε να διαπιστωθεί η επίδραση της διαδικασίας embedding–clustering στην αποδοτικότητα του μοντέλου.

```
# ===== Αξιολόγηση Απόδοσης =====
print("Hamming Loss MLTSVM (με νέα labels):", hamming_loss(y_test_new, MLTSVM_predictions.astype(int)))
end_time = time.time()
print("Accuracy MLTSVM (με νέα labels):", accuracy_score(y_test_new, MLTSVM_predictions.astype(int)))
print("Process Time After Embeddings:", end_time - start_time, "seconds")
```

3.9 Ανάπτυξη Κώδικα για Random Forest

Η παρούσα ενότητα περιγράφει την αξιοποίηση του ταξινομητή Random Forest στο πλαίσιο προβλημάτων ταξινόμησης πολλαπλών ετικετών. Αν και ο συγκεκριμένος αλγόριθμος έχει σχεδιαστεί αρχικά για προβλήματα μονής ετικέτας (single-label), καθίσταται κατάλληλος για την παρούσα μελέτη μέσω της αξιοποίησης τεχνικών μετασχηματισμού του προβλήματος. Συγκεκριμένα, υλοποιήθηκαν και αξιολογήθηκαν τρεις διαφορετικές στρατηγικές:

- **Binary Relevance (BR),**
- **Classifier Chains (CC) και**
- **Label Powerset (LP),**

οι οποίες επιτρέπουν την αναγωγή του προβλήματος πολλαπλών ετικετών σε σύνολα δυαδικών προβλημάτων ή προβλημάτων πολλαπλών τάξεων, με κατάλληλο τρόπο για χρήση με το μοντέλο. Επιπρόσθετα, περιγράφεται η διαδικασία ενσωμάτωσης σημασιολογικής πληροφορίας μέσω των embeddings ετικετών, με στόχο τον εμπλουτισμό της αναπαράστασης των δειγμάτων εισόδου και τη βελτίωση της προβλεπτικής ακρίβειας πρόβλεψης.

3.9.1 Θεωρητικό Υπόβαθρο

Η ταξινόμηση πολλαπλών ετικετών αναφέρεται σε προβλήματα κατά τα οποία κάθε δείγμα δύναται να συσχετίζεται ταυτόχρονα με περισσότερες από μία ετικέτες, σε αντίθεση με τη συμβατική ταξινόμηση μονής ετικέτας (single-label). Οι περισσότερες κλασικές μέθοδοι μηχανικής μάθησης —όπως ο αλγόριθμος Random Forest— είναι σχεδιασμένες για προβλήματα μονής ετικέτας. Προκειμένου να καταστούν κατάλληλες για σενάρια με πολλαπλών ετικετών, εφαρμόζονται τεχνικές μετασχηματισμού του προβλήματος (problem transformation methods), οι οποίες μετατρέπουν το αρχικό πρόβλημα σε μορφή η οποία μπορεί να αντιμετωπιστεί από παραδοσιακούς ταξινομητές. Στην παρούσα εργασία, χρησιμοποιούνται τρεις ευρέως διαδεδομένες προσεγγίσεις:

- **Binary Relevance (BR):**
  Η μέθοδος BR αποσυνθέτει το πρόβλημα σε L ανεξάρτητα δυαδικά υποπροβλήματα, όπου L είναι το πλήθος των ετικετών. Για κάθε ετικέτα εκπαιδεύεται ένας ανεξάρτητος ταξινομητής, ο οποίος αποφασίζει για την παρουσία ή απουσία της ετικέτας σε κάθε δείγμα. Παρά την απλότητα και την υπολογιστική αποδοτικότητα της μεθόδου, ένα βασικό της μειονέκτημα είναι η παράβλεψη των συσχετίσεων μεταξύ ετικετών, γεγονός που μπορεί να περιορίσει την απόδοση σε δεδομένα με ισχυρές εξαρτήσεις.

- **Classifier Chains (CC):**
  Η τεχνική CC αποτελεί επέκταση της BR, ενσωματώνοντας την αλληλεξάρτηση μεταξύ ετικετών. Οι ταξινομητές διατάσσονται σε αλυσίδα, και κάθε ταξινομητής λαμβάνει, εκτός από τα αρχικά χαρακτηριστικά εισόδου, και τις προβλέψεις των προηγούμενων στην αλυσίδα ως πρόσθετες εισόδους. Με τον τρόπο αυτό, διατηρείται η δυνατότητα χρήσης απλών βάσεων ταξινόμησης, ενώ παράλληλα επιτρέπεται η μοντελοποίηση της συσχέτισης μεταξύ ετικετών. Η σειρά των ταξινομητών ενδέχεται να επηρεάσει το τελικό αποτέλεσμα.
- **Label Powerset (LP):**
  Η προσέγγιση LP αντιμετωπίζει κάθε μοναδικό συνδυασμό ετικετών ο οποίος εμφανίζεται στο σύνολο εκπαίδευσης ως μια νέα, σύνθετη ετικέτα, μετατρέποντας το πρόβλημα σε κλασικό πρόβλημα πολλαπλών-κλάσεων μονής-ετικέτας. Το κύριο πλεονέκτημα αυτής της μεθόδου είναι ότι λαμβάνει υπόψη πλήρως τις αλληλεξαρτήσεις μεταξύ των ετικετών. Ωστόσο, η εκθετική αύξηση του αριθμού των νέων κλάσεων σε περιπτώσεις με μεγάλο πλήθος ετικετών ή υψηλή ποικιλία συνδυασμών αποτελεί σημαντικό περιορισμό από πλευράς κλιμάκωσης και υπολογιστικού κόστους.

Πέραν των παραπάνω μετασχηματισμών, στο πλαίσιο της παρούσας εργασίας εφαρμόστηκαν και τεχνικές ενσωμάτωσης πληροφορίας ετικετών, με στόχο την περαιτέρω ενίσχυση της προβλεπτικής ικανότητας των μοντέλων. Συγκεκριμένα, κατασκευάστηκαν γράφοι συνεμφάνισης ετικετών (co-occurrence graphs), πάνω στους οποίους εφαρμόστηκαν αλγόριθμοι δημιουργίας εμβαδών αναπαραστάσεων (embeddings) όπως το Node2Vec, ενώ ακολούθησε ομαδοποίηση (clustering) με τη μέθοδο K-Means. Οι παραγόμενες αναπαραστάσεις ενσωματώθηκαν ως νέα χαρακτηριστικά στο σύστημα ταξινόμησης, προσδίδοντας στο μοντέλο πρόσθετη σημασιολογική πληροφορία σχετικά με τη δομή και τις συσχετίσεις των ετικετών. Αυτή η στρατηγική στοχεύει στη βελτίωση της γενίκευσης και της ακρίβειας των ταξινομητικών αποφάσεων, αξιοποιώντας κρυφές σχέσεις που δεν είναι εμφανείς από τα αρχικά δεδομένα.

3.9.2 Ανάπτυξη Λειτουργικότητας σε Python

Στην ενότητα αυτή, θα αναλύσουμε την υλοποίηση τριών στρατηγικών πολυετικετικής ταξινόμησης (Binary Relevance, Classifier Chains και Label Powerset) με χρήση του ταξινομητή Random Forest. Η υλοποίηση αυτή πραγματοποιήθηκε μέσω βιβλιοθηκών όπως scikit-learn, scikit-multilearn, networkx, node2vec, και scipy. Κάθε στρατηγική εφαρμόστηκε και αξιολογήθηκε σε δύο φάσεις, αρχικά χωρίς εμπλουτισμό των δεδομένων με embeddings, και στη συνέχεια με τη χρήση **Node2Vec embeddings** των ετικετών. Η ενσωμάτωση embeddings είχε στόχο τη μείωση της διάστασης εξόδου και την αναπαράσταση της συσχέτισης μεταξύ ετικετών, με στόχο την ενίσχυση της απόδοσης και τη μείωση της πολυπλοκότητας. Τα αποτελέσματα αξιολογήθηκαν μέσω Hamming Loss, Accuracy και χρόνου επεξεργασίας.

Η διαδικασία ξεκινά με τον κλασικό διαχωρισμό του dataset CAL500 σε σύνολα εκπαίδευσης και δοκιμών. Στη συνέχεια, για την κάθε μέθοδο, εκπαιδεύεται το αντίστοιχο μοντέλο με τις αρχικές ετικέτες και υπολογίζονται δείκτες απόδοσης όπως το **Hamming Loss** και η **Accuracy**.

3.9.2.1 Προσέγγιση Binary Relevance με Embeddings Ετικετών

Ο ταξινομητής Random Forest εφαρμόζεται ξεχωριστά για κάθε ετικέτα μέσω του `BinaryRelevance`. Στην πρώτη φάση χωρίς τις ενσωματώσεις, κάθε ετικέτα αντιμετωπίζεται ως ξεχωριστό δυαδικό

πρόβλημα. Χρησιμοποιείται ο wrapper MutiOutputClassifier για την μετατροπή του Random Forest σε ταξινομητή πολλαπλών ετικετών.

```
#Πριν τις ενσωματώσεις
start_time = time.time()
base_rf = RandomForestClassifier(n_estimators=100, random_state=42)#1/4/2025
clf_rf = MultiOutputClassifier(base_rf) ##1/4/2025
clf_rf.fit(X_train, y_train)

#PREDICTION
# Predict the labels for the test set
y_pred = clf_rf.predict(X_test)
```

Για την αξιολόγηση του ταξινομητή καταγράφονται οι τιμές των δεικτών Hamming Loss και Accuracy. Επίσης, καταγράφεται ο χρόνος επεξεργασίας.

```
# EVALUATE performance
print("Hamming Loss RF Before Embeddings:", hamming_loss(y_test, y_pred.astype(int)))
end_time = time.time()
print("Accuracy RF Before Embeddings:", accuracy_score(y_test, y_pred.astype(int)))
print("Process Time Before Embeddings:", end_time - start_time, "seconds")
```

Στη συνέχεια υλοποιείται η προσέγγιση με τις ενσωματώσεις χαρακτηριστικών και την ομαδοποίηση ετικετών. Κατασκευάζεται γράφος συνεμφάνισης ετικετών και αναπαρίσταται με ενσωματώσεις μέσω του Node2Vec. Για κάθε δείγμα, οι ενσωματώσεις των ενεργών ετικετών του υπολογίζονται και συνενώνονται με τα αρχικά χαρακτηριστικά του.

Η ταξινόμηση πραγματοποιείται στο εμπλουτισμένο σύνολο χαρακτηριστικών.

```
# ===== Εκ νέου Εκπαίδευση Ταξινομητών =====
base_rf = RandomForestClassifier(n_estimators=100, random_state=42)#1/4/2025
clf_rf = MultiOutputClassifier(base_rf) ##1/4/2025
clf_rf.fit(X_train_embedded, y_train_new)
rf_predictions = clf_rf.predict(X_test_embedded)

# ===== Νέα Αξιολόγηση Απόδοσης =====
print("Hamming Loss RF(με νέα labels):", hamming_loss(y_test_new, rf_predictions.astype(int)))
end_time = time.time()
print("Accuracy RF(με νέα labels):", accuracy_score(y_test_new, rf_predictions.astype(int)))
print("Process Time After Embeddings:", end_time - start_time, "seconds")
```

3.9.2.2 Προσέγγιση Classifier Chains με Embeddings

Στη δεύτερη προσέγγιση ο Random Forest χρησιμοποιείται σε συνδυασμό με τον ClassifierChain. Δημιουργείται μία αλυσίδα ταξινομητών RF, όπου κάθε πρόβλεψη μεταφέρεται στην επόμενη ως είσοδος.

```
# ===== Εκπαίδευση Classifier Chain πριν από Embeddings =====
start_time = time.time()
base_rf = RandomForestClassifier(n_estimators=100, random_state=42)
# Χρησιμοποιούμε ClassifierChain αντί για MultiOutputClassifier, με παραμετροποίηση random_state
clf_cc_before = ClassifierChain(base_rf, random_state=42)
clf_cc_before.fit(X_train, y_train)

# Προβλέψεις στο test set
y_pred = clf_cc_before.predict(X_test)

# Αξιολόγηση απόδοσης
print("Hamming Loss RF (Classifier Chain) Before Embeddings:", hamming_loss(y_test, y_pred.astype(int)))
print("Accuracy RF (Classifier Chain) Before Embeddings:", accuracy_score(y_test, y_pred.astype(int)))
end_time = time.time()
print("Process Time Before Embeddings:", end_time - start_time, "seconds")
```

Όπως και πριν, δημιουργούνται embeddings από τις ετικέτες και ενσωματώνονται στα χαρακτηριστικά εισόδου. Η αλυσίδα των ταξινομητών τροφοδοτείται με τα εμπλουτισμένα δεδομένα.

```
# ===== Εκπαίδευση Classifier Chain μετά τα Embeddings =====
base_rf_emb = RandomForestClassifier(n_estimators=100, random_state=42)
clf_cc_after = ClassifierChain(base_rf_emb, random_state=42)
clf_cc_after.fit(X_train_embedded, y_train_new)
rf_predictions = clf_cc_after.predict(X_test_embedded)

# ===== Αξιολόγηση Απόδοσης =====
print("Hamming Loss RF (Classifier Chain) με νέα labels:", hamming_loss(y_test_new, rf_predictions.astype(int)))
print("Accuracy RF (Classifier Chain) με νέα labels:", accuracy_score(y_test_new, rf_predictions.astype(int)))
end_time = time.time()
print("Process Time After Embeddings:", end_time - start_time, "seconds")
```

3.9.2.3 Γ. Προσέγγιση Label Powerset με Clustering Ετικετών

Χρησιμοποιείται η στρατηγική LabelPowerset όπου κάθε μοναδικός συνδυασμός ετικετών αντιμετωπίζεται ως ενιαία κλάση.

```
# ===== Εκπαίδευση Ταξινομητή (Label Powerset) πριν την ενσωμάτωση Embeddings =====
start_time = time.time()
base_rf = RandomForestClassifier(n_estimators=100, random_state=42)
clf_lp = LabelPowerset(base_rf)
clf_lp.fit(X_train, y_train)

# Πρόβλεψη στο test set
y_pred = clf_lp.predict(X_test)

# Αξιολόγηση Απόδοσης
print("Hamming Loss RF Before Embeddings (Label Powerset):", hamming_loss(y_test, y_pred.astype(int)))
print("Accuracy RF Before Embeddings (Label Powerset):", accuracy_score(y_test, y_pred.astype(int)))
end_time = time.time()
print("Process Time Before Embeddings:", end_time - start_time, "seconds")
```

Οι ετικέτες μετατρέπονται σε embeddings μέσω Node2Vec και ομαδοποιούνται με KMeans clustering.

Οι νέες ετικέτες ορίζονται ως ενεργοποιήσεις των clusters. Για κάθε δείγμα, υπολογίζεται ο μέσος όρος των ενσωματώσεων των αρχικών ετικετών και συνενώνεται με τα χαρακτηριστικά.

Η ταξινόμηση γίνεται με βάση τις νέες ετικέτες και τα εμπλουτισμένα χαρακτηριστικά.

```
# ===== Εκπαίδευση Ταξινομητή (Label Powerset) μετά την ενσωμάτωση Embeddings =====
base_rf_after = RandomForestClassifier(n_estimators=100, random_state=42)
clf_lp_after = LabelPowerset(base_rf_after)
clf_lp_after.fit(X_train_embedded, y_train_new)

rf_predictions = clf_lp_after.predict(X_test_embedded)

# ===== Αξιολόγηση Απόδοσης =====
print("Hamming Loss RF (με νέα labels) - Label Powerset:", hamming_loss(y_test_new, rf_predictions.astype(int)))
print("Accuracy RF (με νέα labels) - Label Powerset:", accuracy_score(y_test_new, rf_predictions.astype(int)))
end_time = time.time()
print("Process Time After Embeddings:", end_time - start_time, "seconds")
```

Οι τρεις προσεγγίσεις υλοποιήθηκαν με παρόμοιο δομικό μοτίβο, με διαφορές κυρίως στον τρόπο μετασχηματισμού των ετικετών και την κατασκευή των εμπλουτισμένων χαρακτηριστικών. Επίσης, καταγράφηκε ο χρόνος εκτέλεσης πριν και μετά την εφαρμογή ενσωματώσεων για να αποτιμηθεί το κόστος εμπλουτισμού. Η υλοποίηση ολοκληρώνεται με την οπτικοποίηση των γράφων πριν και μετά την συγχώνευση ετικετών, σε κάθε περίπτωση.

Κεφάλαιο 4ο: Πειραματική Μελέτη - Αποτελέσματα

Στο παρόν κεφάλαιο παρουσιάζονται η πειραματική διαδικασία και τα αποτελέσματα τα οποία προέκυψαν από την εφαρμογή των αλγορίθμων κατηγοριοποίησης που αναπτύχθηκαν στα προηγούμενα κεφάλαια. Η πειραματική μελέτη έχει ως στόχο την αξιολόγηση της απόδοσης των μεθόδων σε διάφορα σύνολα δεδομένων, με έμφαση στη σύγκριση των διαφορετικών προσεγγίσεων και τεχνικών που εφαρμόστηκαν. Επιπλέον, γίνεται ανάλυση των αποτελεσμάτων με σκοπό την εξαγωγή συμπερασμάτων σχετικά με τα πλεονεκτήματα, τους περιορισμούς και τις δυνατότητες βελτίωσης των αλγορίθμων. Η παρουσίαση των αποτελεσμάτων βασίζεται σε κατάλληλους δείκτες αξιολόγησης και συνοδεύεται από σχολιασμό και συζήτηση, ώστε να καταδειχθεί η πρακτική αξία των προτεινόμενων λύσεων.

4.1 Σύνολα Δεδομένων και Εγκαθίδρυση Πειραμάτων

Για τη συγκριτική αξιολόγηση των ταξινομητών πολλαπλών ετικετών, χρησιμοποιήθηκαν οκτώ διαφορετικά σύνολα δεδομένων τα οποία παρουσιάστηκαν αναλυτικά στην παράγραφο 3.3. Η επιλογή των συνόλων αυτών έγινε με γνώμονα την ποικιλομορφία ως προς τη φύση του προβλήματος, την ποικιλία των χαρακτηριστικών, τον αριθμό των ετικετών, καθώς και τη δομή ετικετοδότησης, ώστε να διασφαλιστεί η επάρκεια του πειραματικού πλαισίου.

4.1.1 Σύνολα Δεδομένων

Για κάθε σύνολο δεδομένων αξιοποιούνται δύο βασικές μετρικές που αφορούν στην ποσοτική ανάλυση των χαρακτηριστικών ετικετοδότησης, η πληθικότητα (*cardinality)* και η πυκνότητα (*density)*.

- **Πληθικότητα**
  Η πληθικότητα ενός χαρακτηριστικού (στήλης) σε ένα σύνολο δεδομένων είναι ο αριθμός των μοναδικών τιμών που εμφανίζονται σε αυτό το χαρακτηριστικό. Όταν λέμε ότι ένα χαρακτηριστικό διακρίνεται από χαμηλή πληθικότητα σημαίνει ότι το χαρακτηριστικό έχει λίγες μοναδικές τιμές (π.χ. φύλο: [άνδρας, γυναίκα]). Αντίθετα, υψηλή πληθικότητα σε ένα χαρακτηριστικό σημαίνει ότι έχει πολλές μοναδικές τιμές (π.χ. ΑΦΜ, email, usernames).

  Η πληθικότητα μάς βοηθά να κατανοήσουμε το εύρος της ποικιλίας τιμών σε μια στήλη. Είναι ιδιαίτερα σημαντική όταν καλούμαστε να αποφασίζουμε πώς να κάνουμε την κωδικοποίηση (encoding) κατηγορικών δεδομένων.

- **Πυκνότητα**
  Η πυκνότητα ενός συνόλου δεδομένων ή μιας στήλης περιγράφει το ποσοστό των μη-κενών (μη-μηδενικών ή μη-null) τιμών σε σχέση με το συνολικό πλήθος των τιμών. Μπορεί να υπολογιστεί είτε ως :

$$\text{Density} = \frac{\text{Αριθμός μη-κενών τιμών}}{\text{Συνολικός αριθμός τιμών}}$$

  Είτε για sparse matrices ως:

$$\text{Density} = \frac{\text{Αριθμός μη-μηδενικών στοιχείων}}{\text{Συνολικός αριθμός στοιχείων στον πίνακα}}$$

Η πυκνότητα μάς δείχνει κατά πόσο ένα σύνολο δεδομένων χαρακτηρίζεται ως αραιό (sparse) ή πυκνό (dense). Είναι σημαντική για την αποδοτικότητα της αποθήκευσης και της επεξεργασίας, ειδικά σε εφαρμογές μηχανικής μάθησης.

Οι δύο αυτοί δείκτες προσφέρουν σημαντική πληροφορία, επηρεάζοντας άμεσα τη συμπεριφορά των ταξινομητικών μοντέλων.

Ο Πίνακας 4.1 παρουσιάζει αναλυτικά τις τιμές της πληθικότητας και πυκνότητας για κάθε ένα από τα εξεταζόμενα σύνολα δεδομένων, υπολογισμένες βάσει των αρχικών, μη τροποποιημένων ετικετοδοτήσεων. Οι τιμές αυτές προέρχονται από τις αντίστοιχες δημοσιεύσεις των συνόλων δεδομένων ή έχουν υπολογιστεί απευθείας από τα δεδομένα μέσω της χρήσης της βιβλιοθήκης scikit-multilearn.

Πίνακας 3: Μετρικές Πληθικότητας και Πυκνότητας για τα σύνολα δεδομένων

Dataset	Εγγραφές	Ετικέτες	Cardinality	Density	Σχόλια
CAL500	502	174	26,04	0,1496	Πολύ υψηλή πληθικότητα – κάθε τραγούδι έχει πολλές ετικέτες (μουσικά χαρακτηριστικά). Η πυκνότητα είναι μέτρια, άρα δεν είναι ακραία sparse.
Birds	645	19	1,014	0,0534	Πολύ χαμηλή cardinality – κάθε δείγμα σχετίζεται με περίπου 1 ετικέτα. Η πολύ χαμηλή density δείχνει ότι ο πίνακας είναι αραιός.
Water Quality	1060	14	5,00	0,3571	Μέτρια cardinality και σχετικά υψηλή density. Οι ετικέτες είναι πιο πυκνές – κάθε δείγμα έχει αρκετές σχετικές ετικέτες.
Yeast	2417	14	4,24	0,3029	Παρόμοιο προφίλ με το προηγούμενο. Μέτρια cardinality και density.
Mediamill	43907	101	4,38	0,0434	Παρά τον μεγάλο αριθμό παρατηρήσεων, η density είναι πολύ χαμηλή → εξαιρετικά sparse dataset.
Scene	2407	6	1,07	0,1783	Πολύ χαμηλή cardinality – σχεδόν μονήρης ετικέτα ανά δείγμα. Πυκνότητα πιο υψηλή από Birds/Mediamill, αλλά ακόμα σχετικά αραιή.
Emotions	593	6	1,87	0,3117	Χαμηλή cardinality, αλλά υψηλή density. Αυτό σημαίνει ότι, παρότι έχει λίγες ετικέτες, σχεδόν κάθε δείγμα σχετίζεται με αρκετές.

Dataset	Εγγραφές	Ετικέτες	Cardinality	Density	Σχόλια
Flags	194	7	3.39	0,4843	Υψηλή density – σχεδόν οι μισές πιθανές ετικέτες ανά δείγμα είναι ενεργές. Είναι το πιο "πυκνό" dataset στη λίστα.

Παρατηρώντας τον παραπάνω πίνακα μπορούμε να συμπεράνουμε ότι:

- Το CAL500 ξεχωρίζει με πολύ υψηλή πληθικότητα, καθώς κάθε δείγμα σχετίζεται με πολλές ετικέτες – πρόβλημα πολλαπλών ετικετών με μεγάλη πυκνότητα πληροφοριών ανά δείγμα.
- Τα Mediamill και Birds έχουν χαμηλή πυκνότητα, συνεπώς θεωρούνται πολύ αραιά σύνολα δεδομένων – χρήσιμο για δοκιμές σε αλγορίθμους που διαχειρίζονται καλά τα αραιά δεδομένα.
- Τα Flags και Emotions έχουν υψηλή πυκνότητα παρά τις λίγες ετικέτες – γεγονός που υποδεικνύει ότι σχεδόν όλες οι ετικέτες είναι ενεργές σε κάθε παρατήρηση.
- Τα Water Quality και Yeast παρουσιάζουν ισορροπία μεταξύ πληθικότητας και πυκνότητας – ούτε πολύ αραιά, ούτε πολύ πυκνά.

4.1.2 Διαδικασία Πειραμάτων

Η διαδικασία πειραματικής αξιολόγησης οργανώθηκε σε δύο βασικές φάσεις:

1. **Αρχική Εκπαίδευση**: Οι βασικοί ταξινομητές για προβλήματα πολλαπλών ετικετών εφαρμόστηκαν στα αρχικά δεδομένα χωρίς καμία ενσωμάτωση ή συμπίεση. Οι αλγόριθμοι αξιολογήθηκαν βάσει τριών δεικτών απόδοσης: Hamming Loss, Accuracy και CPU Time.
2. **Εκπαίδευση με Ενσωματώσεις (Embeddings)**: Σε δεύτερο στάδιο, εφαρμόστηκε τεχνική ενσωμάτωσης ετικετών (Node2Vec) με στόχο τη μείωση του πλήθους των ετικετών ή/και τη συσχέτιση μεταξύ τους. Η μείωση έγινε με βάση τις αρχές που παρουσιάστηκαν στο Κεφάλαιο 3.3, με στόχο τη διατήρηση ενός ισορροπημένου αριθμού ετικετών ανά σύνολο, χωρίς να διαταράσσεται η εσωτερική δομή των δεδομένων.

Οι ταξινομητές υλοποιήθηκαν μέσω των βιβλιοθηκών `scikit-multilearn` και `sklearn`, ενώ για τη διαχείριση των ενσωματώσεων χρησιμοποιήθηκε η βιβλιοθήκη `Node2Vec`.

4.2 Πειραματικά αποτελέσματα

Σε αυτήν την ενότητα παρουσιάζουμε και αναλύουμε τα αποτελέσματα των πειραμάτων με βάση τις τρεις βασικές μετρικές αξιολόγησης: Hamming Loss, Accuracy και CPU Time. Τα πειραματικά αποτελέσματα χωρίζονται σε δύο κατηγορίες, ταξινόμηση πριν την ενσωμάτωση χαρακτηριστικών και ταξινόμηση μετά την ενσωμάτωση χαρακτηριστικών.

Οι ταξινομητές που συγκρίνονται είναι οι BRkNNa, MLkNN, RF, MLARAM, και MLTSVM.

4.2.1 Αποτελέσματα πριν την ενσωμάτωση

4.2.1.1 Τιμή του k για το καλύτερο Hamming Loss

Σε όλα τα σύνολα δεδομένων, τόσο για τον BRkNN όσο και για τον MLkNN, παρατηρήθηκε ότι το Hamming Loss είναι καλύτερο (μικρότερη τιμή) όταν χρησιμοποιούμε τιμή για το k ίση με το πλήθος των ετικετών και όχι όταν k = 1.

Πίνακας 4: Hamming Loss για τιμές του k στους BRkNN και MLkNN

Dataset	**BRkNN (k=1)**	**BRkNN (k=#labels)**	**MLkNN (k=1)**	**MLkNN (k=#labels)**
CAL500	0,1978	**0,1366**	0,1978	**0,1397**
BIRDS	0,0867	**0,0520**	0,0867	**0,0536**
WATER QUALITY	0,3376	**0,2826**	0,3376	**0,2916**
YEAST	0,2421	**0,1948**	0,2421	**0,2019**
MEDIAMILL	0,0328	**0,0311**	0,0328	**0,0306**
SCENE	0,1077	**0,0947**	0,1077	**0,0920**
EMOTIONS	0,2949	**0,2762**	0,2949	**0,2800**
FLAGS	0,3632	**0,3148**	0,3632	**0,3172**

4.2.1.2 Βέλτιστος αλγόριθμος ταξινόμησης ανά dataset

Προκειμένου να απαντήσουμε στο ερώτημα, ποιος αλγόριθμος ταξινόμησης είναι πιο αποτελεσματικός για κάθε dataset, θα λάβουμε υπόψη κυρίως τον δείκτη Hamming Loss (όσο μικρότερο, τόσο καλύτερα) και επικουρικά τον δείκτη Accuracy. Στον πίνακα που ακολουθεί καταγράφεται η βέλτιστη τιμή hamming loss ο οποίος επιτυγχάνεται σε κάθε dataset και ποιος αλγόριθμος τον κατάφερε.

Πίνακας 5: Βέλτιστος αλγόριθμος ανά dataset

Dataset	**Καλύτερος Αλγόριθμος**	**Hamming Loss**	**Accuracy**	**Παρατηρήσεις**
CAL500	BRkNN	0,1366	0,0000	Όλοι 0% Accuracy, αλλά BRkNN με το μικρότερο Hamming Loss
BIRDS	RF-LP	0,0398	0,5418	Καλύτερο Hamming Loss & Accuracy
WATER QUALITY	RF-CC	0,2673	0,0377	Καλύτερο Hamming Loss
YEAST	RF-LP	0,2007	0,2631	Υψηλότερη Accuracy
MEDIAMILL	RF-BR	0,0259	0,1628	Πολύ καλό Hamming Loss & αξιοπρεπής Accuracy
SCENE	RF-LP	0,0710	0,7621	Κορυφαίο Hamming Loss & Accuracy
EMOTIONS	RF-LP	0,1910	0,3933	Καλύτερος σε όλα
FLAGS	RF-CC	0,2421	0,2542	Βέλτιστη απόδοση σε όλα

Σύμφωνα με τον συγκριτικό πίνακα που ακολουθεί, ο πιο αποτελεσματικός αλγόριθμος στο σύνολο των datasets είναι ο Random Forest με χρήση Label Powerset, καθώς παρουσιάζει την καλύτερη τιμή στον δείκτη hamming loss στα περισσότερα datasets.

Πίνακας 6: Συνολική απόδοση αλγορίθμων

Αλγόριθμος	Καλύτερος σε πόσα Datasets
RF-LP	4 (BIRDS, YEAST, SCENE, EMOTIONS)
RF-CC	2 (WATER QUALITY, FLAGS)
RF-BR	1 (MEDIAMILL)
BRkNN	1 (CAL500)

Μάλιστα ο Random Forest φαίνεται να υπερτερεί μεταξύ των υπόλοιπων τεσσάρων ταξινομητών καθώς καταφέρνει να έχει τις καλύτερες αποδόσεις στα επτά από τα οκτώ datasets.

4.2.1.3 Βέλτιστη χρήση του Random Forest

Σε μία προσπάθεια ερμηνείας της αποτελεσματικότητας του RF θα λέγαμε ότι εξαρτάται σημαντικά από τη στρατηγική που χρησιμοποιείται για την αντιμετώπιση του προβλήματος των πολλαπλών ετικετών. Από την ανάλυση των αποτελεσμάτων στα οκτώ datasets, προκύπτουν τα εξής:

- Hamming Loss :
  - RF-LP υπερέχει σε 4 από τα 8 datasets (BIRDS, YEAST, SCENE, EMOTIONS), επιτυγχάνοντας τη χαμηλότερη τιμή Hamming Loss.
  - RF-CC επιτυγχάνει την καλύτερη τιμή σε 2 datasets (WATER QUALITY, FLAGS).
  - RF-BR ξεχωρίζει μόνο σε 1 dataset (MEDIAMILL).
- Accuracy :
  - RF-LP καταγράφει την υψηλότερη ακρίβεια σε 4 datasets (BIRDS, YEAST, SCENE, EMOTIONS).
  - RF-CC έχει τη βέλτιστη απόδοση σε 2 datasets (WATER QUALITY, FLAGS).
  - RF-BR υπερέχει μόνο στο MEDIAMILL, όπου όμως το accuracy είναι 0 σε όλα τα μοντέλα, οπότε δεν θεωρείται ουσιαστικά αξιοσημείωτο.

Από τη σύγκριση συμπεραίνουμε ότι η στρατηγική Label Powerset (LP) είναι, κατά κανόνα, η πιο αποδοτική επιλογή για τον ταξινομητή Random Forest σε προβλήματα ταξινόμησης πολλαπλών ετικετών, προσφέροντας τον βέλτιστο συνδυασμό μεταξύ ακρίβειας, χαμηλού σφάλματος και αποδοτικότητας. Ωστόσο, σε συγκεκριμένες περιπτώσεις με ιδιαίτερη ετερογένεια ετικετών, όπως το σύνολο δεδομένων FLAGS, η στρατηγική Classifier Chains (CC) υπερτερεί ελαφρώς.

4.2.2 Αποτελέσματα μετά την ενσωμάτωση

4.2.2.1 Βέλτιστη χρήση του node2Vec

Η ενσωμάτωση αναπαραστάσεων από τον node2Vec στους ταξινομητές πολλαπλών ετικετών έχει ως σκοπό τη βελτίωση της απόδοσής τους μέσω της ενίσχυσης της πληροφορίας που περιγράφει τις συσχετίσεις μεταξύ των ετικετών. Για το σκοπό αυτό, δοκιμάστηκαν τρεις διαφορετικές ρυθμίσεις παραμέτρων του node2Vec, μεταβάλλοντας τη διάσταση του χώρου των ενσωματώσεων (64 ή 128),

τον αριθμό βημάτων περιήγησης (walk-through = 10 ή 20), καθώς και τον αριθμό εργαζομένων (workers = 1, 2 ή 4) κατά την παραγωγή των ενσωματώσεων.

Αναλύοντας τα αποτελέσματα όλων των ταξινομητών στο σύνολο των datasets, παρατηρούμε ότι η τρίτη παραμετροποίηση — δηλαδή αυτή με διαστάσεις 128, πλήθος βημάτων περιήγησης 20 και 4 workers — προσφέρει τις καλύτερες αποδόσεις στο σύνολο σχεδόν των περιπτώσεων. Χαρακτηριστικά παραδείγματα όπου η συγκεκριμένη ρύθμιση υπερτερεί περιλαμβάνουν τους ταξινομητές :

- MLARAM στο FLAGS όπου έχει τιμή Hamming Loss: 0,175, ενώ οι πρώτες δύο παραμετροποιήσεις έχουν τιμές 0,718 και 0,718 αντίστοιχα.
- RF στο CAL500, όπου έχει τιμή Hamming Loss: 0,058, ενώ οι πρώτες δύο παραμετροποιήσεις έχουν τιμές 0,131 και 0,124 αντίστοιχα, και
- MLkNN στο MEDIAMILL, όπου έχει τιμή Hamming Loss: 0,009, ενώ οι πρώτες δύο παραμετροποιήσεις έχουν τιμές 0,019 και 0,025 αντίστοιχα

Η επιλογή αυτής της παραμετροποίησης φαίνεται να προσφέρει έναν ικανοποιητικό συμβιβασμό ανάμεσα στην ποιότητα της αναπαράστασης (λόγω των υψηλότερων διαστάσεων) και την πληρότητα της εξερεύνησης της δομής του γράφου (μέσω του αυξημένου walk-through), ενώ η χρήση περισσότερων εργαζομένων (workers=4) επιταχύνει τη διαδικασία χωρίς να υποβαθμίζει τα αποτελέσματα.

Συνεπώς, για τις ανάγκες της παρούσας μελέτης, η τρίτη ρύθμιση του node2Vec αξιολογείται ως η πλέον ενδεδειγμένη για την ενίσχυση των πολυετικετικών ταξινομητών.

Ο πίνακας που ακολουθεί συνοψίζει τις μέσες αποδόσεις των ταξινομητών για τις τρεις παραμετροποιήσεις του Node2Vec, όπως καταγράφηκαν στα αποτελέσματα για όλων των datasets.

Πίνακας 7: Συνολική απόδοση Ταξινομητών με Node2Vec

Ταξινομητής	**Παραμετροποίηση 1 (dim=64, walk=10, workers=1)**	**Παραμετροποίηση 2 (dim=64, walk=10, workers=2)**	**Παραμετροποίηση 3 (dim=128, walk=20, workers=4)**
BRkNN	0,206	0,174	0,174
MLkNN	0,169	0,164	0,161
RF	0,023	0,022	0,013
MLARAM	0,289	0,266	0,195
MLTSVM	0,593	0,578	0,617

Παρατηρούμε ότι για τους περισσότερους ταξινομητές (BRkNN, MLkNN, RF, MLARAM), η τρίτη παραμετροποίηση (dim=128, walk=20, workers=4) υπερτερεί των δύο πρώτων παραμετροποιήσεων (dim=64, walk=10, workers=1 ή 2), εκτός από την περίπτωση του ταξινομητή MLTSVM, όπου σημειώνεται σημαντική βελτίωση με τη χρήση της δεύτερης παραμετροποίησης.

4.2.2.2 Βέλτιστος αλγόριθμος ταξινόμησης ανά dataset

Η αξιολόγηση των αλγορίθμων ταξινόμησης μετά την ενσωμάτωση χαρακτηριστικών μέσω node2Vec επιτρέπει την εξαγωγή χρήσιμων συμπερασμάτων σχετικά με την αποτελεσματικότητα κάθε μεθόδου σε διαφορετικά σύνολα δεδομένων. Στους πίνακες που προηγήθηκαν, παρατηρούμε ότι η απόδοση των ταξινομητών διαφέρει σημαντικά ανάλογα με τη φύση των δεδομένων, την πυκνότητα των ετικετών, τον αριθμό χαρακτηριστικών, καθώς και τις παραμέτρους του node2Vec.

Στον παρακάτω πίνακα συνοψίζεται ο βέλτιστος ταξινομητής ανά dataset, με βάση την ελάχιστη τιμήτου δείκτη Hamming Loss και τη μέγιστη επίδοση του δείκτη Accuracy η οποία καταγράφηκε στην τρίτη παραμετροποίηση του node2Vec:

Πίνακας 8: Συνολική απόδοση αλγορίθμων

Dataset	**MIN HL**	**MAX ACC**
CAL500	RF-BR: 0,054	RF-BR: 0,040
BIRDS	RF-BR: 0,014	RF-BR: 0,901
WATER QUALITY	RF-BR: 0,024	RF-BR: 0,802
YEAST	RF-BR: 0,002	RF-BR: 0,989
MEDIAMILL	RF-BR: 0,001	RF-BR: 0,970
SCENE	RF (BR,CC,LP): 0,000	RF (BR,CC,LP): 1,000
EMOTIONS	RF (BR,CC,LP): 0,000	RF (BR,CC,LP): 1,000
FLAGS	RF-LP: 0,000	RF-LP: 1,000

Τα συμπεράσματα που αντλούνται από την παραπάνω ανάλυση είναι:

- Ο RF-BR αναδείχθηκε ο πιο αποδοτικός ταξινομητής στα 7 από τα 8 datasets, επιτυγχάνοντας τις υψηλότερες επιδόσεις τόσο σε Hamming Loss όσο και σε Accuracy.
- Ο RF-LP υπερίσχυσε στην περίπτωση του συνόλου FLAGS.
- Οι υπόλοιποι ταξινομητές (BRkNN, MLkNN, MLARAM, MLTSVM) δεν κατέλαβαν την πρώτη θέση σε κανένα dataset, παρά την αξιοσημείωτη σταθερότητά τους σε ορισμένες περιπτώσεις.

Τα παραπάνω ευρήματα καταδεικνύουν ότι ο RF-BR, σε συνδυασμό με πλούσιες ενσωματώσεις από τον node2Vec, είναι ο πιο αποδοτικός ταξινομητής στο μεγαλύτερο πλήθος datasets και αποτελεί εξαιρετική επιλογή για ταξινόμηση πολλαπλών ετικετών όταν χρησιμοποιούνται ενσωματωμένες αναπαραστάσεις.

4.2.3 Παρατηρήσεις για την εκτέλεση του MLTSVM στο Mediamill

Ο αλγόριθμος MLTSVM (Multi-Label Twin Support Vector Machine) αποτελεί μια επέκταση των δίδυμων SVMs στην πολυετικετική μάθηση, βασιζόμενος στην ανεξάρτητη εκπαίδευση ενός ζεύγους υπερπλάνων ανά ετικέτα (hyperplanes per label). Η διαδικασία αυτή περιλαμβάνει την επίλυση συστημάτων γραμμικών εξισώσεων μεγάλου μεγέθους, καθώς και την επαναλαμβανόμενη κατασκευή πινάκων Gram ή παρόμοιων γραμμικών αναπαραστάσεων για κάθε ετικέτα ξεχωριστά.

Σε προβλήματα με αυξημένο πλήθος δειγμάτων και ετικετών, η υπολογιστική πολυπλοκότητα του MLTSVM αυξάνεται σχεδόν κλιμακωτά, καθιστώντας τον ιδιαίτερα επιβαρυντικό τόσο από άποψη μνήμης όσο και από άποψη χρόνου εκτέλεσης. Η πολυπλοκότητα αυτή εκδηλώνεται κυρίως:

- με υψηλές απαιτήσεις μνήμης, λόγω των ενδιάμεσων πινάκων που αποθηκεύονται προσωρινά κατά τη διάρκεια της εκπαίδευσης, και
- με σημαντική αύξηση του χρόνου υπολογισμού, καθώς κάθε επιπλέον ετικέτα συνεπάγεται και μια νέα βαρύτατη υπολογιστική διαδικασία.

Το σύνολο δεδομένων Mediamill συγκαταλέγεται ανάμεσα στα μεγαλύτερα και πιο απαιτητικά datasets που χρησιμοποιήθηκαν στο πλαίσιο της παρούσας μελέτης. Χαρακτηρίζεται από:

- περίπου 43.000 δείγματα (instances),
- 120 διαστάσεις χαρακτηριστικών, και
- 101 ετικέτες.

Το μέγεθος και η πολυπλοκότητα αυτού του dataset καθιστούν εξαιρετικά δύσκολη την εφαρμογή αλγορίθμων που βασίζονται σε kernel-based μηχανισμούς όπως ο MLTSVM. Δεδομένου ότι απαιτείται η ξεχωριστή εκπαίδευση για κάθε μία από τις 101 ετικέτες, σε συνδυασμό με τον όγκο των παραδειγμάτων, η συνολική υπολογιστική απαίτηση καθίσταται μη διαχειρίσιμη χωρίς σημαντική προεπεξεργασία ή επιτάχυνση.

Ο MLTSVM, στη βασική του μορφή, δεν είναι σχεδιασμένος να κλιμακώνεται αποδοτικά σε datasets τέτοιου μεγέθους χωρίς την εφαρμογή τεχνικών βελτιστοποίησης και αποδοτικής παραμετροποίησης. Ειδικότερα, σε αυτή τη μελέτη δεν κατέστη εφικτή η ολοκλήρωση της εκπαίδευσης του MLTSVM στο Mediamill λόγω:

- απουσίας ελαφριών πυρήνων (π.χ. γραμμικού kernel), οι οποίοι θα μπορούσαν να μειώσουν την υπολογιστική επιβάρυνση,
- έλλειψης κανονικοποίησης και μείωσης διαστάσεων (όπως μέσω PCA ή επιλογής χαρακτηριστικών),
- αδυναμίας εκμετάλλευσης αραιών αναπαραστάσεων (sparsity-aware υποδομών), και
- περιορισμένων πόρων υπολογιστικού συστήματος (RAM, CPU).

Ως αποτέλεσμα, κάθε προσπάθεια εκπαίδευσης του MLTSVM στο συγκεκριμένο dataset είτε οδηγούσε σε κατάρρευση λόγω εξάντλησης της μνήμης είτε σε υπερβολικά μεγάλο χρόνο εκτέλεσης χωρίς ολοκλήρωση της διαδικασίας σε αποδεκτό χρονικό πλαίσιο.

Για τους παραπάνω λόγους, η εκπαίδευση του MLTSVM στο σύνολο δεδομένων Mediamill δεν εκτελέστηκε, προκειμένου να διατηρηθεί η συνέπεια και η πρακτικότητα της πειραματικής διαδικασίας, σε αντιστοιχία με τα υπόλοιπα μοντέλα και datasets της μελέτης.

4.3 Συζήτηση

Η παρούσα πειραματική μελέτη ανέδειξε σημαντικές διαφοροποιήσεις στην απόδοση των εξεταζόμενων αλγορίθμων μάθησης σε προβλήματα πολλαπλών ετικετών, τόσο πριν όσο και μετά την ενσωμάτωση αναπαραστάσεων κόμβων μέσω node2Vec. Η αξιολόγηση πραγματοποιήθηκε με χρήση δεικτών όπως Accuracy, Hamming Loss, και του χρόνου υπολογιστικής εκτέλεσης (CPU Time), προσφέροντας μια ολοκληρωμένη εικόνα της αποδοτικότητας και πρακτικής χρησιμότητας κάθε μεθόδου.

Πριν την εφαρμογή ενσωματώσεων, ιδιαίτερο ενδιαφέρον παρουσίασε ο ταξινομητής Random Forest, ο οποίος, ανάλογα με τη στρατηγική πολυετικετικής μοντελοποίησης που χρησιμοποιήθηκε (Binary Relevance, Classifier Chains ή Label Powerset), παρουσίασε διαφοροποιημένες επιδόσεις. Συγκεκριμένα, η στρατηγική Label Powerset (RF-LP) υπερίσχυσε στα περισσότερα σύνολα δεδομένων (4 από τα 8), επιτυγχάνοντας τη χαμηλότερη τιμή Hamming Loss και τη μεγαλύτερη ακρίβεια, γεγονός που καταδεικνύει τη δυνατότητά της να εκμεταλλεύεται αποτελεσματικά τις συσχετίσεις μεταξύ των ετικετών. Αντίστοιχα, η στρατηγική Classifier Chains (RF-CC) υπερείχε σε σύνολα δεδομένων με ετερογενείς ή πολυδιάστατες ετικέτες, όπως τα WATER QUALITY και FLAGS, υποδεικνύοντας ότι σε τέτοιες περιπτώσεις η μοντελοποίηση εξαρτήσεων μεταξύ ετικετών αποδίδει καλύτερα. Τέλος, η προσέγγιση RF-BR απέδωσε ικανοποιητικά στο σύνολο MEDIAMILL, αν και η μέτρηση Accuracy δεν ήταν αξιοσημείωτη.

Ο ταξινομητής BRkNN, αν και δεν επικράτησε συνολικά, κατέγραψε την καλύτερη απόδοση στο σύνολο CAL500, όπου οι υπόλοιποι ταξινομητές παρουσίασαν ιδιαίτερα υψηλό Hamming Loss. Ενδιαφέρον αποτέλεσε η παρατήρηση ότι τόσο για τον BRkNN όσο και για τον MLkNN, η βέλτιστη τιμή του παραμέτρου k ως προς το Hamming Loss επιτυγχάνεται όταν το k ισούται με το πλήθος των ετικετών, και όχι με την ελάχιστη τιμή k=1. Η συμπεριφορά αυτή υποδηλώνει τη σημασία της πλήρους αξιοποίησης της πληροφορίας πολυετικετικής συσχέτισης ακόμη και σε απλούς βασικούς ταξινομητές βασισμένους σε k-πλησιέστερους γείτονες.

Μετά την ενσωμάτωση αναπαραστάσεων κόμβων μέσω node2Vec, παρατηρήθηκε σαφής βελτίωση των επιδόσεων των περισσότερων ταξινομητών, ιδίως όταν χρησιμοποιήθηκε η τρίτη παραμετροποίηση (διάσταση = 128, walk-through = 20, workers = 4). Η συγκεκριμένη ρύθμιση προσέφερε τον καλύτερο συμβιβασμό μεταξύ ποιότητας ενσωμάτωσης και πληρότητας εξερεύνησης της τοπολογίας του γράφου, ενώ η αξιοποίηση περισσότερων επεξεργαστών επιτάχυνε σημαντικά τη διαδικασία χωρίς να μειώσει την ακρίβεια.

Στο πλαίσιο αυτό, αναδείχθηκε ξεκάθαρα ότι ο ταξινομητής Random Forest με στρατηγική Binary Relevance (RF-BR) υπερείχε στα 7 από τα 8 datasets, επιτυγχάνοντας τις χαμηλότερες τιμές Hamming Loss και τις υψηλότερες τιμές Accuracy. Μοναδική εξαίρεση αποτέλεσε το σύνολο FLAGS, όπου ο RF-LP κατέγραψε την καλύτερη απόδοση.

Τα αποτελέσματα αυτά αποδεικνύουν τη συνέργεια μεταξύ της μεθόδου node2Vec και του RF-BR, καθώς η ενσωμάτωση των χαρακτηριστικών ενίσχυσε τη διακριτική ικανότητα του μοντέλου, ιδιαίτερα σε περιβάλλοντα με έντονη πολυπλοκότητα ως προς τις ετικέτες και τα χαρακτηριστικά. Είναι αξιοσημείωτο ότι ταξινομητές όπως οι MLkNN, MLARAM, MLTSVM και BRkNN, αν και παρουσίασαν σταθερή συμπεριφορά, δεν κατέλαβαν την πρώτη θέση σε κανένα από τα εξεταζόμενα σύνολα. Αυτό υποδηλώνει ότι η απόδοση αυτών των μοντέλων δεν ενισχύθηκε επαρκώς από την ενσωμάτωση κόμβων, τουλάχιστον σε σύγκριση με την προσαρμοστικότητα του Random Forest.

Ειδική αναφορά αξίζει να γίνει στις περιπτώσεις των συνόλων SCENE και EMOTIONS, όπου όλες οι στρατηγικές RF (BR, CC, LP) κατέγραψαν μηδενικό Hamming Loss και ακρίβεια 100%, αποκαλύπτοντας είτε ακραία καταλληλότητα των χαρακτηριστικών είτε ισχυρές δομικές συσχετίσεις που μοντελοποιούνται αποτελεσματικά μέσω των ενσωματώσεων node2Vec.

Τα συνολικά ευρήματα ενισχύουν την υπόθεση ότι:

- Η ενσωμάτωση χαρακτηριστικών μέσω node2Vec αποτελεί κρίσιμο παράγοντα βελτίωσης στην πολυετικετική ταξινόμηση.
- Ο Random Forest με στρατηγική Binary Relevance είναι ο πλέον αξιόπιστος και αποδοτικός ταξινομητής, όταν συνδυάζεται με κατάλληλη ενσωμάτωση.

- Η απόδοση των ταξινομητών εξαρτάται σε μεγάλο βαθμό από τις ιδιότητες των δεδομένων, όπως η πυκνότητα ετικετών και η διασπορά των χαρακτηριστικών.

Συμπερασματικά, η παρούσα μελέτη καταδεικνύει ότι ο συνδυασμός ισχυρών ενσωματώσεων node2Vec και προσαρμοστικών αλγορίθμων όπως ο RF-BR προσφέρει εξαιρετικά αποτελέσματα σε ευρύ φάσμα πολυετικετικών προβλημάτων, παρέχοντας ένα ισχυρό εργαλείο για εφαρμογές μηχανικής μάθησης σε περιβάλλοντα με υψηλό βαθμό πολυπλοκότητας.

Κεφάλαιο 5ο: Συμπεράσματα και Μελλοντική Έρευνα

5.1 Συμπεράσματα

Η παρούσα πτυχιακή εργασία είχε ως βασικό στόχο τη μελέτη, υλοποίηση και συγκριτική αξιολόγηση διαφόρων αλγορίθμων ταξινόμησης πολλαπλών ετικετών (multi-label classification), εστιάζοντας τόσο σε παραδοσιακές μεθόδους όσο και σε πιο πρόσφατες προσεγγίσεις που ενσωματώνουν τεχνικές αναπαράστασης (embeddings). Η εργασία κάλυψε το πλήρες φάσμα της διαδικασίας: από τη μελέτη του θεωρητικού υπόβαθρου και την υλοποίηση των αλγορίθμων σε Python, έως την εκτενή πειραματική αξιολόγηση σε πληθώρα συνόλων δεδομένων με διαφορετικά χαρακτηριστικά.

Αρχικά, παρουσιάστηκαν και υλοποιήθηκαν τρεις βασικές στρατηγικές μετασχηματισμού πολυετικετικών προβλημάτων: Binary Relevance (BR), Classifier Chains (CC) και Label Powerset (LP), εφαρμόζοντας ως βασικό ταξινομητή το Random Forest. Παράλληλα, εξετάστηκαν και υλοποιήθηκαν τέσσερις αλγόριθμοι πολυετικετικής μάθησης: BRkNN, MLkNN, MLARAM και MLTSVM, προσφέροντας μια ευρεία βάση σύγκρισης διαφορετικών προσεγγίσεων.

Η πειραματική μελέτη πραγματοποιήθηκε σε οκτώ πραγματικά σύνολα δεδομένων τα οποία παρουσιάζουν ποικιλία ως προς τον αριθμό των χαρακτηριστικών, των δειγμάτων και των ετικετών, επιτρέποντας την εξαγωγή γενικεύσιμων συμπερασμάτων. Οι δείκτες που χρησιμοποιήθηκαν για την αξιολόγηση των μετρήσεων περιλάμβαναν την Accuracy, τον Hamming Loss και τον χρόνο υπολογισμού, επιτρέποντας τη σφαιρική εκτίμηση της απόδοσης κάθε αλγορίθμου.

Τα πειραματικά αποτελέσματα ανέδειξαν τον RF-BR ως τον πλέον αποδοτικό ταξινομητή στο σύνολο σχεδόν των datasets, επιτυγχάνοντας σταθερά τις καλύτερες επιδόσεις τόσο ως προς την Accuracy όσο και τον Hamming Loss. Η απόδοσή του ενισχύθηκε ακόμη περισσότερο μετά την ενσωμάτωση των embeddings από τον node2Vec, καθιστώντας τον την πλέον ενδεδειγμένη επιλογή σε προβλήματα πολυετικετικής ταξινόμησης με χρήση μετασχηματισμένων χαρακτηριστικών. Οι υπόλοιποι ταξινομητές, όπως ο BRkNN, MLkNN, MLARAM και MLTSVM, δεν κατέλαβαν την πρώτη θέση σε κανένα dataset, αν και παρουσίασαν ικανοποιητική σταθερότητα και συγκρίσιμα αποτελέσματα σε επιλεγμένα σύνολα δεδομένων.

Στη δεύτερη φάση της μελέτης, η χρήση του node2Vec οδήγησε σε σημαντική ενίσχυση της απόδοσης, κυρίως όταν χρησιμοποιήθηκε σε συνδυασμό με τον RF-BR. Η αξιοποίηση των συνεμφανίσεων μεταξύ ετικετών μέσω γραφημάτων αποδείχθηκε ιδιαίτερα αποτελεσματική στη βελτίωση της γενίκευσης των μοντέλων, με τις καλύτερες επιδόσεις να καταγράφονται στην τρίτη παραμετροποίηση του node2Vec. Το σύνολο των ευρημάτων καταδεικνύει τη σημασία της ενσωμάτωσης εξωτερικής πληροφορίας (όπως η γραφική συσχέτιση των ετικετών) για την επίτευξη υψηλής ακρίβειας και αξιοπιστίας.

Συνολικά, η εργασία κατέδειξε τη σημασία της επιλογής κατάλληλης στρατηγικής ταξινόμησης ανάλογα με τα χαρακτηριστικά του προβλήματος, καθώς και την προστιθέμενη αξία των τεχνικών αναπαράστασης κόμβων στη βελτίωση της ακρίβειας και της γενίκευσης των μοντέλων. Η εμπειρική τεκμηρίωση των αποτελεσμάτων ενισχύει τη σημασία της πειραματικής αξιολόγησης σε πολλαπλά επίπεδα και ανοίγει τον δρόμο για ακόμη πιο αποδοτικές υβριδικές προσεγγίσεις στο μέλλον.

5.2 Μελλοντική Έρευνα

Η παρούσα εργασία ανέδειξε τη σημαντική επίδραση που μπορεί να έχει η ενσωμάτωση αναπαραστάσεων κόμβων (embeddings) στην απόδοση αλγορίθμων ταξινόμησης σε προβλήματα πολλαπλών ετικετών. Ωστόσο, προέκυψαν ορισμένες ενδιαφέρουσες κατευθύνσεις για περαιτέρω μελέτη, οι οποίες θα μπορούσαν να ενισχύσουν την κατανόηση και την αποτελεσματικότητα των προσεγγίσεων που εφαρμόστηκαν.

Πρώτον, μία φυσική επέκταση της παρούσας έρευνας είναι η διερεύνηση εναλλακτικών μεθόδων δημιουργίας ενσωματώσεων, όπως οι GraphSAGE, DeepWalk, LINE ή ακόμα και τεχνικές από το πεδίο του representation learning όπως το BERT για περιπτώσεις όπου τα δεδομένα συνοδεύονται από κείμενο. Η συγκριτική αξιολόγηση διαφορετικών μεθόδων αναπαράστασης, σε συνδυασμό με ποικίλους ταξινομητές, θα μπορούσε να οδηγήσει σε πιο γενικεύσιμα συμπεράσματα.

Επιπλέον, αξίζει να μελετηθεί η αυτόματη επιλογή ή βελτιστοποίηση υπερπαραμέτρων τόσο για τις τεχνικές ενσωμάτωσης (π.χ. διαστάσεις, μήκος διαδρομών) όσο και για τους ίδιους τους αλγορίθμους ταξινόμησης. Η εφαρμογή τεχνικών βελτιστοποίησης όπως οι Grid Search, Bayesian Optimization, ή AutoML, μπορεί να αναδείξει ακόμα πιο αποδοτικές παραμετροποιήσεις για κάθε συγκεκριμένο σύνολο δεδομένων.

Μια άλλη ενδιαφέρουσα κατεύθυνση είναι η διεύρυνση του πλήθους και της ποικιλίας των datasets, συμπεριλαμβάνοντας πραγματικά, σύνθετα σύνολα δεδομένων από τομείς όπως η βιοϊατρική ή τα συστήματα συστάσεων, όπου τα σενάρια πολλαπλών ετικετών είναι ιδιαίτερα απαιτητικά και οι ετικέτες παρουσιάζουν υψηλή αλληλεξάρτηση ή σπανιότητα.

Παράλληλα, μπορεί να διερευνηθεί η δυναμική προσαρμογή των ενσωματώσεων κατά τη διάρκεια της εκπαίδευσης ενός ταξινομητή (end-to-end training), αντί της χρήσης στατικών αναπαραστάσεων. Ένα τέτοιο ενοποιημένο πλαίσιο ενδέχεται να επιτρέψει την καλύτερη εξειδίκευση των χαρακτηριστικών στις ανάγκες του ταξινομητή.

Τέλος, μία πολλά υποσχόμενη προοπτική είναι η εφαρμογή deep learning προσεγγίσεων, όπως οι sequence-to-sequence αρχιτεκτονικές ή τα Graph Neural Networks (GNNs), που μπορούν να ενσωματώσουν εγγενώς τη δομή των δεδομένων και τις εξαρτήσεις μεταξύ ετικετών, χωρίς την ανάγκη μετασχηματισμού προβλήματος.

Συμπερασματικά, η μελλοντική έρευνα μπορεί να επεκτείνει ουσιαστικά τα αποτελέσματα της παρούσας μελέτης, ενσωματώνοντας νεότερες τεχνολογίες και πιο εξελιγμένες στρατηγικές αναπαράστασης και μάθησης, προκειμένου να ενισχύσει περαιτέρω την ακρίβεια και τη γενίκευση σε εφαρμογές ταξινόμησης δεδομένων πολλαπλών ετικετών.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Kowsari K, Jafari Meimandi K, Heidarysafa M, Mendu S, Barnes LE, Brown DE. Text Classification Algorithms: A Survey. *Information*. 2019.

[2] Géron A. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. 2nd ed. O'Reilly Media; 2019

[3] Murphy KP. Machine Learning: A Probabilistic Perspective. Cambridge (MA): MIT Press; 2012.

[4] Bishop CM. Pattern Recognition and Machine Learning. New York (NY): Springer; 2006.

[5] Tsoumakas G, Katakis I. Multi-label classification: An overview. International Journal of Data Warehousing and Mining. 2007.

[6] Zhang M-L, Zhou Z-H. A review on multi-label learning algorithms. IEEE Transactions on Knowledge and Data Engineering. 2014.

[7] Boutell MR, Luo J, Shen X, Brown CM. Learning multi-label scene classification. Pattern Recognition. 2004;37(9):1757–71.

[8] Tsoumakas G, Katakis I, Vlahavas I. Mining Multi-label Data. In: Maimon O, Rokach L, editors. *Data Mining and Knowledge Discovery Handbook*. Springer; 2010. p. 667–685.

[9] Grover A, Leskovec J. node2vec: Scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD. 2016. p. 855–64.

[10] Read J, Pfahringer B, Holmes G, Frank E. Classifier chains for multi-label classification. Mach Learn. 2011;85(3):333–59.

[11] Dembczyński K, Waegeman W, Cheng W, Hüllermeier E. On label dependence and loss minimization in multi-label classification. Mach Learn. 2012;88(1):5–45

[12] Tsoumakas G, Vlahavas I. Random k-labelsets: An ensemble method for multilabel classification. In: Proceedings of the 18th European Conference on Machine Learning (ECML); 2007. p. 406–17.

[13] Gibaja E, Ventura S. A tutorial on multilabel learning. ACM Comput Surv. 2014;47(3):1–38.

[14] Dembczyński K, Waegeman W, Cheng W, Hüllermeier E. An exact algorithm for F-measure maximization. In: Advances in Neural Information Processing Systems (NeurIPS); 2011. p. 1404–12.

[15] Zhang M, Zhang K. Multi-label learning by exploiting label dependency. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 2010. p. 999–1008.

[16] Zhang M, Zhou Z. A k-nearest neighbor based algorithm for multi-label classification. IEEE International Conference on Granular Computing (GrC). 2005; 718–721

[17] Zhang M, Zhou Z. ML-KNN: A lazy learning approach to multi-label learning. Pattern Recognition. 2007;40(7):2038–2048.

[18] Ghosh A, Shivakumar M. Multi-label classification using associative memory classifiers. Applied Soft Computing. 2017;52:545–559.

[19] Jayadeva, Khemchandani R, Chandra S. Twin support vector machines for pattern classification. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2007;29(5):905–910.

[20] Breiman L. Random Forests. Mach Learn. 2001.

[21] Biau G. Analysis of a Random Forests Model. J Mach Learn Res. 2012.

[22] Louppe G, Wehenkel L, Sutera A, Geurts P. Understanding variable importances in forests of randomized trees. Adv Neural Inf Process Syst. 2013.

[23] Szymański P, Kajdanowicz T. A scikit-based Python environment for performing multi-label classification. arXiv preprint arXiv:1702.01460. 2017

[24] Tang J, Qu M, Wang M, Zhang M, Yan J, Mei Q. Line: Large-scale information network embedding. In: Proceedings of the 24th international conference on world wide web. 2015. p. 1067–77.

[25] Mikolov T, Chen K, Corrado G, Dean J. Efficient Estimation of Word Representations in Vector Space. arXiv preprint arXiv:1301.3781. 2013.

[26] Nam J, Mencia EL, Kim HJ, Gurevych I, Fürnkranz J. Large-scale multi-label text classification—revisiting neural networks. In: Joint European conference on machine learning and knowledge discovery in databases. Springer; 2014. p. 437–52.

[27] Rios A, Kavuluru R. EMBEDDIA: Embedding-Based Multi-label Classification of Medical Texts with Hierarchies. J Biomed Inform. 2018;84:25–36.

[28] Madjarov G, Kocev D, Gjorgjevikj D, Džeroski S. An extensive experimental comparison of methods for multi-label learning. Pattern Recognit. 2012;45(9):3084–104.

[29] Sokolova M, Lapalme G. A systematic analysis of performance measures for classification tasks. Inf Process Manag. 2009;45(4):427–37.

[30] Tsoumakas G, Spyromitros-Xioufis E, Vilcek J, Vlahavas I. Mulan: A Java library for multi-label learning. J Mach Learn Res. 2011;12:2411–14.

[31] Schapire RE, Singer Y. Boostexter: A boosting-based system for text categorization. Mach Learn. 2000;39(2-3):135–68.

[32] Li Y, Guo Y, Schuurmans D. Semi-supervised multi-label learning by constrained non-negative matrix factorization. In: Proceedings of the AAAI Conference on Artificial Intelligence. 2010;24(1):421–6.

[33] Wu X, Zhu X, Wu GQ, Ding W. Data mining with big data. IEEE Trans Knowl Data Eng. 2014;26(1):97–107.

[34] https://www.uco.es/kdis/mllresources/ [Ανακτήθηκε στις 21-02-2025]

[35] Douglas Turnbull, Luke Barrington, David Torres and Gert Lanckriet. Semantic Annotation and Retrieval of Music and Sound Effects, IEEE Transactions on Audio, Speech and Language Processing 16(2), pp. 467-476, 2008.

[36] Forrest Briggs, Yonghong Huang, Raviv Raich, Konstantinos Eftaxias, Zhong Lei, William Cukierski, Sarah Frey Hadley, Adam Hadley, Matthew Betts, Xiaoli Z. Fern, Jed Irvine, Lawrence Neal, Anil Thomas, Gábor Fodor, Grigorios Tsoumakas, Hong Wei Ng, Thi Ngoc Tho Nguyen, Heikki Huttunen, Pekka Ruusuvuori, Tapio Manninen, Aleksandr Diment, Tuomas Virtanen, Julien Marzat, Joseph Defretin, Dave Callender, Chris Hurlburt, Ken Larrey, and Maxim Milakov. The 9th annual MLSP competition: New methods for acoustic classification of multiple simultaneous bird species in a

noisy environment. In IEEE International Workshop on Machine Learning for Signal Processing, MLSP 2013, Southampton, United Kingdom, September 22-25, 2013, pages 1–8, 2013.

[37] H. Blockeel, S. Džeroski, and J. Grbovic. Simultaneous prediction of multiple chemical parameters of river water quality with tilde. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 1704:32–40, 1999.

[38] Andre Elisseeff and Jason Weston. A kernel method for multi-labelled classification. In In Advances in Neural Information Processing Systems 14, volume 14, pages 681–687, 2001.

[39] C.G.M. Snoek, M.Worring, J.C. van Gemert, J.-M. Geusebroek, A.W.M. Smeulders. 2006. The Challenge Problem for Automated Detection of 101 Semantic Concepts in Multimedia, In Proceedings of ACM Multimedia, 421-430.

[40] Matthew R. Boutell, Jiebo Luo, Xipeng Shen, and Christopher M. Brown. Learning multi-label scene classification. Pattern Recognition, 37(9):1757–1771, September 2004.

[41] G. Tsoumakas, I. Katakis, and I. Vlahavas. Effective and Efficient Multilabel Classification in Domains with Large Number of Labels. In Proc. ECML/PKDD 2008 Workshop on Mining Multidimensional Data (MMD'08), 2008.

[42] E.C. Goncalves, Alexandre Plastino, and Alex A. Freitas. A genetic algorithm for optimizing the label ordering in multi-label classifier chains. In IEEE 25th International Conference on Tools with Artificial Intelligence, pages 469–476. IEEE Computer Society Conference Publishing Services (CPS), 2013.

ΚΩΔΙΚΑΣ ΥΛΟΠΟΙΗΣΗΣ

Κώδικας Υλοποίησης BRkNNaClassifier

```
import numpy as np
import networkx as nx
from node2vec import Node2Vec
from skmultilearn.adapt import BRkNNaClassifier
from sklearn.metrics import hamming_loss, accuracy_score
from scipy.sparse import csr_matrix
from sklearn.neighbors import NearestNeighbors
import pandas as pd
from sklearn.model_selection import train_test_split
from scipy.io import arff
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import time #μετρηση του χρόνου εκτέλεσης

# ===== Monkey Patch για MLkNN, BRkNN =====
from sklearn.base import BaseEstimator
def _get_param_names(cls):
    return [key for key in cls.__init__.__code__.co_varnames if key != 'self']
BaseEstimator._get_param_names = classmethod(_get_param_names)

# Monkey Patch για NearestNeighbors
_original_init = NearestNeighbors.__init__
def patched_init(self, *args, **kwargs):
    if len(args) == 1 and not kwargs.get('n_neighbors'):
        kwargs['n_neighbors'] = args[0]
        args = ()
    _original_init(self, *args, **kwargs)
NearestNeighbors.__init__ = patched_init

# ===== Φόρτωση δεδομένων ARFF =====
data, meta = arff.loadarff('CAL500.arff')
df = pd.DataFrame(data)

X = df.iloc[:, :68].values.astype(float)
y = df.iloc[:, -174:].apply(lambda x: x == b'1').astype(int).values  #Binary
μετατροπή
numOfLabels = 174 #metavliti pou krataei to plithos twn labels

# ===== Διαχωρισμός σε Training και Test Set =====
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)
start_time = time.time()

#Arxikos Yplogismos prin ti sygxwneusi
##classifierBrKnn = BRkNNaClassifier(k=1)
```

```
classifierBrKnn = BRkNNaClassifier(k=174) #gia k=174 dinei to xamilotero
hamming loss = 0.1365608586435259
classifierBrKnn.fit(X_train, y_train)  # Ekpaidefsi tou taksinomiti

#PREDICTION on the test set
y_pred = classifierBrKnn.predict(X_test) # Provlepw ta labels gia to X_test

#EVALUATE performance
# Μέτρηση Χρόνου πριν τα embeddings
print("Hamming Loss BRkNN Before Embeddings:", hamming_loss(y_test,
y_pred.astype(int)))
end_time = time.time()
print("Accuracy BRkNN Before Embeddings:", accuracy_score(y_test,
y_pred.astype(int)))
print("Process Time Before Embeddings:", end_time - start_time, "seconds")

# ===== Δημιουργία Γράφου Συνεμφάνισης Ετικετών =====
start_time = time.time()
def build_label_cooccurrence_graph(y):
    num_labels = y.shape[1] #plithos twn labels
    G = nx.Graph()

    for label in range(num_labels):
        G.add_node(label)

    for sample in y:
        labels = np.where(sample == 1)[0]
        for i in range(len(labels)):
            for j in range(i + 1, len(labels)):
                if G.has_edge(labels[i], labels[j]):
                    G[labels[i]][labels[j]]['weight'] += 1
                else:
                    G.add_edge(labels[i], labels[j], weight=1)
    return G

G = build_label_cooccurrence_graph(y_train)

# ===== Ενσωμάτωση Ετικετών με Node2Vec =====
##node2vec = Node2Vec(G, dimensions=64, walk_length=10, num_walks=100,
workers=1)
##node2vec = Node2Vec(G, dimensions=64, walk_length=10, num_walks=100,
workers=2)
node2vec = Node2Vec(G, dimensions=128, walk_length=20, num_walks=200,
workers=4)
model = node2vec.fit(window=5, min_count=1, batch_words=4)

label_embeddings = np.array([model.wv[str(label)] for label in
range(y_train.shape[1])])
```

```
# ===== Clustering για Μείωση Ετικετών =====
num_new_labels = int(numOfLabels*0.3) #για labels>20
##num_new_labels = int(numOfLabels*0.5) #για labels<10
##num_new_labels = 10 #για labels μεταξύ 10-20

kmeans = KMeans(n_clusters=num_new_labels, random_state=42)
label_clusters = kmeans.fit_predict(label_embeddings)

# ===== Συγχώνευση Labels σύμφωνα με τα Clusters =====
def merge_labels(y, label_clusters):
    new_y = np.zeros((y.shape[0], num_new_labels), dtype=int)
    for i, sample_labels in enumerate(y):
        active_labels = np.where(sample_labels == 1)[0]
        for label in active_labels:
            new_cluster = label_clusters[label]
            new_y[i, new_cluster] = 1
    return new_y

# Εφαρμογή στους πίνακες train και test
y_train_new = merge_labels(y_train, label_clusters)
y_test_new = merge_labels(y_test, label_clusters)

# ===== Εμπλουτισμός Χαρακτηριστικών με Embeddings =====
def transform_with_embeddings(X, y, label_embeddings):
    transformed_features = []
    for i in range(X.shape[0]):
        labels = np.where(y[i] == 1)[0]
        if len(labels) > 0:
            embedded_vector = np.mean(label_embeddings[labels], axis=0)
        else:
            embedded_vector = np.zeros(label_embeddings.shape[1])
        combined_features = np.hstack((X[i].flatten(), embedded_vector))
        transformed_features.append(combined_features)
    return np.array(transformed_features)

X_train_embedded = transform_with_embeddings(X_train, y_train,
label_embeddings)
X_test_embedded = transform_with_embeddings(X_test, y_test, label_embeddings)

# ===== Εκπαίδευση Ταξινομητών =====
clfBrKnn = BRkNNaClassifier(k=num_new_labels)
clfBrKnn.fit(X_train_embedded, y_train_new)  # Ekpaidefsi tou taksinomiti
BRkNN_predictions = clfBrKnn.predict(csr_matrix(X_test_embedded))

# ===== Αξιολόγηση Απόδοσης =====
print("Hamming Loss BRkNN(με νέα labels):", hamming_loss(y_test_new,
BRkNN_predictions.astype(int)))
end_time = time.time()
```

```
print("Accuracy BRkNN(με νέα labels):", accuracy_score(y_test_new,
BRkNN_predictions.astype(int)))
print("Process Time After Embeddings:", end_time - start_time, "seconds")

# ===== Οπτικοποίηση =====
# 1. Αρχικός Γράφος
plt.figure(figsize=(10, 8))
pos = nx.spring_layout(G, seed=42)

nx.draw_networkx_nodes(G, pos, node_size=700, node_color='skyblue')
edges = G.edges(data=True)
weights = [edge[2]['weight'] for edge in edges]
nx.draw_networkx_edges(G, pos, edgelist=G.edges(), width=[w * 0.5 for w in
weights])

# Ετικέτες των αρχικών labels
initial_labels = {i: f'Label {i}' for i in G.nodes()}
nx.draw_networkx_labels(G, pos, labels=initial_labels, font_size=9,
font_color='yellow')

plt.title("Γράφος Συνεμφάνισης Ετικετών (Αρχικά Labels)")
plt.axis('off')
plt.show()

# 2. Γράφος μετά τη Συγχώνευση Labels
# Δημιουργία Νέου Γράφου για τα Clusters
G_new = nx.Graph()

# Προσθήκη κόμβων (clusters)
for cluster_id in range(num_new_labels):
    G_new.add_node(cluster_id)

# Δημιουργία ακμών μεταξύ clusters αν υπήρχαν σχέσεις στα αρχικά δεδομένα
for (label1, label2, data) in G.edges(data=True):
    cluster1 = label_clusters[label1]
    cluster2 = label_clusters[label2]
    if cluster1 != cluster2:  # Ενώνουμε διαφορετικά clusters
        if G_new.has_edge(cluster1, cluster2):
            G_new[cluster1][cluster2]['weight'] += data['weight']
        else:
            G_new.add_edge(cluster1, cluster2, weight=data['weight'])

# Οπτικοποίηση Νέου Γράφου
plt.figure(figsize=(8, 6))
pos_new = nx.spring_layout(G_new, seed=42)

nx.draw_networkx_nodes(G_new, pos_new, node_size=800, node_color='lightgreen')
new_weights = [edge[2]['weight'] for edge in G_new.edges(data=True)]
nx.draw_networkx_edges(G_new, pos_new, width=[w * 0.5 for w in new_weights])
```

```
# Ετικέτες των νέων clusters
new_labels = {i: f'Cluster {i}' for i in G_new.nodes()}
nx.draw_networkx_labels(G_new, pos_new, labels=new_labels, font_size=9,
font_color='red')

plt.title("Γράφος μετά τη Συγχώνευση Ετικετών (Clusters)")
plt.axis('off')
plt.show()
```