ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ
ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΤΜΗΜΑΤΟΣ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΕΝΙΣΧΥΣΗ ΕΠΙΔΟΣΗΣ ΣΕ ΠΡΟΒΛΗΜΑΤΑ ΚΑΤΗΓΟΡΙΟΠΟΙΗΣΗΣ ΜΕ ΤΗ ΧΡΗΣΗ ΜΕΘΕΥΡΕΤΙΚΩΝ ΑΛΓΟΡΙΘΜΩΝ: ΑΝΑΛΥΣΗ ΑΛΓΟΡΙΘΜΩΝ

Διπλωματική Εργασία

του
Ιωάννη Τσάμη

Θεσσαλονίκη, Οκτώβριος 2023

ΕΝΙΣΧΥΣΗ ΕΠΙΔΟΣΗΣ ΣΕ ΠΡΟΒΛΗΜΑΤΑ ΚΑΤΗΓΟΡΙΟΠΟΙΗΣΗΣ ΜΕ ΤΗ ΧΡΗΣΗ ΜΕΘΕΥΡΕΤΙΚΩΝ ΑΛΓΟΡΙΘΜΩΝ: ΑΝΑΛΥΣΗ ΑΛΓΟΡΙΘΜΩΝ

Ιωάννης Τσάμης

Πτυχίο Μηχανικός Πληροφορικής, Σέρρες, 2018

Διπλωματική Εργασία

υποβαλλόμενη για τη μερική εκπλήρωση των απαιτήσεων του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΤΙΤΛΟΥ ΣΠΟΥΔΩΝ ΣΤΗΝ ΕΦΑΡΜΟΣΜΕΝΗ ΠΛΗΡΟΦΟΡΙΚΗ

Επιβλέπων Καθηγητής
Άγγελος Σιφαλέρας

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 31/11/2023

Σιφαλέρας Άγγελος	Σαμαράς Νικόλαος	Δημήτριος Χρήστου-Βαρσακέλης
.....	.....	.....

Ιωάννης Τσάμης

Περίληψη

Η παρούσα εργασία εξετάζει την βελτίωση της απόδοσης κατηγοριοποίησης μέσω μεθευρετικών αλγορίθμων. Η εργασία αρχίζει με μια εισαγωγή, παρουσιάζοντας τον γενικό σκοπό της μελέτης. Στη συνέχεια, το πρώτο κεφάλαιο εστιάζει στην εξόρυξη δεδομένων και την κατηγοριοποίηση, παρουσιάζοντας τις εφαρμογές τους και τη μηχανική μάθηση.

Στο δεύτερο κεφάλαιο, εξετάζονται οι μεθευρετικοί αλγόριθμοι, με έμφαση στη βελτίωση της απόδοσης μέσω έξυπνων ευρετικών προσεγγίσεων. Επίσης, παρουσιάζονται τα χαρακτηριστικά τους και οι εφαρμογές τους.

Το τρίτο κεφάλαιο εστιάζει στις βασικές αρχές της κατηγοριοποίησης και της εξόρυξης δεδομένων. Παρουσιάζονται τεχνικές, μέθοδοι και αλγόριθμοι κατηγοριοποίησης, όπως ο SVM, το k-NN, τα Decision Trees, τα Random Forests, Naive Bayes και τα Τεχνητά Νευρωνικά Δίκτυα. Το κεφάλαιο κλείνει με την ανάλυση των μετρικών που χρησιμοποιούνται στα προβλήματα κατηγοριοποίησης.

Στο τέταρτο κεφάλαιο, εξετάζεται η εφαρμογή μεθευρετικών αλγορίθμων στην κατηγοριοποίηση ιατρικών δεδομένων. Παρουσιάζεται η χρήση των αλγορίθμων σε μεγάλα δεδομένα του τομέα της υγειονομικής περίθαλψης και της βιολογίας. Επίσης, περιγράφεται η διαδικασία ταξινόμησης και κατηγοριοποίησης ιατρικών δεδομένων, με εφαρμογή μεθευρετικών αλγορίθμων και γενετικού αλγορίθμου. Το κεφάλαιο κλείνει με την παρουσίαση του συνόλου δεδομένων που χρησιμοποιήθηκε και των αποτελεσμάτων της εφαρμογής.

Λέξεις Κλειδιά: Εξόρυξη δεδομένων, μηχανική μάθηση, μεθευρετικοί, αλγόριθμος προσομοιωμένης ανόπτησης, K-NN

Abstract

This thesis examines the improvement of classification performance through meta-heuristic algorithms. The paper begins with an introduction, presenting the general purpose of the study. Then, the first chapter focuses on data mining and classification, presenting their applications and machine learning.

In the second chapter, meta-heuristic algorithms are discussed, with an emphasis on improving performance through intelligent heuristic approaches. Also, their characteristics and applications are presented.

The third chapter focuses on the fundamentals of categorization and data mining. Techniques, methods and classification algorithms are presented, such as SVM, PCA, k-NN, Decision Trees, Random Forests, Naive Bayes and Artificial Neural Networks. The chapter closes with the exposition of metrics, which are used for classification problems.

In the fourth chapter, the application of meta-heuristic algorithms to the categorization of medical data is examined. The use of algorithms in big data in the field of health care and biology is presented. Also, the process of classifying and categorizing medical data is described, with the application of meta-heuristic algorithms and genetic algorithm. The chapter closes with the presentation of the data set used and the results of the application.

Keywords: Data mining, machine learning, metaheuristic algorithms, simulated annealing, K-NN

Περιεχόμενα

Κατάλογος Εικόνων

Κατάλογος Κώδικα

Εισαγωγή

Η εξόρυξη δεδομένων χρησιμοποιεί μια ποικιλία αλγορίθμων και τεχνικών από τον τομέα της Μηχανικής Μάθησης, της Στατιστικής, της Τεχνητής Νοημοσύνης και άλλων επιστημονικών πεδίων, για να εξάγει πληροφορίες από τα δεδομένα[1]. Σκοπός της είναι να αναδείξει κρυμμένα πρότυπα, γνώση και εργαλεία που μπορούν να βοηθήσουν στην καλύτερη κατανόηση και λήψη αποφάσεων για τον τομέα ενδιαφέροντος. Η κατηγοριοποίηση αναφέρεται στην τεχνική της ομαδοποίησης αντικειμένων ή δειγμάτων σε κατηγορίες ή κλάσεις, βάσει των χαρακτηριστικών τους. Είναι μια μορφή επιβλεπόμενης μάθησης, όπου υπάρχουν έτοιμες ετικέτες κατηγορίας για το σύνολο εκπαίδευσης και τον στόχο είναι να εκπαιδευτεί ένα μοντέλο που να μπορεί να κατατάξει νέα δεδομένα σε μία από τις κατηγορίες [2].

Τα τελευταία χρόνια, στον ερευνητικό τομέα γίνονται πολλές προσπάθειες και δοκιμές υπαρχουσών αλλά και νέων τεχνικών με σκοπό της βελτιστοποίηση της εξόρυξης των δεδομένων και την εξαγωγή χρήσιμης γνώσης και κατηγοριοποίησης δεδομένων, όπως για παράδειγμα στον ιατρικό τομέα [3].

Πρόβλημα – Σημαντικότητα του θέματος

Σκοπός – Στόχοι

Ο σκοπός της εργασίας είναι να μελετήσει και να αναλύσει την εφαρμογή των μεθευρετικών (meta-heuristic) αλγορίθμων στην εξόρυξη δεδομένων και την κατηγοριοποίηση. Η εργασία αναζητά νέους τρόπους βελτίωσης της απόδοσης των κλασικών αλγορίθμων που χρησιμοποιούνται για την κατηγοριοποίηση, όπως οι SVM και PCA, και επιδιώκει να εξετάσει την αποτελεσματικότητα των μεθευρετικών μεθόδων σε αυτό το πλαίσιο.

Οι μεθευρετικοί αλγόριθμοι είναι ισχυρά εργαλεία που εμπνέονται από φυσικά φαινόμενα ή αναλογούν στην επίλυση προβλημάτων βασιζόμενοι σε μετα-γνώση ή εμπειρία. Με την εφαρμογή τους στην εξόρυξη δεδομένων και την κατηγοριοποίηση, μπορούν να βελτιώσουν την ακρίβεια, την ταχύτητα και την απόδοση των αλγορίθμων που χρησιμοποιούνται για την ανάλυση δεδομένων [1].

Η εργασία αναλύει τους διάσημους αλγορίθμους και τεχνικές που συνδυάζουν μεθευρετικούς όπως ο αλγόριθμος Προσομοιωμένης Ανόπτησης (*Simulated Annealing Algorithm),* και διερευνά τις επιπτώσεις τους στην απόδοση της κατηγοριοποίησης. Με την εξέταση και την σύγκριση αυτών των αλγορίθμων με τις κλασικές προσεγγίσεις, η εργασία επιδιώκει να παράσχει μια καλύτερη κατανόηση των πλεονεκτημάτων και των περιορισμών της κάθε μεθόδου.

1 Εξόρυξη δεδομένων και Μηχανική Μάθηση

1.1 Εξόρυξη δεδομένων

Η εξόρυξη δεδομένων αποτελεί ένα σημαντικό πεδίο της επιστήμης των υπολογιστών που ασχολείται με την ανακάλυψη πολύτιμων πληροφοριών από μεγάλα και πολυδιάστατα σύνολα δεδομένων. Μέσω της εξόρυξης δεδομένων, οι ερευνητές και επαγγελματίες μπορούν να ανακαλύψουν πρότυπα, τάσεις και δομές που κρύβονται μέσα στα δεδομένα και να λάβουν σημαντικές αποφάσεις βασιζόμενοι στις πληροφορίες αυτές.

Ένα από τα πιο διαδεδομένα εργαλεία στην εξόρυξη δεδομένων είναι η κατηγοριοποίηση. Η κατηγοριοποίηση είναι μια διαδικασία κατά την οποία τα αντικείμενα ομαδοποιούνται σε κατηγορίες ή κλάσεις βάσει των χαρακτηριστικών τους. Οι αλγόριθμοι κατηγοριοποίησης, όπως οι SVM, αναλαμβάνουν τον ρόλο να μάθουν τα μοτίβα από ένα σύνολο εκπαίδευσης και στη συνέχεια να τα εφαρμόσουν σε νέα δεδομένα για να τα κατηγοριοποιήσουν αποτελεσματικά[4].

Ένα από τα πλεονεκτήματα της κατηγοριοποίησης είναι η ευρεία της εφαρμογή σε διάφορους τομείς, όπως η ιατρική. Στην ιατρική, η κατηγοριοποίηση παίζει κρίσιμο ρόλο στη διάγνωση ασθενειών και στην πρόβλεψη θεραπειών. Αναλυτές δεδομένων μπορούν να χρησιμοποιήσουν αλγορίθμους κατηγοριοποίησης για να διαχωρίσουν ασθενείς που ανταποκρίνονται σε συγκεκριμένα φάρμακα από αυτούς που δεν ανταποκρίνονται, καθώς και για να προβλέψουν πιθανές παθήσεις βάσει κλινικών και γενετικών δεδομένων [5].

Επιπλέον, η εφαρμογή τεχνικών όπως ο αλγόριθμος Προσομοιωμένης Ανόπτησης επιτρέπει την αποτελεσματική και αυτοματοποιημένη εξόρυξη δεδομένων, καθιστώντας την πιο αξιόπιστη και αποδοτική.

1.2 Εφαρμογές της εξόρυξης και της κατηγοριοποίησης δεδομένων

Η εξόρυξη δεδομένων και η κατηγοριοποίηση εφαρμόζονται σε πολλούς διαφορετικούς τομείς και κλάδους, καθιστώντας τις δυνατότητες τους ευρέως εφαρμόσιμες σε προβλήματα πρακτικού ενδιαφέροντος. Ορισμένοι από αυτούς τους τομείς περιλαμβάνουν [6]:

1. Χρηματοοικονομική: Η εξόρυξη δεδομένων εφαρμόζεται στο χρηματοοικονομικό τομέα για πρόβλεψη των αγορών, αναγνώριση τάσεων και πρόβλεψη κινδύνων.
2. Λιανικό εμπόριο: Κατηγοριοποίηση δεδομένων χρησιμοποιείται για την εντοπισμό προτιμήσεων και συμπεριφοράς των πελατών και τον σχεδιασμό πιο αποτελεσματικών στρατηγικών πωλήσεων.
3. Κοινωνικά δίκτυα και Μέσα Κοινωνικής Δικτύωσης: Η εξόρυξη δεδομένων χρησιμοποιείται για την ανάλυση και πρόβλεψη των συμπεριφορών και των τάσεων στα κοινωνικά δίκτυα.
4. Υγεία και Ιατρική: Στον τομέα της υγείας, η κατηγοριοποίηση δεδομένων βοηθά στη διάγνωση ασθενειών, την πρόβλεψη εξελίξεων και στην ανάλυση των ιατρικών δεδομένων.
5. Επιστήμη και Έρευνα: Η εξόρυξη δεδομένων και η κατηγοριοποίηση χρησιμοποιούνται για την ανακάλυψη νέων προτύπων και την εξαγωγή γνώσης από επιστημονικά δεδομένα.
6. Εκπαίδευση: Στον τομέα της εκπαίδευσης, η κατηγοριοποίηση δεδομένων χρησιμοποιείται για τον αναγνώριση της προόδου και της απόδοσης των μαθητών.
7. Περιβάλλον και Ενέργεια: Η εξόρυξη δεδομένων εφαρμόζεται για τον προσδιορισμό των παραγόντων που επηρεάζουν το περιβάλλον και την ανανεώσιμη ενέργεια.

Αυτά είναι μόνο μερικοί από τους τομείς στους οποίους η εξόρυξη δεδομένων και η κατηγοριοποίηση έχουν καθιερωθεί ως ισχυρά εργαλεία για την ανάλυση και την πρόβλεψη πολύπλοκων δεδομένων. Η συνεχής εξέλιξη της τεχνολογίας και η ανάπτυξη νέων αλγορίθμων ανοίγουν νέες προοπτικές για την εφαρμογή τους σε περισσότερους τομείς και για την επίλυση πιο σύνθετων προβλημάτων.

1.3 Machine Learning- Μηχανική μάθηση

Η μηχανική μάθηση (Machine Learning-ML) διαδραματίζει κρίσιμο ρόλο στις τεχνικές της εξόρυξης δεδομένων και της κατηγοριοποίησης. Στην πραγματικότητα, η μηχανική μάθηση είναι ένας από τους κύριους πυλώνες που τροφοδοτούν αυτές τις τεχνικές και τους επιτρέπουν να λειτουργούν αποτελεσματικά και αποδοτικά [5].

Ο ρόλος της μηχανικής μάθησης στην εξόρυξη δεδομένων είναι να ανακαλύψει μοτίβα, τάσεις και κρυμμένες σχέσεις μεταξύ των δεδομένων, χωρίς να χρειάζεται να προγραμματιστούν εξαρχής συγκεκριμένοι κανόνες. Οι αλγόριθμοι ML, όπως οι SVM, k-NN, Decision Trees, Random Forests κ.ά., μπορούν να εκπαιδευτούν πάνω σε ένα σύνολο δεδομένων, και με βάση την εμπειρία αυτή, να προβλέψουν και να κατηγοριοποιήσουν νέα δεδομένα[3].

Για παράδειγμα, στην κατηγοριοποίηση ιατρικών δεδομένων, η μηχανική μάθηση μπορεί να εκπαιδευτεί να αναγνωρίζει διάφορες ασθένειες βάσει του κειμένου των εγγράφων και να δημιουργεί αυτόματα ετικέτες κατηγορίας για τις διάφορες ασθένειες [7].

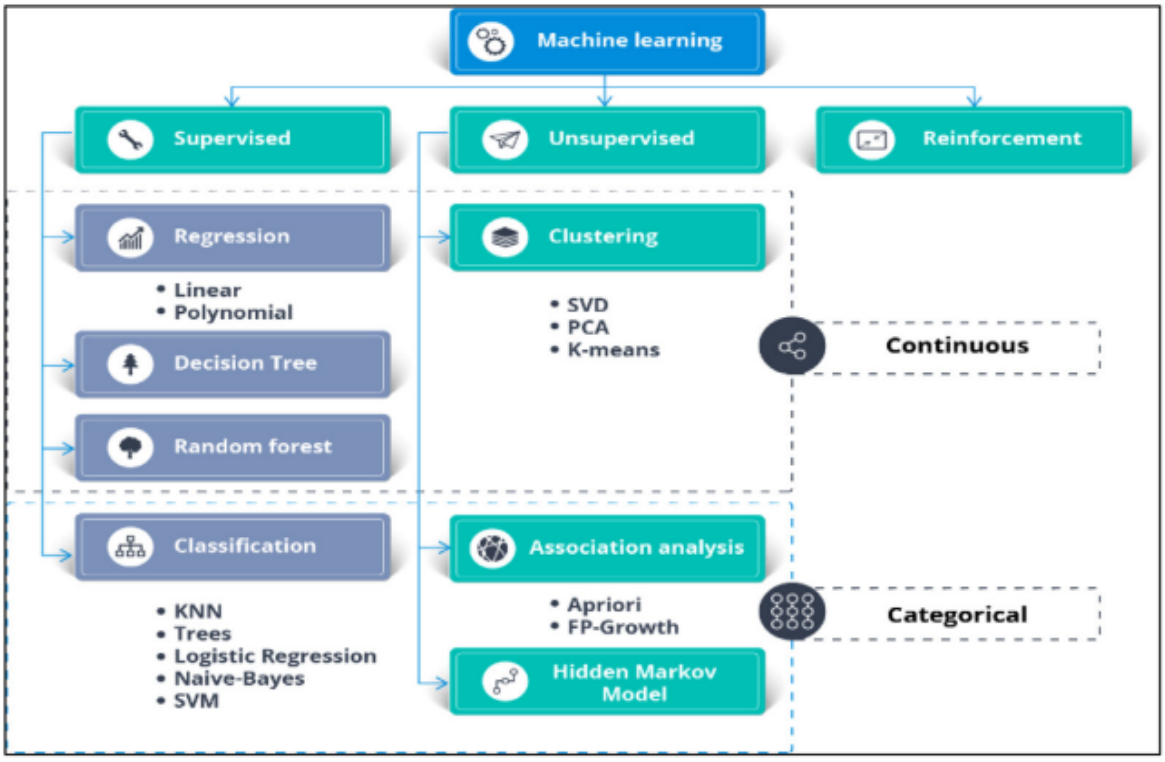
Τέλος, οι τεχνικές μηχανικής μάθησης συχνά χρησιμοποιούνται για την επεξεργασία και προετοιμασία των δεδομένων πριν από την εφαρμογή των τεχνικών εξόρυξης και κατηγοριοποίησης. Αυτό περιλαμβάνει την καθαρισμό των δεδομένων, τη μείωση της διάστασης και την κανονικοποίηση, με στόχο να βελτιώσει την απόδοση των αλγορίθμων και να μειώσει τον χρόνο εκπαίδευσης και εκτέλεσης [7].

1.4 Κατηγορίες αλγορίθμων Μηχανικής Μάθησης

Η τεχνολογία της μηχανικής μάθησης (ML) ανήκει στον τομέα της τεχνητής νοημοσύνης και επικεντρώνεται στη χρήση αλγορίθμων και στατιστικών προσεγγίσεων για να δοθεί στους υπολογιστές τη δυνατότητα "μάθησης" από δεδομένα. Στόχος είναι να βελτιώσουν την απόδοσή τους στην επίλυση καθημερινών εργασιών χωρίς να χρειάζεται να προγραμματιστούν εκ των προτέρων για κάθε περίπτωση. Αυτά τα συστήματα αναπτύσσουν την ικανότητά τους να μάθουν αυτόνομα καθώς περνά ο χρόνος, χρησιμοποιώντας τη γνώση που αποκομίζουν από δεδομένα και παρατηρήσεις στον πραγματικό κόσμο. Η γνώση που αποκτούν τους επιτρέπει να εφαρμόζουν αυτήν την γνώση σε νέες καταστάσεις.

Σύμφωνα με τον Bishop [8], οι αλγόριθμοι της μηχανικής μάθησης μπορούν να διακριθούν σε τρεις κύριες κατηγορίες:

1. **Αλγόριθμοι εποπτευόμενης μάθησης**: Σε αυτήν την περίπτωση, οι τιμές των εισόδων και τις αντίστοιχες εξόδων είναι γνωστές εκ των προτέρων. Αυτοί οι αλγόριθμοι επιδιώκουν να κατανοήσουν τη σχέση μεταξύ των εισόδων και των εξόδων, προκειμένου να προβλέπουν τις εξόδους για νέες εισόδους. Αυτοί διαιρούνται σε αλγορίθμους ταξινόμησης και παλινδρόμησης, ανάλογα με τον τύπο της μάθησης που επιδιώκουν.
2. **Αλγόριθμοι μάθησης χωρίς επίβλεψη**: Αυτοί οι αλγόριθμοι χρησιμοποιούνται όταν τα δεδομένα εκπαίδευσης δεν έχουν κατηγοριοποιηθεί. Στη μάθηση χωρίς επίβλεψη, οι τιμές των εισόδων γνωρίζονται, αλλά δεν υπάρχουν συγκεκριμένες τιμές εξόδου. Ο στόχος είναι να ανιχνεύσουν μοτίβα και δομές στα δεδομένα εισόδου [9].
3. **Αλγόριθμοι ενισχυτικής μάθησης (RL)**: Σε αυτήν την περίπτωση, ένας "πράκτορας" μαθαίνει από την αλληλεπίδρασή του με το περιβάλλον του. Κατά τη διάρκεια των επαναλήψεων, ο πράκτορας αναπτύσσει αυτόματα στρατηγικές που μεγιστοποιούν την ανταμοιβή ή ελαχιστοποιούν τον κίνδυνο, βασιζόμενος σε ανατροφοδότηση ανταμοιβής [9].



Εικόνα 1- ***Κατηγορίες αλγορίθμων μηχανικής μάθησης*** **Πηγή:** [9]

Σε αυτό το πλαίσιο, κάθε κατηγορία περιλαμβάνει αρκετούς αλγορίθμους όπως η γραμμική παλινδρόμηση, η κοινόχρηστη ομαδοποίηση, ο Q-Learning και πολλοί άλλοι, που χρησιμοποιούνται για να αντιμετωπίσουν διαφορετικούς τύπους προβλημάτων και καταστάσεις.

1.4.1 Αλγόριθμοι εποπτευόμενης μάθησης

Η εποπτευόμενη μάθηση αποτελεί ένα βασικό υποπεδίο της μηχανικής μάθησης, το οποίο επιδιώκει να αναπτύξει μοντέλα πρόβλεψης ή ταξινόμησης μέσα από την ανάλυση και εκμάθηση των σχέσεων μεταξύ εισόδων και εξόδων. Στο πλαίσιο αυτής της έκθεσης, θα εξετάσουμε διάφορους αλγορίθμους εποπτευόμενης μάθησης, τη λειτουργία τους, τις εφαρμογές τους και παραδείγματα χρήσης [10].

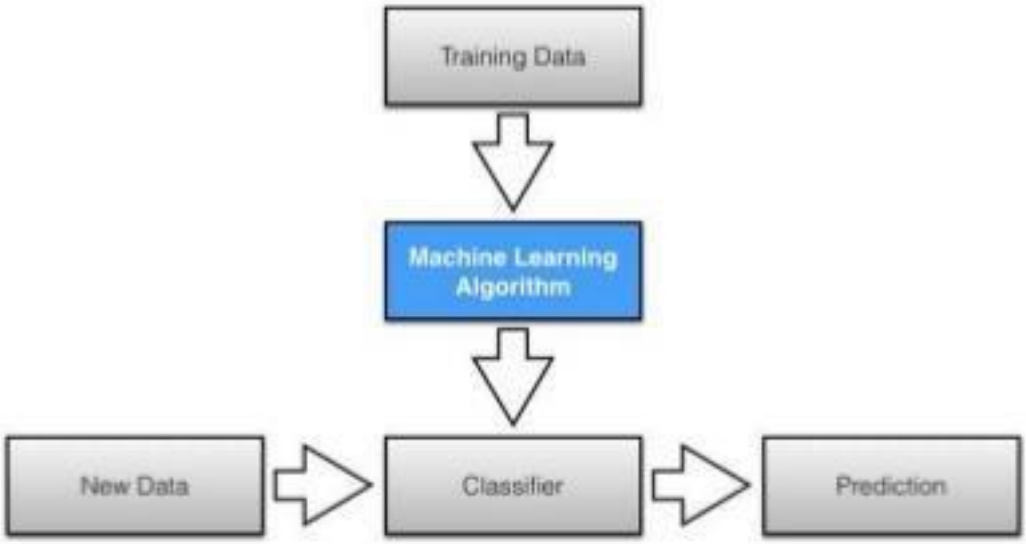
Στην εποπτευόμενη μάθηση, το σύνολο δεδομένων αποτελείται από εισόδους και τις αντίστοιχες επιθυμητές εξόδους (ετικέτες). Ο στόχος είναι να εκπαιδεύσουμε το μοντέλο ώστε να μάθει τη σχέση μεταξύ των εισόδων και των εξόδων, ώστε να μπορεί να προβλέπει τις εξόδους για νέες εισόδους.

Στον τομέα της εποπτευόμενης μάθησης υπάρχουν οι παρακάτω κατηγορίες Αλγορίθμων [11]:

1. **Γραμμική Παλινδρόμηση (Linear Regression)**: Ο αλγόριθμος αυτός επιδιώκει να προσεγγίσει τη σχέση μεταξύ των μεταβλητών εισόδου και εξόδου με ένα γραμμικό μοντέλο. Χρησιμοποιείται κυρίως για προβλέψεις συνεχών τιμών, όπως η πρόβλεψη τιμής ακινήτου βάσει χαρακτηριστικών.
2. **Λογιστική Παλινδρόμηση (Logistic Regression)**: Αν και το όνομα του περιέχει τον όρο "παλινδρόμηση", αυτός ο αλγόριθμος χρησιμοποιείται για προβλήματα ταξινόμησης, κυρίως για δυαδικές κατηγορίες. Προσεγγίζει την πιθανότητα ενός δείγματος να ανήκει σε μία από τις δύο κατηγορίες.
3. **Υποστήριξη Διανυσματικής Μηχανής (Support Vector Machine, SVM)**: Οι SVM αποσκοπούν στην εύρεση ενός υπερεπιπέδου που να διαχωρίζει τα δείγματα των διαφορετικών κατηγοριών με τον μεγαλύτερο δυνατό τρόπο. Χρησιμοποιούνται για προβλήματα ταξινόμησης και μπορούν να επεκταθούν και σε περίπλοκα προβλήματα [11].
4. **Δέντρα Αποφάσεων (Decision Trees)**: Αυτοί οι αλγόριθμοι χτίζουν ένα δέντρο αποφάσεων με βάση τα δεδομένα εκπαίδευσης. Κάθε φύλλο του δέντρου

αντιπροσωπεύει μία απόφαση και το μονοπάτι από τη ρίζα σε ένα φύλλο αντιπροσωπεύει την ακολουθία των αποφάσεων που οδήγησαν στην πρόβλεψη.

5. **Τυχαίο Δάσος (Random Forests)**: Τα τυχαία δάση είναι σύνολα από δέντρα αποφάσεων. Κάθε δέντρο εκπαιδεύεται σε ένα υποσύνολο των δεδομένων και οι προβλέψεις από τα δέντρα συνδυάζονται για την τελική πρόβλεψη.



Εικόνα 2 -*Διαδικασία εποπτευόμενης μάθησης* Πηγή: [11]

Οι αλγόριθμοι εποπτευόμενης μάθησης χρησιμοποιούνται σε πολλές εφαρμογές, όπως:

- Πρόβλεψη τιμών ακινήτων
- Αναγνώριση εικόνων και αντικειμένων
- Αυτόματη κατηγοριοποίηση email (spam ή μη spam)
- Ιατρική διάγνωση
- Προβλέψεις οικονομικών δεδομένων

Οι αλγόριθμοι εποπτευόμενης μάθησης αποτελούν ένα σημαντικό εργαλείο στον τομέα της μηχανικής μάθησης. Από την πρόβλεψη τιμών μέχρι την αναγνώριση εικόνων, αυτοί οι αλγόριθμοι μπορούν να παρέχουν λύσεις σε ποικίλα προβλήματα. Η επιτυχής εφαρμογή τους απαιτεί κατανόηση των αλγορίθμων και προσεκτική επιλογή των χαρακτηριστικών και των παραμέτρων για κάθε πρόβλημα [10].

Μέχρι κα σήμερα, οι αλγόριθμοι εποπτευόμενης μάθησης συνεχίζουν να αναπτύσσονται και να βελτιώνονται, επιτρέποντας στις μηχανές να προσεγγίζουν την ανθρώπινη ικανότητα της μάθησης από τα δεδομένα.

1.4.2 Αλγόριθμοι μάθησης χωρίς επίβλεψη

Οι αλγόριθμοι μη εποπτευόμενης μάθησης αποτελούν μία σημαντική πτυχή της μηχανικής μάθησης, καθώς επιτρέπουν την ανάκτηση αξιόλογων πληροφοριών από μη ετικετοποιημένα δεδομένα. Με τη χρήση αυτών των αλγορίθμων, μπορούμε να ανακαλύψουμε μοτίβα, δομές και κατηγορίες εντελώς αυτόνομα, χωρίς την ανάγκη προκαθορισμένων ετικετών. Στο πλαίσιο αυτής της έκθεσης, θα εξετάσουμε τους αλγορίθμους μη εποπτευόμενης μάθησης με έμφαση στην εξόρυξη δεδομένων και την κατηγοριοποίηση δεδομένων [12].

Στη μη εποπτευόμενη μάθηση, δεν υπάρχουν προκαθορισμένες ετικέτες για τα δεδομένα. Ο στόχος είναι να ανακαλύψουμε δομές, συσχετίσεις και πρότυπα ανάμεσα στα δεδομένα χωρίς να γνωρίζουμε τις κατηγορίες εκ των προτέρων. Δύο κύριες κατηγορίες αλγορίθμων μη εποπτευόμενης μάθησης είναι η εξόρυξη δεδομένων και η κατηγοριοποίηση δεδομένων [12].

a) Εξόρυξη Δεδομένων (Data Mining) Η εξόρυξη δεδομένων αποσκοπεί στο να ανακαλύψει κρυμμένες πληροφορίες, μοτίβα και τάσεις σε μεγάλα σύνολα δεδομένων. Αυτό γίνεται μέσω αλγορίθμων που εφαρμόζουν τεχνικές όπως η ομαδοποίηση (clustering) και η ανίχνευση συσχετίσεων (association rule mining).

- **Ομαδοποίηση (Clustering)**: Οι αλγόριθμοι ομαδοποίησης επιτρέπουν την κατάταξη των δεδομένων σε ομάδες (clusters) βάσει των χαρακτηριστικών τους. Ομάδες δεδομένων μπορεί να περιέχουν παρόμοια στοιχεία που μοιράζονται κάποιο κοινό χαρακτηριστικό, ενώ διαφορετικές ομάδες αντιπροσωπεύουν διαφορετικά μοτίβα.
- **Ανίχνευση Συσχετίσεων (Association Rule Mining)**: Αυτή η τεχνική αναζητά τις συσχετίσεις ανάμεσα στα διάφορα χαρακτηριστικά των δεδομένων. Με βάση τις συσχετίσεις αυτές, μπορούμε να εξάγουμε συμπεράσματα και να προβλέψουμε πιθανές συμπεριφορές.

b) Κατηγοριοποίηση Δεδομένων (Data Categorization) Η κατηγοριοποίηση αποσκοπεί στο να αναθέσει ετικέτες ή κατηγορίες σε μη ετικετοποιημένα δεδομένα βάσει των χαρακτηριστικών τους. Οι αλγόριθμοι που χρησιμοποιούνται για αυτήν την εργασία είναι οι αλγόριθμοι ταξινόμησης.

- **Αλγόριθμοι Ταξινόμησης (Classification Algorithms)**: Οι αλγόριθμοι ταξινόμησης εκπαιδεύονται σε ένα σύνολο δεδομένων που περιλαμβάνει τα χαρακτηριστικά των δειγμάτων και τις αντίστοιχες ετικέτες. Κατά τη διάρκεια της εκπαίδευσης, οι αλγόριθμοι αναπτύσσουν ένα μοντέλο που μπορεί να προβλέψει την κατηγορία των νέων δεδομένων [13].

Οι αλγόριθμοι μη εποπτευόμενης μάθησης έχουν εφαρμογές σε ποικίλους τομείς:

- **Μάρκετινγκ και Διαφήμιση**: Οι αλγόριθμοι εξόρυξης δεδομένων μπορούν να ανακαλύψουν προτεραιότητες και συμπεριφορές καταναλωτών, ενώ οι αλγόριθμοι κατηγοριοποίησης μπορούν να βοηθήσουν στην πρόβλεψη προτιμήσεων και αντιδράσεων.
- **Ιατρική**: Η εξόρυξη δεδομένων μπορεί να ανακαλύψει μοτίβα σε μεγάλα ιατρικά σύνολα δεδομένων, ενώ η κατηγοριοποίηση δεδομένων μπορεί να χρησιμοποιηθεί για τη διάγνωση και την πρόβλεψη ασθενειών.
- **Διαχείριση Ρίσκου**: Η εξόρυξη δεδομένων μπορεί να βοηθήσει στην αναγνώριση πρότυπων ρίσκου και ανωμαλιών σε οικονομικά δεδομένα.

Οι αλγόριθμοι μη εποπτευόμενης μάθησης αποτελούν ισχυρά εργαλεία για την ανάκτηση πληροφοριών από μη ετικετοποιημένα δεδομένα. Με τη χρήση τεχνικών εξόρυξης δεδομένων και κατηγοριοποίησης δεδομένων, μπορούμε να ανακαλύψουμε σημαντικά πρότυπα, να προβλέψουμε συμπεριφορές και να λάβουμε αξιόλογες αποφάσεις σε ποικίλους τομείς. Οι αλγόριθμοι αυτοί συνεχίζουν να εξελίσσονται, ανοίγοντας νέες δυνατότητες για την ανάλυση και την εκμετάλλευση μη ετικετοποιημένων δεδομένων [13].

1.4.3 Αλγόριθμοι Reinforcement Learning

Οι αλγόριθμοι RL επιλύουν προβλήματα στα οποία ένας αποφασιστικός πράκτορας λαμβάνει αποφάσεις σε διαδοχικά χρονικά βήματα με στόχο τη μεγιστοποίηση της συνολικής ανταμοιβής που θα συγκεντρώσει. Κάθε απόφαση που λαμβάνεται επηρεάζει την κατάσταση του περιβάλλοντος και την επόμενη ανταμοιβή που θα ληφθεί. Η διαδικασία εκπαίδευσης του πράκτορα στον RL απαιτεί την εξερεύνηση των διάφορων δράσεων προκειμένου να βρει τη βέλτιστη στρατηγική [14].

Στοιχεία του Αλγορίθμου Reinforcement Learning

Οι αλγόριθμοι RL αποτελούνται από τα εξής βασικά στοιχεία:

1. **Περιβάλλον (Environment)**: Πρόκειται για τον εξωτερικό κόσμο με τον οποίο αλληλεπιδρά ο πράκτορας. Το περιβάλλον καθορίζει τις ανταμοιβές που λαμβάνει ο πράκτορας με βάση τις ενέργειές του.
2. **Πολιτική (Policy)**: Είναι η στρατηγική που ακολουθεί ο πράκτορας για την επιλογή δράσης σε κάθε κατάσταση. Σκοπός είναι η εύρεση της πολιτικής που θα μεγιστοποιήσει την συνολική ανταμοιβή.
3. **Ανταμοιβές (Rewards)**: Οι ανταμοιβές αναπαριστούν το κίνητρο για τον πράκτορα να επιδιώξει μια συγκεκριμένη συμπεριφορά. Ο στόχος του πράκτορα είναι να μεγιστοποιήσει τη συνολική ανταμοιβή που θα συγκεντρώσει κατά τη διάρκεια της αλληλεπίδρασής του με το περιβάλλον.
4. **Μοντέλο (Model)**: Σε ορισμένους αλγορίθμους RL, μπορεί να υπάρχει ένα μοντέλο που προσεγγίζει το περιβάλλον και τις επιπτώσεις των ενεργειών του πράκτορα. Αυτό το μοντέλο μπορεί να χρησιμοποιηθεί για προεκτίμηση των επιπτώσεων των ενεργειών και την ανάπτυξη πολιτικών.

Οι αλγόριθμοι RL έχουν ευρεία εφαρμογή σε πολλούς τομείς, όπως τα αυτόνομα οχήματα, τη ρομποτική, τον χρηματοοικονομικό τομέα, τα παιχνίδια και πολλά άλλα. Επιτρέπουν σε έναν πράκτορα να μάθει πώς να προσαρμόζεται σε ένα πολύπλοκο και αβέβαιο περιβάλλον, παίρνοντας αποφάσεις που μπορεί να είναι εξαιρετικά δύσκολες να προβλεφθούν [14].

Ορισμένοι κλασικοί αλγόριθμοι RL περιλαμβάνουν:

Q-Learning: Ένας αλγόριθμος μοντέλου-ελεύθερου RL που μαθαίνει μια πολιτική που μεγιστοποιεί την αναμενόμενη ανταμοιβή. Το **Q-Learning** είναι ένας εξαιρετικά επιτυχημένος αλγόριθμος της κατηγορίας του ενισχυτικού μάθησης που στοχεύει στο να εκμεταλλευτεί την αλληλεπίδραση ενός πράκτορα με το περιβάλλον του, προκειμένου να μάθει μια βέλτιστη στρατηγική λήψης αποφάσεων. Αυτός ο αλγόριθμος αποτελεί θεμέλιο λίθο της τεχνητής νοημοσύνης και έχει εφαρμοστεί με επιτυχία σε πολλούς τομείς, όπως τα αυτόνομα οχήματα, τη ρομποτική, τη βελτιστοποίηση πόρων και άλλα [15].

Η βασική ιδέα πίσω από το Q-Learning είναι η εκτίμηση της "ποιότητας" μιας δράσης σε μια συγκεκριμένη κατάσταση. Κάθε κατάσταση-δράση ζεύγος αξιολογείται με βάση το Q-Value (ή Quality Value), που αναπαριστά πόσο "καλή" είναι μια δράση σε μια δεδομένη κατάσταση. Ο στόχος είναι να εκμεταλλευτείτε την αναμενόμενη ανταμοιβή που μπορεί να λάβει ο πράκτορας κάνοντας μια συγκεκριμένη δράση, καθώς και το Q-Value της επόμενης κατάστασης που ακολουθεί μετά την εκτέλεση αυτής της δράσης. Ο αλγόριθμος Q-Learning εκπαιδεύει το μοντέλο των Q-Values με βάση την ακόλουθη εξίσωση [15]:

$$Q(s,a) = Q(s,a) + a * \left(r + \gamma * \max\left(Q(s',a')\right) - Q(s,a)\right)$$

Εξίσωση 1

Όπου:

- $Q(s,a)$: Το Q-Value της δράσης "a" στην κατάσταση "s".
- α $(alpha)$: Η εκμάθηση ή ρυθμαπόδοση, που καθορίζει πόσο γρήγορα ενημερώνονται οι Q-Values.
- r: Η ανταμοιβή που λαμβάνει ο πράκτορας κάνοντας τη δράση "a" στην κατάσταση "s".
- γ $(gamma)$: Ο διακριτικός παράγοντας που αντιπροσωπεύει την εκπτωτική στρατηγική, δηλαδή το πόσο σημαντική είναι η μελλοντική ανταμοιβή σε σχέση με την τρέχουσα.
- $max(Q(s',a'))$: Το μέγιστο Q-Value ανάμεσα σε όλες τις δράσεις που είναι εφικτές από την επόμενη κατάσταση "s'".

Ο Q-Learning έχει ευρεία εφαρμογή σε πολλούς τομείς. Ένα από τα πιο γνωστά παραδείγματα είναι το Παιχνίδι του Ταμπλό, όπου ο αλγόριθμος μαθαίνει ποιες κινήσεις είναι οι βέλτιστες για να φτάσει τα πούλια στο τελικό τους σημείο. Επίσης, χρησιμοποιείται σε αυτόνομα οχήματα για την λήψη αποφάσεων κατά την πλοήγηση και την αποφυγή εμποδίων. Επιπλέον, χρησιμοποιείται σε ρομποτικές εφαρμογές για την εκπαίδευση ρομπότ να εκτελούν περίπλοκες εργασίες, όπως το πιάσιμο και τοποθέτηση αντικειμένων [16].

Ο Q-Learning αποτελεί έναν θεμέλιο λίθο της ενισχυτικής μάθησης και έχει προσφέρει εξαιρετικά επιτυχημένες λύσεις σε πολλά προβλήματα από διάφορους τομείς. Η ιδέα της εκμάθησης μέσω αλληλεπίδρασης με το περιβάλλον και της αυτόματης βελτίωσης της στρατηγικής λήψης αποφάσεων έχει αποδειχθεί απίστευτα ισχυρή και αποτελεσματική.

SARSA: Ένας αλγόριθμος που επιλύει το πρόβλημα ελέγχου του Markov, προσεγγίζοντας τη συνάρτηση Q.

Ο αλγόριθμος **SARSA (State-Action-Reward-State-Action)** ανήκει στην κατηγορία των ενισχυτικών αλγορίθμων μάθησης και αποτελεί μια εξέλιξη του Q-Learning. Αυτός ο αλγόριθμος επιλύει το πρόβλημα ελέγχου του Markov (Markov Control Problem), στοχεύοντας να βρει τη βέλτιστη στρατηγική λήψης αποφάσεων σε ένα περιβάλλον που εκπέμπει σήματα ανταμοιβής με βάση τις επιλογές του πράκτορα [16].

Βασική Ιδέα του SARSA

Η βασική ιδέα του SARSA είναι να μάθει τη συνάρτηση Q που εκτιμά την ποιότητα μιας δράσης σε μια συγκεκριμένη κατάσταση, λαμβάνοντας υπόψη τις δράσεις που ο πράκτορας θα ακολουθήσει και την ανταμοιβή που θα λάβει κάθε φορά που πραγματοποιεί μια δράση. Σε αντίθεση με το Q-Learning, ο αλγόριθμος SARSA ανανεώνει τη συνάρτηση Q ανάλογα με την πραγματική δράση που ακολουθήθηκε από τον πράκτορα [17].

Ο Αλγόριθμος SARSA

Ο αλγόριθμος SARSA βασίζεται στην ενημέρωση των Q-Values με βάση την ακόλουθη εξίσωση:

$$Q(s,a) = Q(s,a) + \alpha * (r + \gamma * Q(s',a') - Q(s,a))$$

Εξίσωση 2

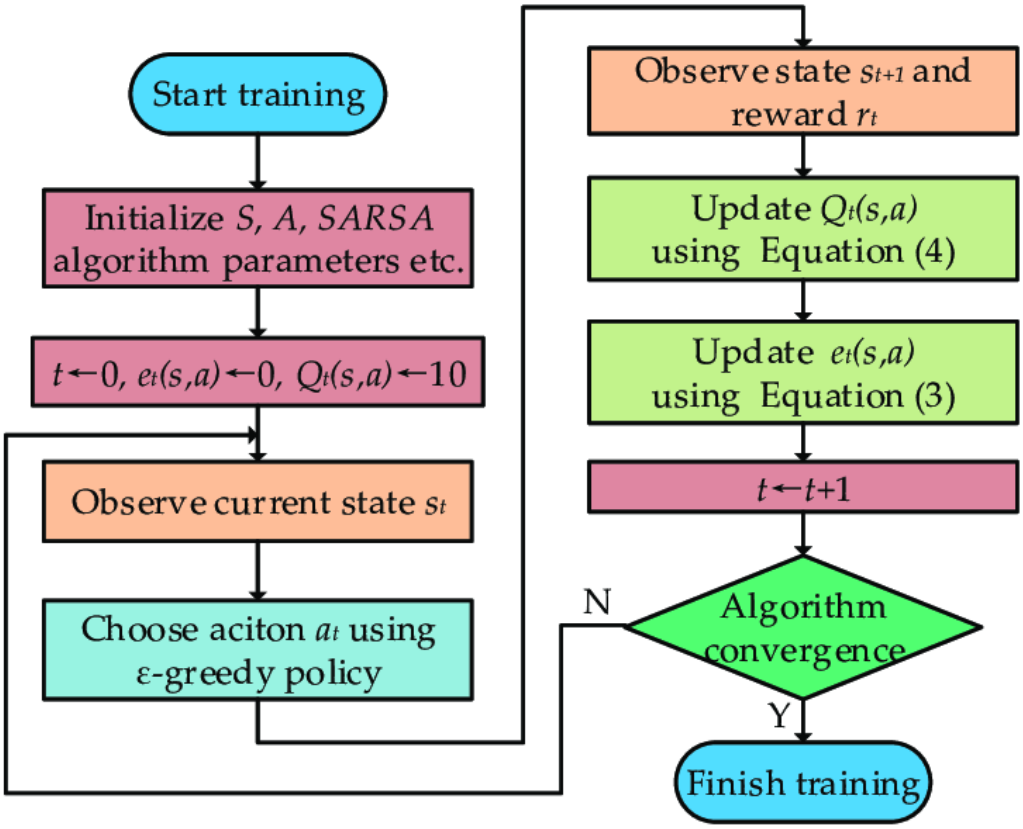
Όπου:

- $Q(s,a)$: Το Q-Value της δράσης "a" στην κατάσταση "s".
- α $(alpha)$: Η εκμάθηση ή ρυθμαπόδοση, που καθορίζει πόσο γρήγορα ενημερώνονται οι Q-Values.
- r: Η ανταμοιβή που λαμβάνει ο πράκτορας κάνοντας τη δράση "a" στην κατάσταση "s".
- γ $(gamma)$: Ο διακριτικός παράγοντας που αντιπροσωπεύει την εκπτωτική στρατηγική.
- $Q(s',a')$: Το Q-Value της επόμενης δράσης "a'" στην επόμενη κατάσταση "s'".

Ο αλγόριθμος SARSA είναι πιο συντηρητικός από το Q-Learning, καθώς προσεγγίζει το Q-Value με βάση τις πραγματικές δράσεις που ακολουθεί ο πράκτορας.

Αυτό το καθιστά κατάλληλο για προβλήματα όπου οι συνέπειες των δράσεων είναι σημαντικές και χρειάζεται προσεκτική εξισορρόπηση της εξερεύνησης και της εκμάθησης [17].

Στην εικόνα παρουσιάζεται ο τρόπος εκπαίδευσης και λειτουργίας του αλγορίθμου SARSA



Εικόνα 3 - *Εκπαίδευση και τρόπος λειτουργίας του αλγορίθμου SARSA* Πηγή: [18]

Εφαρμογές του αλγορίθμου SARSA

Ο SARSA έχει ευρεία εφαρμογή σε πολλούς τομείς. Χρησιμοποιείται σε ρομποτικές εφαρμογές όπου οι δράσεις του ρομπότ επηρεάζουν την κατάσταση του περιβάλλοντος, όπως η ελαχιστοποίηση του χρόνου πλοήγησης και η αποφυγή εμποδίων. Επίσης, μπορεί να χρησιμοποιηθεί για τη βελτιστοποίηση πόρων σε δίκτυα υπολογιστών, σε συστήματα αυτοματισμού και σε αναστολή αποφάσεων.

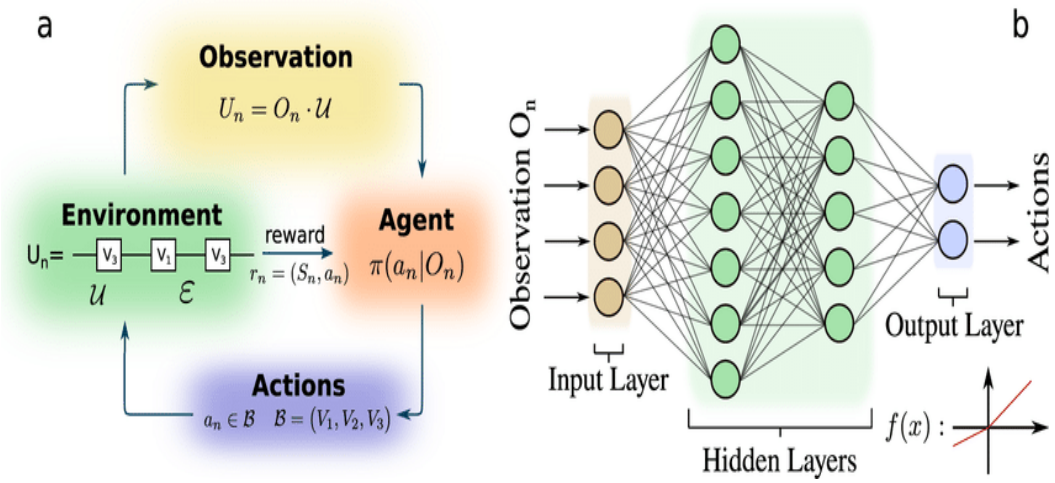
Deep Reinforcement Learning (DRL): Χρησιμοποιεί νευρωνικά δίκτυα για την εκτίμηση της συνάρτησης Q σε περίπλοκα προβλήματα

Οι αλγόριθμοι Reinforcement Learning αντιπροσωπεύουν μια ισχυρή προσέγγιση για την εκπαίδευση υπολογιστικών συστημάτων να προσαρμόζονται σε περιβάλλοντα και να λαμβάνουν αποφάσεις με βάση την αλληλεπίδρασή τους με αυτά. Με τη χρήση των αλγορίθμων αυτών, ο πράκτορας μπορεί να μάθει αυτόνομα τη βέλτιστη στρατηγική προσαρμογής σε μία ποικιλία περιβαλλόντων, ενισχύοντας την αποδοτικότητα και την επίλυση προβλημάτων που παρουσιάζουν υψηλή αβεβαιότητα και πολυπλοκότητα [19].

Η αρχιτεκτονική μάθησης βαθιάς ενίσχυσης (DRL). Το περιβάλλον DRL μπορεί να περιγραφεί ως ένα κβαντικό κύκλωμα μοντελοποιημένο από την προσεγγιστική ακολουθία Un, τη σταθερή ανοχή ε και τον ενιαίο στόχο για την κατά προσέγγιση που αλλάζει γενικά σε κάθε επεισόδιο.

Στην παρακάτω εικόνα παρουσιάζεται ο τρόπος λειτουργίας του αλγορίθμου deep reinforcement learning (DRL) . Σε κάθε χρονικό βήμα n, ο πράκτορας λαμβάνει την τρέχουσα παρατήρηση On και με βάση αυτές τις πληροφορίες, επιλέγει από τη βάση που θα εφαρμοστεί στο κβαντικό κύκλωμα. Επομένως, το περιβάλλον επιστρέφει την πραγματικής αξίας ανταμοιβή rn στον πράκτορα, η οποία είναι συνάρτηση της κατάστασης Sn και της ενέργειας an. β Η πολιτική π του πράκτορα κωδικοποιείται σε ένα βαθύ νευρωνικό δίκτυο [19].

Η πολιτική του πράκτορα κωδικοποιείται σε ένα βαθύ νευρωνικό δίκτυο. Σε κάθε χρονικό βήμα, το DNN λαμβάνει ως είσοδο ένα διάνυσμα που δημιουργείται από τα πραγματικά και τα φανταστικά μέρη της παρατήρησης On. Τέτοιες πληροφορίες επεξεργάζονται από τα κρυφά επίπεδα και επιστρέφονται μέσω του επιπέδου εξόδου. Οι νευρώνες στο επίπεδο εξόδου σχετίζονται με την ενέργεια που θα εκτελέσει ο παράγοντας στο επόμενο χρονικό βήμα. Στην κάτω δεξιά γωνία αναφέρεται ένα παράδειγμα της συνάρτησης μη γραμμικής ενεργοποίησης, δηλαδή η διορθωμένη συνάρτηση γραμμικής μονάδας RELU [20].



Εικόνα 4 - Τ*ρόπος λειτουργίας του deep learning reinforcement* Πηγή: [20]

Νευρωνικά Δίκτυα στο DRL

Η βασική ιδέα του DRL είναι να χρησιμοποιηθούν νευρωνικά δίκτυα για την εκτίμηση της συνάρτησης Q, η οποία αναπαριστά την ποιότητα μιας δράσης σε μια δεδομένη κατάσταση. Τα νευρωνικά δίκτυα μπορούν να εκμεταλλευτούν την ισχύ της βαθιάς μηχανικής μάθησης για να μοντελοποιήσουν πολύπλοκες συναρτήσεις Q και να αντλήσουν πληροφορίες από περίπλοκα περιβάλλοντα.

Προκλήσεις του DRL

Η εφαρμογή του DRL αντιμετωπίζει ορισμένες προκλήσεις λόγω του υψηλού βαθμού πολυπλοκότητας των περιβαλλόντων και της δυσκολίας στο να εκτιμηθούν ακριβώς οι συναρτήσεις Q σε πραγματικό χρόνο. Επίσης, το DRL μπορεί να απαιτεί μεγάλο όγκο δεδομένων για την εκπαίδευση των νευρωνικών δικτύων και την αποφυγή της υπερπροσαρμογής (overfitting) [19][20].

Εφαρμογές του DRL

Το DRL έχει εφαρμογές σε πολλούς τομείς. Στον τομέα της ρομποτικής, μπορεί να χρησιμοποιηθεί για την αυτόνομη πλοήγηση και τον έλεγχο των ρομπότ σε περιβάλλοντα με εμπόδια. Επίσης, μπορεί να εφαρμοστεί σε παιχνίδια όπως τα video games, όπου τα περιβάλλοντα είναι πολύπλοκα και δυναμικά. Άλλες εφαρμογές περιλαμβάνουν την αυτόνομη οδήγηση σε αυτοκίνητα, τον έλεγχο ενεργειακών συστημάτων και την ανάλυση βιολογικών δεδομένων [19].

2 Μεθευρετικοί Αλγόριθμοι

2.1 Μεθευρετικοί Αλγόριθμοι: Βελτίωση της Απόδοσης με Έξυπνες Ευρετικές Προσεγγίσεις

Στον χώρο της μηχανικής μάθησης και της βελτιστοποίησης, οι μεθευρετικοί αλγόριθμοι αποτελούν μια κατηγορία προηγμένων τεχνικών που στοχεύουν στην επίλυση πολύπλοκων προβλημάτων βελτιστοποίησης. Αυτοί οι αλγόριθμοι αναπτύχθηκαν με σκοπό να αντιμετωπίσουν την ανυπέρβλητη πολυπλοκότητα ορισμένων προβλημάτων, όπως η εύρεση του καλύτερου συνδυασμού παραμέτρων για ένα μοντέλο μηχανικής μάθησης. Ολοένα και περισσότερο, στο πεδίο της μηχανικής, οι τεχνικές και οι μέθοδοι βελτιστοποίησης γίνονται ζωτικά εργαλεία [9].

Παρ' όλα αυτά, η χρήση τους απαιτεί συνήθως πολύ χρόνο. Αυτό οφείλεται στην ανάγκη για εκτεταμένη υπολογιστική ισχύ κατά την διαδικασία εξαγωγής λύσεων μέσω αυτών των τεχνικών και βασίζεται επίσης στη φύση των ιδιοτήτων των ίδιων των μεθόδων και αλγορίθμων που χρησιμοποιούνται. Οι μέθοδοι που χρησιμοποιούνται ευρέως σήμερα για την εξαγωγή λύσεων σε προβλήματα βελτιστοποίησης περιλαμβάνουν τις *gradient methods* και τους ευρετικούς αλγορίθμους, με αξιόλογα αποτελέσματα. Ωστόσο, και οι δύο αυτές προσεγγίσεις, πέρα από τα πλεονεκτήματά τους, παρουσιάζουν και ορισμένα μειονεκτήματα.

Τα πλεονεκτήματα της χρήσης *gradient methods* περιλαμβάνουν την ταχεία σύγκλιση προς το κοντινότερο βέλτιστο, αν και όχι πάντα το ολικό. Ωστόσο, όταν χρησιμοποιούνται σε πολυτροπικές συναρτήσεις, συχνά παγιδεύονται σε τοπικά βέλτιστα. Από την άλλη πλευρά, οι ευρετικοί αλγόριθμοι είναι επίσης αποδοτικοί, αλλά επηρεάζονται από τις συνθήκες της αντικειμενικής συνάρτησης. Πρέπει να είναι συνεχής και να έχει θετικά ορισμένο Hessian, ενώ οι επιλογές αρχικοποίησης των υπολογισμών μπορεί να επηρεάσουν τη σύγκλιση και να οδηγήσουν σε τοπικά ακρότατα [9].

Οι μεθευρετικοί αλγόριθμοι προσφέρουν μια ποικιλία από επιλογές για την επίλυση προβλημάτων βελτιστοποίησης και εξαρτώνται από τον τύπο του προβλήματος και τις ειδικές απαιτήσεις του. Μεταξύ αυτών, περιλαμβάνονται οι εξελικτικοί αλγόριθμοι, ο αλγόριθμος προσομοιωμένης ανόπτησης, ο αλγόριθμος σμήνους σωματιδίων και ο αλγόριθμος αναζήτησης μεταβλητής γειτνίασης. Αυτές οι προσεγγίσεις είναι συχνά χρήσιμες για τη διαδικασία εκπαίδευσης νευρωνικών δικτύων και παρέχουν εναλλακτικές

λύσεις σε σχέση με τις παραδοσιακές μεθόδους όπως οι διαδικασίες Back Propagation. Ανάλογα με τον τύπο του προβλήματος και τις απαιτήσεις, επιλέγοντας τον κατάλληλο μεθευρετικό αλγόριθμο μπορεί να βελτιώνεται σημαντικά η απόδοση του βελτιστοποιημένου μοντέλου.

Τελούν επίσης υπό διεύρυνση, με νέες μεθευρετικές μεθόδους που αναπτύσσονται και εφαρμόζονται για την επίλυση πολυπλοκότερων προβλημάτων βελτιστοποίησης. Οι εξελικτικοί αλγόριθμοι, ο αλγόριθμος προσομοιωμένης ανόπτησης, ο αλγόριθμος αναζήτησης μεταβλητής γειτνίασης και πολλοί άλλοι αντιπροσωπεύουν μια δυναμική και συνεχώς αναπτυσσόμενη κατηγορία αλγορίθμων που συμβάλλουν στην βελτίωση της επίλυσης προβλημάτων βελτιστοποίησης και στην αναβάθμιση των επιδόσεων των μηχανικών και ερευνητών σε αυτό το πεδίο.

2.2 Χαρακτηριστικά των Μεθευρετικών Αλγορίθμων

Οι μεθευρετικοί αλγόριθμοι βασίζονται σε έξυπνες ευρετικές προσεγγίσεις που εμπνέονται από φυσικά φαινόμενα, όπως η επιλογή του καλύτερου μονοπατιού από τα μυρμήγκια ή η συμπεριφορά των σωματιδίων σε ένα πεδίο βαρύτητας. Αυτοί οι αλγόριθμοι επιδιώκουν να εξερευνήσουν τον χώρο αναζήτησης των λύσεων με έξυπνο τρόπο, συνδυάζοντας εκτενείς αναζητήσεις με επιλογές που μεταφέρουν την πληροφορία για την κατεύθυνση προς τις βέλτιστες λύσεις.

Τύποι μεθευρετικών Αλγορίθμων

Υπάρχουν πολλοί διαφορετικοί μεθευρετικοί αλγόριθμοι, κάθε ένας από τους οποίους επικεντρώνεται σε διαφορετικούς τρόπους αναζήτησης και βελτιστοποίησης. Οι πιο γνωστοί μεθευρετικοί αλγόριθμοι περιλαμβάνουν [21]:

1. **Γενετικοί Αλγόριθμοι (Genetic Algorithm - GA):** Εμπνευσμένοι από την εξέλιξη των οργανισμών, οι GA εκμεταλλεύονται την έννοια της φυσικής επιλογής για την εύρεση βέλτιστων λύσεων. Αυτοί οι αλγόριθμοι χρησιμοποιούν τυχαίους πληθυσμούς λύσεων και εφαρμόζουν διαδοχικές διαδικασίες επιλογής, διασταύρωσης και μετάλλαξης για τη βελτιστοποίηση των λύσεων.
2. **Αλγόριθμος Σμήνους Σωματιδίων (Particle Swarm Optimization - PSO):** Οι αλγόριθμοι PSO εμμένουν στον τρόπο που τα σωματίδια σε ένα χώρο αναζήτησης αλληλεπιδρούν και συνεργάζονται για την εύρεση των βέλτιστων λύσεων. Κάθε

σωματίδιο αντιπροσωπεύει μια δυνητική λύση και κινείται μέσα στον χώρο αναζήτησης με βάση την επίδοσή του.

3. **Αναζήτηση Μεταβλητής Γειτονιάς (Variable Neighborhood Search - VNS):** Οι αλγόριθμοι VNS εξερευνούν τον χώρο αναζήτησης πραγματοποιώντας διαδοχικές μεταβολές σε κοντινά σημεία. Η διαδικασία αυτή επαναλαμβάνεται με βάση τον βαθμό επίλυσης του προβλήματος.
4. **Αλγόριθμος Προσομοιωμένης Ανόπτησης (Simulated Annealing - SA):** Ο αλγόριθμος Προσομοιωμένης Ανόπτησης, όπως προδίδει και η ονομασία του, προσομοιώνει τη φυσική συμπεριφορά των υλικών στην υψηλή θερμοκρασία και τη σταδιακή μείωσή της.

2.3 Αλγόριθμος Προσομοιωμένης Ανόπτησης (SA)

Ο μεθευρετικός αλγόριθμος της Προσομοιωμένης Ανόπτησης είναι ένας στοχαστικός αλγόριθμους αναζήτησης ολικής αναζήτησης (*global search*). Θα μπορούσε να θεωρηθεί μια πιο στοχαστική έκδοση του Hill-Climbing αλγορίθμου.

Όπως αναφέραμε και παραπάνω ο SA προσομοιώνει τη συμπεριφορά των υλικών στις υψηλές θερμοκρασίες και τη σταδιακή ελάττωσή τους. Αυτά τα χαρακτηριστικά είναι και υπερπαράμετροι του SA.

Υπάρχουν πολλοί τρόποι να γίνει ο υπολογισμός της μείωσης της θερμοκρασίας. Ο πιο διαδεδομένος είναι η «γρήγορη προσομοίωση ανόπτησης» (*fast simulated annealing*). Όπου η νέα θερμοκρασία υπολογίζεται με την ακόλουθη εξίσωση:

$$temperature = \frac{initial\ temperature}{iteration\ number + 1}$$

Εξίσωση 3

Προσθέτουμε μια μονάδα για να αποφύγουμε τη διαίρεση με τον μηδέν. Σε περίπτωση που η επανάληψη ξεκινήσει από το μηδέν. Στη συνέχεια, η διαδικασία αποδοχής χειρότερης λύσης αξιοποιεί και τη θερμοκρασία και τη διαφορά αξιολόγησης της αντικειμενικής συνάρτησης της καλύτερης λύσης με την χειρότερη. Το εύρος των τιμών είναι μεταξύ του 0 και 1, ώστε να μπορέσει να εκφραστεί η πιθανότητα αποδοχής μια χειρότερης τιμής. Η μεταβλητή αυτή ονομάζεται *metropolis acceptance* και εκφράζεται με τον ακόλουθο μαθηματικό τύπο

$$metropolis = \exp\left(-\frac{objective(new) - objective(current)}{temperature}\right)$$

Εξίσωση 4

Με αυτό το τρόπο είναι πιο πιθανόν να γίνει αποδοχή μια χειρότερης τιμής όταν βρίσκεται στην αρχή της αναζήτησης παρά προς το τέλος. Με αποτέλεσμα, να γίνεται αναζήτηση του ολικού βέλτιστου αποδοτικότερα.

2.4 Εφαρμογές Μεθευρετικών Αλγορίθμων

Σε ολόκληρη την διαδικασία της μηχανικής μάθησης, υπάρχουν ανάγκες βελτιστοποίησης. Όπως για παράδειγμα την προετοιμασία των δεδομένων που θα χρησιμοποιηθούν για την εκπαίδευση και την αξιολόγηση του μοντέλου, την εύρεση των πιο αποτελεσματικών υπερ-παραμέτρων (*hyperparameters*) του εκάστοτε αλγορίθμου, ακόμα και την επιλογή του καταλληλότερου μοντέλου, σύμφωνα με τις ανάγκες του προβλήματος. Με αποτέλεσμα η βελτιστοποίηση έχει σημαντικό μέρος στην μηχανική μάθηση.

2.4.1 Βελτιστοποίηση της Προετοιμασίας Δεδομένων

Η προετοιμασία των δεδομένων ορίζεται ως η μετατροπή και το φιλτράρισμα των ανεπεξέργαστων δεδομένων (*raw data*) σε κατάλληλα για τον επιθυμητό αλγόριθμο μηχανικής μάθησης. Η διαδικασία μπορεί να περιλαμβάνει κλιμάκωση των τιμών (*scaling values*), διαχείριση μη διαθέσιμων τιμών (*handling missing values*) και αλλαγή κατανομής πιθανοτήτων των τιμών.

Ολόκληρη η διαδικασία γίνεται χειροκίνητα από ανθρώπους, βάση δοκιμών. Παρόλα αυτά, η μεθοδολογία που θα χρησιμοποιηθεί ώστε να γίνει η προετοιμασία των δεδομένων μπορεί να θεωρηθεί ως ένα πρόβλημα αναζήτησης ή βελτιστοποίησης. Όπου ως εισόδους της συνάρτησης θα είναι μια σειρά από μετατροπές των δεδομένων, όπου θα εφαρμοστούν στα δεδομένα εκπαίδευσης (*training data*), και το πρόβλημα που χρίζει βελτιστοποίηση ολικής αναζήτησης (*global search*). [22]

Σύμφωνα με τη μελέτη [23] γίνεται η εφαρμογή μίας υβριδικής μεθόδου, με σκοπό την βελτίωση της επίδοσης του μοντέλου, μειώνοντας τα δεδομένα που δυσκολεύουν την διαδικασία της εκπαίδευσης. Η υβριδική μέθοδος, συνδυάζει και προσπαθεί να

χρησιμοποιήσει τα θετικά χαρακτηριστικά γνωστών αλγορίθμων στη μηχανική μάθηση. Οι αλγόριθμοι είναι ο KNN, Bayesian μέθοδος και Γενετικούς αλγορίθμους. Το μοντέλο εκπαιδεύεται σε 5 διαφορετικές βάσεις δεδομένων (datasets) του UCI machine learning datasets.

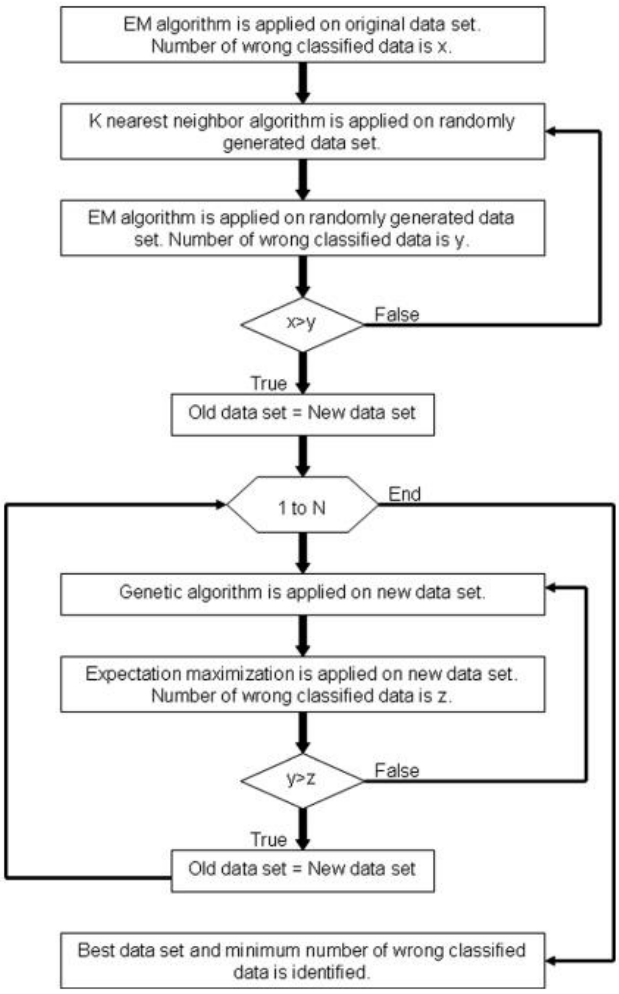
Η διαδικασία που ακολουθεί σε αυτήν την υβριδική διαδικασία είναι εξής:

1. Αρχικά, εφαρμόζεται ο αλγόριθμος EM (Expectation maximization algorithm), όπου είναι βασισμένος στην Bayesian μέθοδο, πάνω στα επιλεγμένα σύνολα δεδομένων. Όλα τα δεδομένα από το επιλεγμένο σύνολο χρησιμοποιούνται ως δεδομένα δοκιμής. Στη συνέχεια, υπολογίζεται ο αριθμός των λανθασμένα ταξινομημένων δεδομένων. Ας υποθέσουμε ότι αυτός ο αριθμός είναι x.
2. Επόμενο βήμα, να δημιουργηθεί ένα νέο σύνολο δεδομένων τυχαία μεταξύ των μέγιστων και ελάχιστων τιμών κάθε κλάσης του αρχικού συνόλου δεδομένων. Ο αριθμός των στοιχείων του νέου συνόλου δεδομένων είναι διπλάσιος σε σχέση με το αρχικό.
3. Εφαρμόζεται η μέθοδος KNN. Για κάθε δεδομένο, υπολογίζονται οι αποστάσεις προς το μέσο όρο κάθε κλάσης. Τα δεδομένα ταξινομούνται ανάλογα με τις αποστάσεις και επιλέγεται το πλησιέστερο μισό τους. Έτσι, η τιμή k επιλέγεται ως το πενήντα τοις εκατό του πλήθους των δεδομένων που δημιουργήθηκαν.
4. Το πρώτο βήμα επαναλαμβάνεται σε αυτό το στάδιο. Τα δεδομένα δοκιμής είναι πάλι το αρχικό σύνολο δεδομένων, ωστόσο, τα δεδομένα εκπαίδευσης είναι το δημιουργημένο σύνολο δεδομένων. Εφαρμόζεται ο αλγόριθμος EM και υπολογίζεται ο αριθμός των λανθασμένα ταξινομημένων δεδομένων. Αυτός ο αριθμός ονομάζεται y.
5. Στη συνέχεια, οι τιμές x και y συγκρίνονται και, εάν x είναι μικρότερο ή ίσο με y, τότε το νέο δημιουργημένο σύνολο δεδομένων δεν θα είναι καλύτερο από το αρχικό. Έτσι, τα βήματα 2-4 θα πρέπει να επαναληφθούν. Αυτός ο βρόχος θα συνεχίσει μέχρι το y να γίνει μικρότερο από το x. Όταν συμβεί αυτό, σημαίνει ότι το δημιουργημένο σύνολο δεδομένων είναι καλύτερο για εκπαίδευση από το αρχικό.
6. Από αυτό το βήμα, ξεκινά ένας νέος βρόχος και ο αριθμός του βρόχου εξαρτάται από τον χρήστη, ανάλογα με τα χαρακτηριστικά των τιμών του συνόλου δεδομένων. Εφαρμόζεται ο γενετικός αλγόριθμος στο τελευταίο δημιουργημένο σύνολο δεδομένων. Τα δεδομένα ταξινομούνται ξανά ανάλογα με τις αποστάσεις

τους και τον καθορισμένο λόγο διασταυρώσεως. Τα χειρότερα δεδομένα τίθενται στη διαδικασία διασταυρώσεως μεταξύ τους και δημιουργείται ένα νέο σύνολο δεδομένων.

7. Το Βήμα 1 εφαρμόζεται ξανά στο νέο σύνολο δεδομένων και υπολογίζεται ο αριθμός των λανθασμένα ταξινομημένων δεδομένων, ο οποίος ονομάζεται z. Το z συγκρίνεται με το y και, εάν το z είναι μικρότερο από το y, το νέο σύνολο δεδομένων θα είναι καλύτερο από το παλιό. Ο βρόχος συνεχίζεται με το νέο σύνολο δεδομένων. Διαφορετικά συνεχίζει ο γενετικός αλγόριθμος για καλύτερα σύνολα δεδομένων.
8. Αυτό είναι το τελευταίο βήμα και μετά την ολοκλήρωση του βρόχου, το καλύτερο σύνολο δεδομένων που βρέθηκε και τον ελάχιστο αριθμό λανθασμένα ταξινομημένων δεδομένων αποθηκεύονται.

Γίνονται ακόμα πιο διακριτά τα βήματα με το διάγραμμα ροής στην εικόνα 5



Εικόνα 5 – Διάγραμμα ροής Υβριδικού Αλγορίθμου

Το συμπέρασμα του έργου ήταν, οι συγγραφείς παρουσίασαν μια υβριδική μέθοδο ταξινόμησης που συνδυάζει τον αλγόριθμο KNN, τις Bayesian μεθόδους και τον γενετικό αλγόριθμο. Η κεντρική ιδέα είναι η δημιουργία ενός νέου βελτιστοποιημένου συνόλου δεδομένων για ταξινόμηση. Όταν δοκιμάστηκε η μέθοδος σε διαφορετικά σύνολα δεδομένων, όπως το iris και το καρκίνο μαστού, διαπιστώθηκε ότι επιτεύχθηκε καλύτερη απόδοση σε σχέση με κάποιες κλασικές μεθόδους ταξινόμησης.

2.4.2 Βελτιστοποίηση των Υπερ-παραμέτρων

Οι υπερ-παράμετροι δίνουν τη δυνατότητα στους αλγορίθμους μηχανικής μάθησης να ρυθμιστεί ο τρόπος που θα προσαρμοστούν οι αλγόριθμοι στα εκάστοτε δεδομένα. Η διαδικασία της εύρεσης των βέλτιστων υπερ-παραμέτρων ονομάζεται «βελτιστοποίηση υπερ-παραμέτρων» (*Hyperparameter tuning or Hyperparameter optimization*).

Η διαδικασία της βελτιστοποίησης υπερ-παραμέτρων είναι σημαντική, διότι ενώ γνωρίζουμε τη δυναμική των υπερ-παραμέτρων στους αλγορίθμους, δεν μπορούμε να γνωρίζουμε πως θα επηρεάσουν την επίδοση του μοντέλου στα ζητούμενα δεδομένα. Για να μπορέσει να βρεθεί η βέλτιστη υπερ-παράμετρο, ως εισόδους της συνάρτησης ορίζουμε τις υπερ-παραμέτρους του αλγορίθμου και το πρόβλημα που χρειάζεται βελτιστοποίηση. [22]

Στη δημοσίευση [24], ερευνάται μια καινοτόμα προσέγγιση για το πως μπορεί να γίνει βελτιστοποίηση των υπερ-παραμέτρων C και γ, του SVM αλγορίθμου μηχανικής μάθησης με τη βοήθεια γενετικού αλγορίθμου. Επιπλέον ερευνάτε πως μπορεί να γίνει μείωση της διάστασης των χαρακτηριστικών (features reduction) με τη χρήση του PCA, συγκεκριμένα στη βάση δεδομένων Iris.

Στην ενότητα αυτή θα αναπτυχθεί η χρήση του γενετικού αλγορίθμου για τη βελτίωση των υπερ-παραμέτρων του SVM, όπου είναι και το ζητούμενο της ενότητας.

Το SVM αποτελεί μια δημοφιλή τεχνική της μηχανικής μάθησης, η οποία εφαρμόζεται ευρέως σε προβλήματα ταξινόμησης, παλινδρόμησης και ομαδοποίησης, χρησιμοποιώντας επίβλεψη για την εκπαίδευση. Έχει εφαρμοστεί με επιτυχία σε ποικίλους τομείς, όπως η έκφραση γονιδίων και η εξόρυξη κειμένου από το διαδίκτυο.

Η επίτευξη υψηλής ακρίβειας και αξιοπιστίας στην ταξινόμηση με το SVM βασίζεται στην αναζήτηση των ιδανικών παραμέτρων, καθώς αυτές διαδραματίζουν κρίσιμο ρόλο. Η απόδοση της ταξινόμησης μπορεί να υποστεί διακυμάνσεις λόγω μη βέλτιστων ρυθμίσεων παραμέτρων.

Στην προσέγγιση αυτή, προτάθηκε μια νέα μέθοδος για τη βελτιστοποίηση των παραμέτρων του SVM, χρησιμοποιώντας γενετικό αλγόριθμο με πραγματική αξία (Nested Real Valued Genetic Algorithm - NRGA). Αυτή η προσέγγιση αποδείχθηκε αποτελεσματική στη μείωση του χρόνου βελτιστοποίησης και του κόστους υπολογισμών, συγκριτικά με συμβατικές μεθόδους βελτιστοποίησης που εξετάζουν όλες τις παραμέτρους μαζί. Επιπλέον, η προσέγγιση SVM+NRGA επιτυγχάνει αυξημένη ακρίβεια ταξινόμησης, καθιστώντας την προτιμητέα σε σχέση με άλλες προσεγγίσεις.

Τα βήματα που προτείνονται στην έρευνα για την ταξινόμηση των δεδομένων του Iris με βάση τη βελτιστοποίηση των υπερ-παραμέτρων του SVM είναι τα εξής:

1. Αρχικά, έχουμε τη βάση δεδομένων του Iris σε μορφή CSV.

2. Διαχωρίζουμε τα δεδομένα σε ελέγχου (test dataset) και εκπαίδευσης (train dataset). Με ποσοστό 70% τα δεδομένα εκπαίδευσης και 30% τα δεδομένα ελέγχου.
3. Χωρίζουμε τα δεδομένα εκπαίδευσης ανάλογα με τις κλάσεις τους
4. Υπολογίζουμε την τυπική απόκλιση και τη μέση τιμή για κάθε περίπτωση δεδομένων ανάλογα με τις τιμές των κλάσεων
5. Επιλέγουμε τις παραμέτρους του SVM (C και γ) ως είσοδο για τη βελτιστοποίηση μέσω γενετικού αλγορίθμου.
6. Εφαρμόζουμε τις βέλτιστες τιμές των παραμέτρων (C και γ) ως αρχική τιμή στη διαδικασία ταξινόμησης με χρήση του SVM.
7. Χρησιμοποιούμε το μοντέλο και παράγουμε προβλέψεις.
8. Υπολογίζουμε την ακρίβεια των προβλέψεων συγκρίνοντας τα με τα δεδομένα ελέγχου. Η ακρίβεια αξιολογείται με εύρος μεταξύ 0% - 100%.

Τα αποτελέσματα της μελέτης, φανερώνουν πως η ακρίβεια των προβλέψεων του SVM όταν εφαρμόζεται ο GA στην βελτιστοποίηση των υπερ-παραμέτρων C και γ είναι 98.7%, ενώ όταν δεν εφαρμόζεται η βελτιστοποίηση το ποσοστό μειώνεται στο 97.78%.

2.4.3 Βελτιστοποίηση της Επιλογής Μοντέλου

Όπως και στις προηγούμενες ενότητες, έτσι και σε αυτή την ενότητα η επιλογή των μοντέλων γίνεται χειροκίνητα. Η διαδικασία στην ουσία περιλαμβάνει την επιλογή του αλγόριθμου μηχανικής μάθησης ή τα machine learning pipelines που παράγουν τα μοντέλα.

Ως machine learning pipeline ορίζεται μια σειρά από αυτοματοποιημένα βήματα, ολόκληρης της διαδικασίας. Η διαδικασίας περιέχει την εκτίμηση των δεδομένων, την προετοιμασία των δεδομένων, την εκπαίδευση του μοντέλου, την ανάλυση και την ανάπτυξη (*deployment*)

Ώστε να μπορέσει να πλαισιωθεί ως πρόβλημα βελτιστοποίησης ολόκληρο το project ή να ενσωματωθεί ένα μέρος του, θα πρέπει ως εισόδους της συνάρτησης να οριστούν η μετατροπή των δεδομένων, ο αλγόριθμος μηχανικής μάθησης και οι υπερ-παράμετροι του αλγορίθμου. Στην ουσία έχουμε αυτοματοποιημένους αλγορίθμους μηχανικής μάθησης (*AutoML*). [22]

Μια από τις μελέτες που έχουν γίνει για το AutoML είναι η [25], όπου παρουσιάζεται η μεθοδολογία TPOT (Tree-based Pipeline Optimization Tool) v0.3. Η TPOT μεθοδολογία είναι ανοιχτού κώδικα και βασίζεται στους γενετικούς αλγορίθμου, όπου βελτιστοποιείται η διαδικασία της προετοιμασίας δεδομένων και η επιλογή του κατάλληλου μοντέλου, με σκοπό την καλύτερη επίδοση.

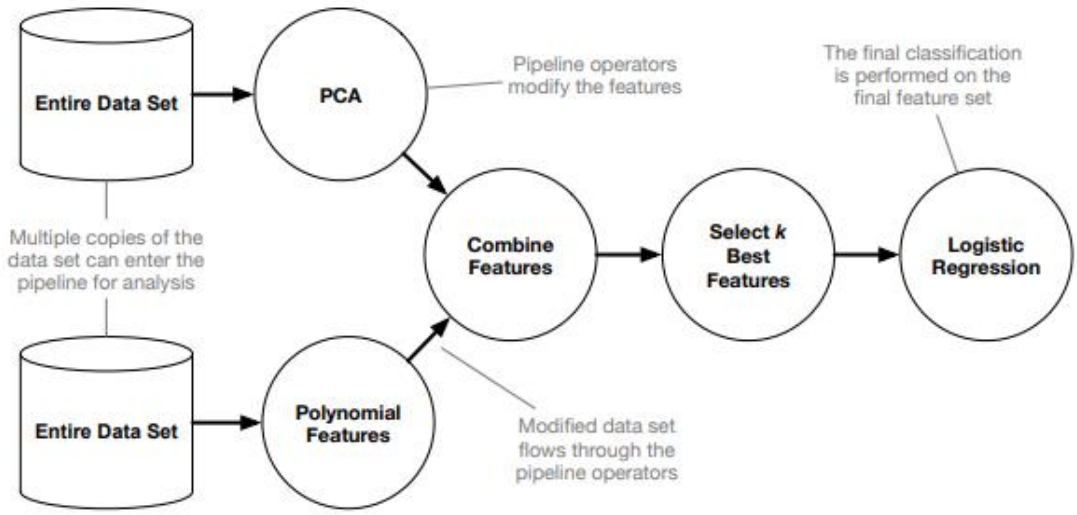
Η διαδικασία της μεθόδου είναι διακρίνεται στα εξής βήματα:

Machine Learning Pipeline Operators

Όπου το TPOT είναι ένας wrapper των πακέτων μηχανικής μάθησης της Python. Έτσι κάθε machine learning pipeline operator είναι ένας αλγόριθμος μηχανικής μάθησης, ή μια προ επεξεργασία δεδομένων. Πιο συγκεκριμένα υπάρχουν τριών ειδών operators. Οι Supervised Classification Operators, όπου είναι οι αλγόριθμοι μηχανικής μάθησης. Όπως για παράδειγμα ο KNN. Το επόμενο είδος είναι οι Feature Preprocessing Operators, όπου είναι οι αλγόριθμοι για την επεξεργασία των δεδομένων. Για παράδειγμα η συνάρτηση StandartScaler. Το τελευταίο είδος είναι οι Feature Selection Operators , όπου είναι αλγόριθμοι για την μείωση των δεδομένων.

Δόμηση Tree-based Pipelines

Το TPOT αναπαριστά τα pipelines ως δεντρικές δομές, εφαρμόζοντας βασικές αρχές των Γενετικών Αλγορίθμων. Στην εικόνα 6 φαίνεται η δεντρική δομή του παραδείγματος, όπου δύο αντίγραφα των δεδομένων συνόλου έχουν τροφοδοτηθεί σε ένα pipeline, όπου επεξεργάζονται από κάθε operator επιτυχώς, συνδυάζονται ως ένα σύνολο δεδομένων και τέλος γίνεται η κατηγοριοποίηση.



Εικόνα 6 – Παράδειγμα δεντρικής δομής pipeline TPOT

Βελτιστοποίηση των Pipelines

Το TPOT χρησιμοποιεί γενετικούς αλγορίθμους για την εύρεση των βέλτιστων pipelines. Δηλαδή κατά τη διαδικασία, διαφορετικά pipelines "ανταγωνίζονται" μεταξύ τους για να επιτύχουν την καλύτερη απόδοση στα δεδομένα. Τα pipelines που αποδίδουν καλύτερα "επιλέγονται" για "διασταύρωση", δημιουργώντας νέα, βελτιωμένα pipelines για την επόμενη γενιά.

Το συμπέρασμα της έρευνας οδήγησε στο ότι το TPOT αποτελεί ένα ιδιαίτερα χρήσιμο εργαλείο για την αυτοματοποίηση και βελτιστοποίηση της επιλογής μοντέλων και προεπεξεργασίας των δεδομένων στον τομέα της μηχανικής μάθησης

3 Βασικές αρχές κατηγοριοποίησης και εξόρυξης δεδομένων

3.1 Τεχνικές- Μέθοδοι- αλγόριθμοι κατηγοριοποίησης

Στο κεφάλαιο αυτό αναλύεται η φιλοσοφία και η λειτουργία των πιο γνωστών αλγορίθμων μηχανικής μάθησης. Οι περισσότεροι από αυτούς του αλγορίθμους υλοποιούνται και ως αλγόριθμοι μάθησης χωρίς επίβλεψη. Παρ' όλα αυτά η εργασία εστιάζει στους αλγορίθμους εποπτευόμενης μάθησης και πιο συγκεκριμένα στα προβλήματα κατηγοριοποίησης.

Οι αλγόριθμοι κατηγοριοποίησης μπορούν να βασίζονται σε διάφορες μεθόδους, όπως οι SVM, k-NN, Decision Trees, Naive Bayes κ.ά. Κάθε αλγόριθμος έχει τις δικές του δυνατότητες και περιορισμούς, και επιλέγεται ανάλογα με τα χαρακτηριστικά των δεδομένων και τον στόχο της κατηγοριοποίησης [23].

3.2 SVM

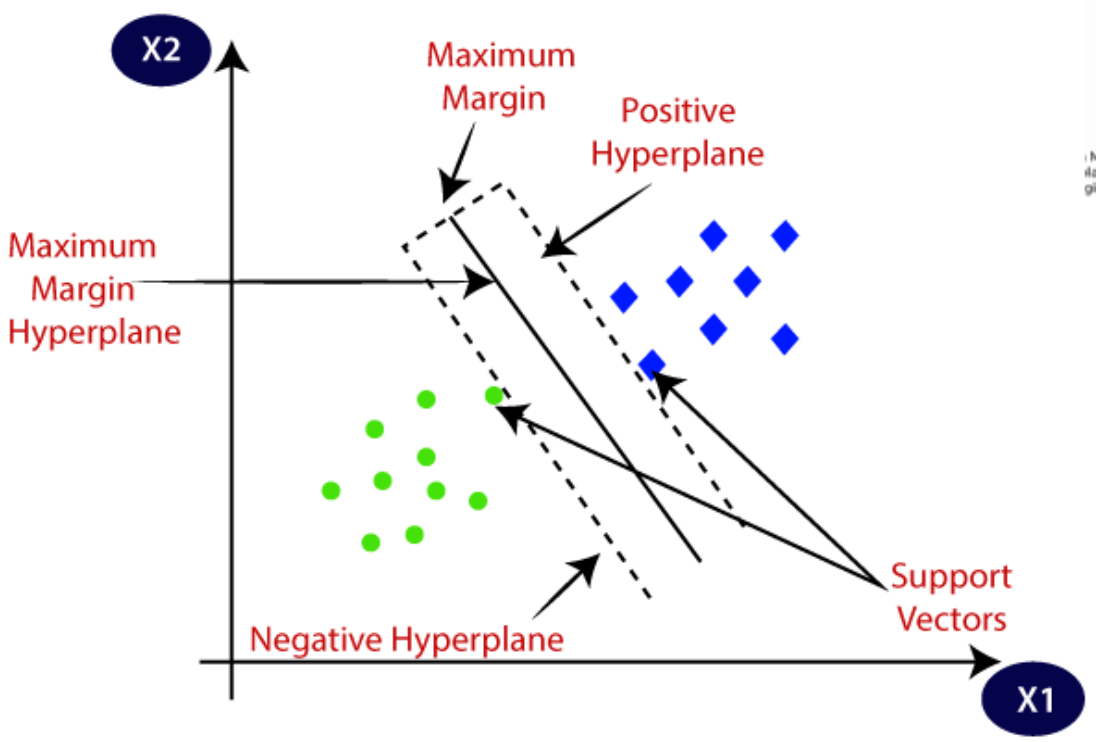
Οι Ταξινομητές Διανυσματικής Μηχανής (Support Vector Machines - SVM) είναι μία από τις πιο δημοφιλείς τεχνικές μηχανικής μάθησης που χρησιμοποιούνται για προβλήματα κατηγοριοποίησης και παλινδρόμησης. Αυτοί οι αλγόριθμοι ανήκουν στην κατηγορία των επιβλεπόμενων μοντέλων μηχανικής μάθησης, δηλαδή απαιτούν ετικετοποιημένα δεδομένα εκπαίδευσης για να εκπαιδευτούν [23].

Λειτουργία των SVM

Η ιδέα πίσω από τους SVM βασίζεται στον γεωμετρικό χώρο. Ο SVM προσπαθεί να βρει ένα υπερεπίπεδο (hyperplane) που διαχωρίζει τις διαφορετικές κατηγορίες δεδομένων με τέτοιο τρόπο, ώστε να μεγιστοποιεί την απόσταση από τα πλησιέστερα σημεία δεδομένων των διαφορετικών κατηγοριών. Αυτά τα πλησιέστερα σημεία ονομάζονται "διανύσματα υποστήριξης" (support vectors), και αποτελούν τα κλειδιά σημεία που ορίζουν το υπερεπίπεδο.

Η υπόθεση των SVM είναι ότι όσο μεγαλύτερη η απόσταση ανάμεσα στα διανύσματα υποστήριξης, τόσο πιο γενικοποιημένο και ακριβές είναι το μοντέλο στην κατηγοριοποίηση νέων δεδομένων. Αυτή η απόσταση ονομάζεται "μέγιστο περιθώριο"

(maximum margin), και οι SVM στοχεύουν στο να βρουν το υπερεπίπεδο που το μεγιστοποιεί (εικόνα 7).



Εικόνα 7 - Ο στόχος του αλγόριθμου SVM είναι να δημιουργήσει την καλύτερη γραμμή ή όριο απόφασης που μπορεί να διαχωρίσει το χώρο n-διαστάσεων σε κλάσεις, ώστε να μπορούμε εύκολα να βάλουμε το νέο σημείο δεδομένων στη σωστή κατηγορία στο μέλλον. Αυτό το όριο καλύτερης απόφασης ονομάζεται υπερεπίπεδο. Πηγή: [23]

Τα δεδομένα συχνά μπορεί να μην είναι γραμμικά διαχωρίσιμα, δηλαδή δεν μπορούν να χωριστούν από ένα απλό υπερεπίπεδο. Για τέτοιες περιπτώσεις, οι SVM χρησιμοποιούν την τεχνική του πυρήνα (kernel trick). Οι πυρήνες είναι συναρτήσεις που μετασχηματίζουν τον χώρο των χαρακτηριστικών σε έναν υψηλότερης διάστασης χώρο, όπου τα δεδομένα μπορούν να γίνουν γραμμικά διαχωρίσιμα. Η προβολή των δεδομένων στο νέο αυτό χώρο επιτρέπει στο SVM να βρει ένα υπερεπίπεδο που τα διαχωρίζει [23].

Πλεονεκτήματα των SVM:

1. Αποτελεσματική για μεγάλα σύνολα δεδομένων: Οι SVM έχουν καλή απόδοση σε μεγάλα σύνολα δεδομένων, καθώς είναι αποτελεσματικοί και μπορούν να χειριστούν υψηλές διαστάσεις δεδομένων.
2. Καλή γενίκευση: Η χρήση του μεγίστου περιθωρίου βοηθά στην καλή γενίκευση του μοντέλου και την αποφυγή υπερεκπαίδευσης.

3. Πυρήνας SVM: Η δυνατότητα χρήσης πυρήνας επιτρέπει την αντιμετώπιση μη γραμμικών προβλημάτων κατηγοριοποίησης.

Εφαρμογές των SVM

Οι SVM έχουν εφαρμογές σε ποικίλους τομείς, όπως:

- Κατηγοριοποίηση εικόνων για αναγνώριση αντικειμένων.
- Ανίχνευση μοτίβων σε βιολογικά δεδομένα.
- Πρόβλεψη τιμών ακινήτων και χρηματοοικονομικών αγορών.
- Ιατρική διάγνωση και πρόβλεψη νόσων.

Από τα παραπάνω προκύπτει ότι οι Ταξινομητές Διανυσματικής Μηχανής είναι μια ισχυρή τεχνική μηχανικής μάθησης που μπορεί να επιλύσει πολύπλοκα προβλήματα κατηγοριοποίησης. Η ικανότητα τους να επιχειρηματολογούν βάσει του μεγίστου περιθωρίου και η χρήση πυρήνας τους τους κάνουν ευέλικτους και αποτελεσματικούς για ποικίλες εφαρμογές. Με την συνεχή εξέλιξη της τεχνολογίας, οι SVM αναμένεται να συνεχίσουν να είναι ένα σημαντικό εργαλείο για την εξόρυξη γνώσης από δεδομένα σε διάφορους τομείς.

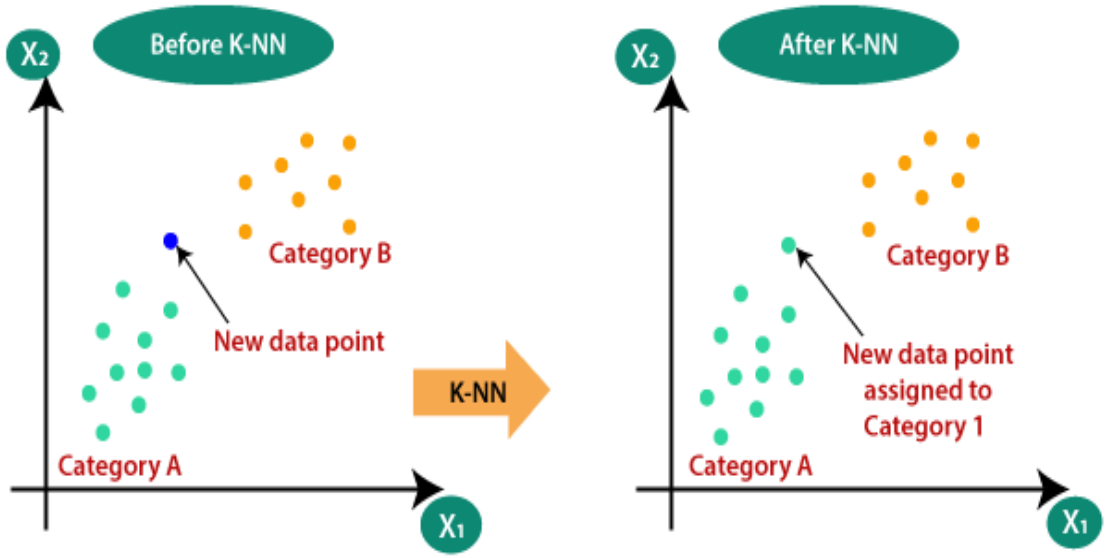
3.3 k-NN

Ο k-Πλησιέστερων Γειτόνων (k-Nearest Neighbors - k-NN) είναι μια απλή και δημοφιλής μέθοδος ταξινόμησης και παλινδρόμησης στο πεδίο της μηχανικής μάθησης. Η k-NN είναι ένας αλγόριθμος επιβλεπόμενης μάθησης, που σημαίνει ότι απαιτεί ετικετοποιημένα δεδομένα εκπαίδευσης για να μάθει το μοντέλο του [26].

Λειτουργία των k-NN

Η λειτουργία του k-NN είναι απλή. Για να ταξινομήσει ένα νέο δείγμα, ο αλγόριθμος υπολογίζει την απόσταση του από τα υπάρχοντα δείγματα του συνόλου εκπαίδευσης. Στη συνέχεια, επιλέγει τα k κοντινότερα δείγματα (τα k-πλησιέστερα γείτονες) και τους αναθέτει την πιο συχνή ετικέτα (στην περίπτωση της ταξινόμησης) ή υπολογίζει την μέση τιμή (στην περίπτωση της παλινδρόμησης) των ετικετοποιημένων δειγμάτων για να προβλέψει την κατηγορία ή την τιμή του νέου δείγματος. Για παράδειγμα, ας υποθέσουμε ότι υπάρχουν δύο κατηγορίες, δηλαδή η Κατηγορία Α και η Κατηγορία Β, και έχουμε ένα νέο σημείο δεδομένων x1, οπότε αυτό το σημείο δεδομένων θα βρίσκεται σε ποια από αυτές τις κατηγορίες (εικόνα 8). Για να λύσουμε αυτό το είδος

προβλήματος, χρειαζόμαστε έναν αλγόριθμο K-NN. Με τη βοήθεια του K-NN, μπορούμε εύκολα να αναγνωρίσουμε την κατηγορία ή την κλάση ενός συγκεκριμένου συνόλου δεδομένων [26].



Εικόνα 8 - Παράδειγμα εφαρμογής του αλγορίθμου k-NN.

Η παράμετρος k αντιπροσωπεύει τον αριθμό των γειτόνων που επιλέγονται. Η επιλογή του κατάλληλου k είναι σημαντική και μπορεί να επηρεάσει την απόδοση του αλγορίθμου. Ένα μικρό k μπορεί να οδηγήσει σε πολύ ανεπαρκή προσαρμογή (overfitting), ενώ ένα μεγάλο k μπορεί να οδηγήσει σε πολύ απλοποιημένο μοντέλο (underfitting) [27].

Πλεονεκτήματα και Μειονεκτήματα των k-NN

Τα σημαντικότερα πλεονεκτήματα της τεχνικής k-NN είναι [27]:

1. Απλός και εύκολος στην υλοποίηση: Ο k-NN είναι ένας από τους πιο εύκολους αλγορίθμους που υπάρχουν και μπορεί να υλοποιηθεί σχετικά γρήγορα.
2. Δεν απαιτεί εκπαίδευση: Ο αλγόριθμος δεν απαιτεί κάποιο κόστος εκπαίδευσης για τη δημιουργία του μοντέλου, καθώς αποθηκεύει τα ετικετοποιημένα δεδομένα εκπαίδευσης και πραγματοποιεί τις προβλέψεις απευθείας.
3. Καλή απόδοση για απλά προβλήματα: Όταν τα δεδομένα είναι απλά και οι κατηγορίες διαχωρίζονται καθαρά, ο k-NN μπορεί να παρουσιάσει καλή απόδοση.

Βέβαια, υπάρχουν και ορισμένα σημαντικά μειονεκτήματα της μεθόδου k-NN, μερικά από τα οποία είναι [27]:

1. Υπολογιστική πολυπλοκότητα: Κατά την πρόβλεψη για ένα νέο δείγμα, ο αλγόριθμος πρέπει να υπολογίσει τις αποστάσεις από όλα τα ετικετοποιημένα δείγματα του συνόλου εκπαίδευσης, οπότε η υπολογιστική πολυπλοκότητα αυξάνει όσο μεγαλώνει το σύνολο δεδομένων.
2. Αποτελέσματα ευαίσθητα στην κλίμακα: Ο k-NN μπορεί να είναι ευαίσθητος στην κλίμακα των χαρακτηριστικών, οπότε η κανονικοποίηση των δεδομένων μπορεί να είναι απαραίτητη.

Ο αλγόριθμος k-NN έχει πολλές εφαρμογές, όπως:

- Κατηγοριοποίηση εικόνων και αναγνώριση προτύπων.
- Συστήματα συστάσεων και φιλτράρισμα πληροφορίας.
- Ιατρική διάγνωση και πρόβλεψη νόσων.
- Πρόβλεψη τιμών στις χρηματοοικονομικές αγορές.

Η μέθοδος των k-NN αποτελεί μια απλή και αποτελεσματική τεχνική για ταξινόμηση και παλινδρόμηση. Παρότι έχει κάποια μειονεκτήματα, όπως την υπολογιστική πολυπλοκότητα, η k-NN παραμένει μια από τις πρώτες επιλογές για πολλές εφαρμογές, ειδικά όταν τα δεδομένα είναι απλά και διαχωρίζονται καθαρά [27].

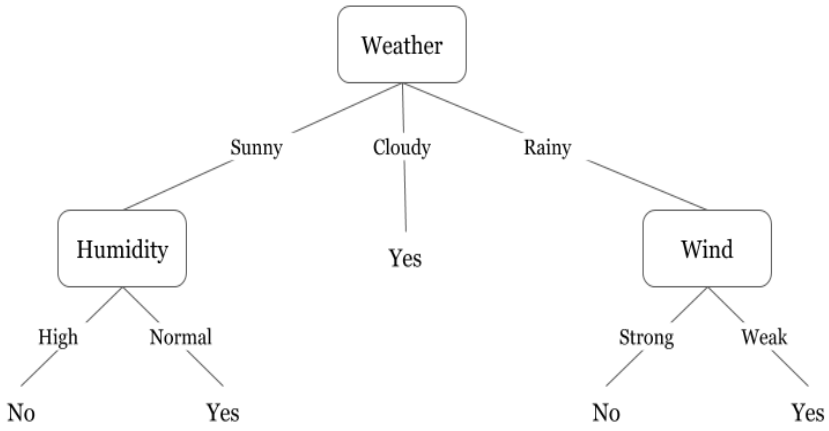
3.4 Decision Trees

Τα Δέντρα Απόφασης (Decision Trees) είναι μια αποτελεσματική και δημοφιλής μέθοδος της μηχανικής μάθησης που χρησιμοποιείται για κατηγοριοποίηση και παλινδρόμηση. Ανήκουν στην κατηγορία των αλγορίθμων επιβλεπόμενης μάθησης, καθώς απαιτούν ετικετοποιημένα δεδομένα εκπαίδευσης για να μάθουν το μοντέλο τους.

Τα Δέντρα Απόφασης είναι ιεραρχικές δομές που χρησιμοποιούνται για να λάβουν αποφάσεις με βάση τις τιμές των χαρακτηριστικών. Ξεκινούν από έναν ριζικό κόμβο (root node) που περιέχει το σύνολο των δεδομένων εκπαίδευσης και κάθε εσωτερικό κόμβο (internal node) αντιπροσωπεύει μια δοκιμασία για ένα χαρακτηριστικό. Οι κλάδοι των δέντρων αντιπροσωπεύουν τις πιθανές τιμές αυτού του χαρακτηριστικού, ενώ τα φύλλα (leaf nodes) περιέχουν τις τελικές κατηγορίες ή προβλέψεις [28].

Η διαδικασία της δημιουργίας ενός δέντρου απόφασης περιλαμβάνει την επιλογή του καλύτερου χαρακτηριστικού για κάθε κόμβο, με βάση κάποιο κριτήριο ομοιογένειας, όπως η πληροφοριακή κέρδους ή η μείωση της ανάμιξης (gini impurity). Η διαδικασία συνεχίζεται αναδρομικά για κάθε υποδέντρο μέχρι να επιτευχθεί κάποιο κριτήριο τερματισμού, όπως η επίτευξη ενός συγκεκριμένου βάθους ή ο μικρότερος αριθμός δειγμάτων σε ένα φύλλο [28].

Ένα παράδειγμα εφαρμογής των δένδρων αποφάσεων φαίνεται στην εικόνα 9 η οποία απεικονίζει ένα δέντρο μαθημένης απόφασης. Μπορούμε να δούμε ότι κάθε κόμβος αντιπροσωπεύει ένα χαρακτηριστικό και ο κλάδος από κάθε κόμβο αντιπροσωπεύει το αποτέλεσμα αυτού του κόμβου. Τέλος, είναι τα φύλλα του δέντρου όπου λαμβάνεται η τελική απόφαση. Εάν τα χαρακτηριστικά είναι συνεχή, οι εσωτερικοί κόμβοι μπορούν να δοκιμάσουν την τιμή ενός χαρακτηριστικού έναντι ενός ορίου.



Εικόνα 9 - Παράδειγμα χρήσης των δένδρων αποφάσεων

Ρόλος των Δέντρων Απόφασης στην Εξόρυξη Δεδομένων και την Κατηγοριοποίηση

Τα Δέντρα Απόφασης είναι πολύ ισχυρά εργαλεία στην εξόρυξη δεδομένων και την κατηγοριοποίηση, καθώς παρέχουν απλά, ερμηνεύσιμα και αποτελεσματικά μοντέλα. Ο ρόλος τους περιλαμβάνει:

1. Ταξινόμηση: Χρησιμοποιούνται για την κατηγοριοποίηση δεδομένων σε διάφορες κατηγορίες ή κλάσεις, όπως πρόβλεψη ασθενειών, αναγνώριση ψηφίων σε εικόνες, κλπ.
2. Παλινδρόμηση: Μπορούν επίσης να χρησιμοποιηθούν για πρόβλεψη συνεχών τιμών, όπως πρόβλεψη τιμών ακινήτων ή ενέργειας.
3. Επεξεργασία φυσικής γλώσσας: Μπορούν να χρησιμοποιηθούν για τον καθορισμό της κατηγορίας μιας πρότασης ή ενός κειμένου, όπως στην ανάλυση συναισθημάτων.

Το κύριο πλεονέκτημα των Δέντρων Απόφασης είναι η εύκολη ερμήνευσή τους, καθώς μπορούμε εύκολα να κατανοήσουμε πώς ο αλγόριθμος λαμβάνει τις αποφάσεις του, εξετάζοντας τις διαδρομές που ακολουθεί από τον ριζικό κόμβο έως τα φύλλα. Επιπλέον, μπορούν να χρησιμοποιηθούν με μεγάλα σύνολα δεδομένων, αν και κάποιες τροποποιήσεις, όπως τα δέντρα απόφασης μετά από δέντρα (Random Forests) μπορούν να βελτιώσουν την απόδοσή τους και να μειώσουν την πιθανότητα υπερπροσαρμογής.

3.5 Random Forests

Με τον ραγδαίο ρυθμό ανάπτυξης των πληροφοριακών συστημάτων, παράγονται τεράστιες ποσότητες δεδομένων που απαιτούν αποτελεσματικές μεθόδους εξόρυξης και κατηγοριοποίησης. Ένα από τα ισχυρότερα εργαλεία που έχουν αναπτυχθεί για αυτόν τον σκοπό είναι οι "Τυχαίοι Δάση" (Random Forests).

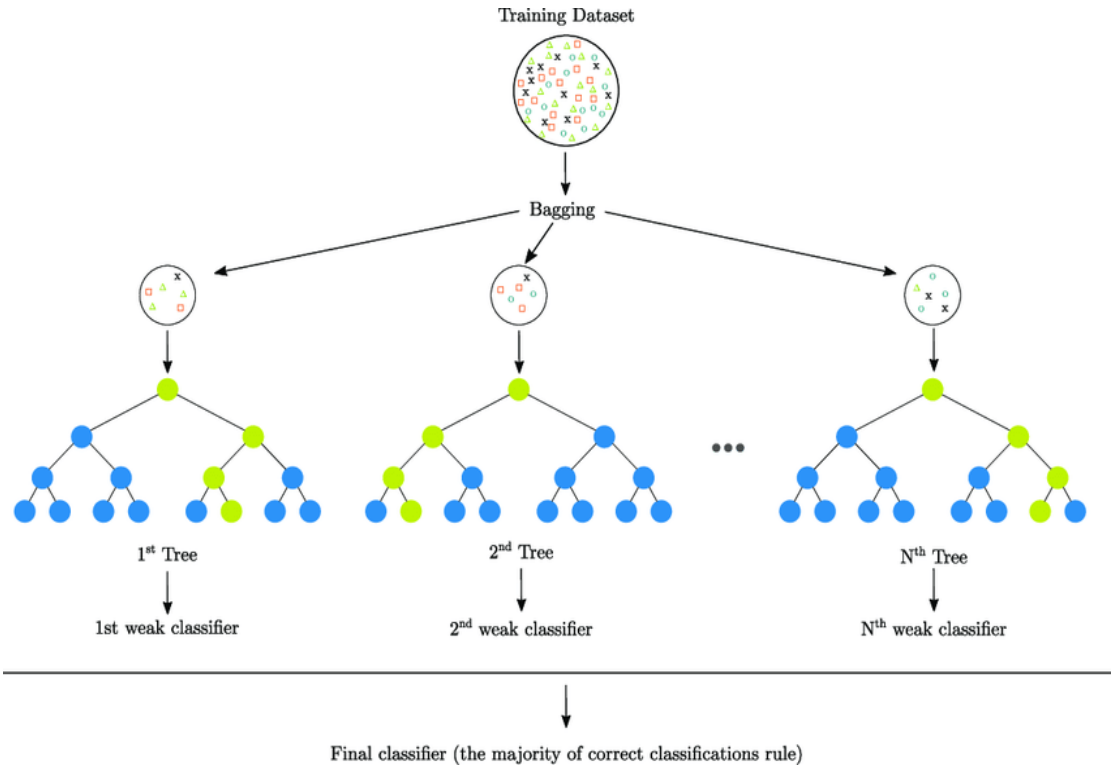
Τι είναι τα Random Forests; Τα Random Forests είναι ένας αλγόριθμος μηχανικής μάθησης που βασίζεται στην ιδέα της συνδυαστικής δύναμης πολλών απλών μοντέλων για την επίτευξη ενός πιο σταθερού και ακριβούς μοντέλου πρόβλεψης. Το Random Forest αποτελείται από πολλά δέντρα αποφάσεων που εκπαιδεύονται πάνω σε διαφορετικά υποσύνολα των δεδομένων εισόδου. Κάθε δέντρο αποφάσεων προβλέπει την έξοδο και η τελική απόφαση επιτυγχάνεται μέσω κάποιου συνόλου ψήφων ή μέσου όρου.

Στον τομέα της εξόρυξης δεδομένων, οι αλγόριθμοι Random Forests μπορούν να αναδείξουν σημαντικά χαρακτηριστικά και συσχετίσεις μεταξύ των δεδομένων. Επιπλέον, μπορούν να ανιχνεύσουν εξαιρέσεις και εσφαλμένα δεδομένα που ίσως δεν έχουν προηγουμένως προσδιοριστεί.

Ο ρόλος των Random Forests στην Κατηγοριοποίηση Δεδομένων Η κατηγοριοποίηση είναι η διαδικασία τοποθέτησης κάθε εισόδου σε μία από πολλές δυνατές κατηγορίες. Οι Random Forests είναι ιδιαίτερα αποτελεσματικοί στην κατηγοριοποίηση λόγω της ικανότητάς τους να αναλύουν τα πολύπλοκα χαρακτηριστικά των δεδομένων. Επιπλέον, με τη δυνατότητα των δέντρων να αντιμετωπίζουν αυξητικά την παραλληλοποίηση, οι Random Forests μπορούν να χειριστούν μεγάλα σύνολα δεδομένων με σχετικά χαμηλούς χρόνους εκπαίδευσης [28].

Τα Random Forests αποτελούν ένα ισχυρό εργαλείο στον τομέα της εξόρυξης και κατηγοριοποίησης δεδομένων. Με την ικανότητά τους να αντιμετωπίζουν πολύπλοκα προβλήματα, να ανακαλύπτουν συσχετίσεις και να προβλέπουν τα μελλοντικά δεδομένα, αποτελούν βασικό εργαλείο για τη λήψη αποφάσεων και την ανακάλυψη πληροφοριών από τα δεδομένα μας. Είναι σημαντικό, ωστόσο, να γίνει η σωστή επιλογή παραμέτρων και υπερπαραμέτρων κατά τη διάρκεια της εκπαίδευσης του μοντέλου, προκειμένου να επιτευχθούν βέλτιστα αποτελέσματα.

Ένα παράδειγμα χρήσης των αλγορίθμων random forests παρουσιάζεται στην εικόνα 10.



Εικόνα 10 - Παράδειγμα χρήσης του random forest. Πηγή: [28]

3.6 Η τεχνική Naive Bayes

Η τεχνική Naive Bayes είναι ένας από τους απλούστερους αλγορίθμους της μηχανικής μάθησης και χρησιμοποιείται ευρέως για προβλήματα κατηγοριοποίησης και πρόβλεψης. Βασίζεται στον θεωρητικό κανόνα του Bayes, ο οποίος περιγράφει πώς μπορούμε να υπολογίσουμε την πιθανότητα ενός συμβάντος βάσει της γνώσης μας για συναφή συμβάντα.

Ο αλγόριθμος Naive Bayes θεωρεί ότι τα χαρακτηριστικά των δεδομένων είναι ανεξάρτητα μεταξύ τους, πράγμα που ονομάζεται "αφελής" (naive) υπόθεση. Παρά την αφελή αυτή υπόθεση, ο αλγόριθμος συχνά παρουσιάζει εντυπωσιακή απόδοση σε πολλά προβλήματα.

Βασικά στάδια της τεχνικής Naive Bayes:

1. **Εκπαίδευση (Training):** Κατά το στάδιο αυτό, το μοντέλο εκπαιδεύεται χρησιμοποιώντας ένα σύνολο δεδομένων που περιλαμβάνει τις κατηγορίες (κλάσεις) και τα αντίστοιχα χαρακτηριστικά. Ο αλγόριθμος υπολογίζει τις συχνότητες των χαρακτηριστικών για κάθε κατηγορία.
2. **Πρόβλεψη (Prediction):** Κατά τη διάρκεια της πρόβλεψης, το μοντέλο χρησιμοποιεί τον αλγόριθμο Bayes για να υπολογίσει την πιθανότητα ενός

συμβάντος (κατηγορία) βάσει των χαρακτηριστικών που παρέχονται. Η κατηγορία με τη μεγαλύτερη πιθανότητα γίνεται η πρόβλεψη του μοντέλου.

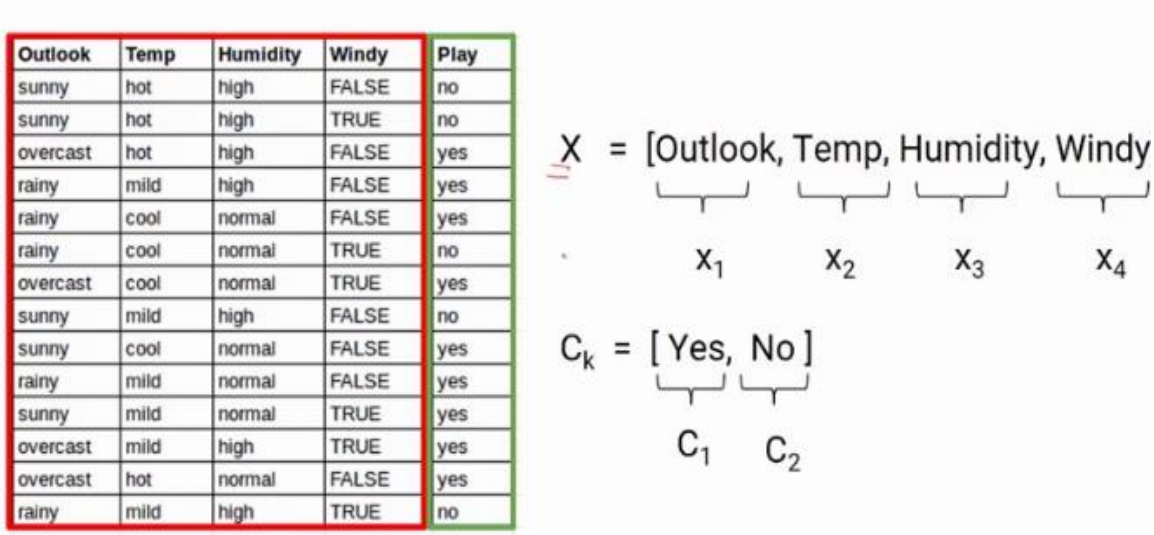
Πλεονεκτήματα της τεχνικής Naive Bayes:

- **Απλότητα:** Ο αλγόριθμος είναι εύκολος στην υλοποίηση και κατανόηση.
- **Επεκτασιμότητα:** Μπορεί να επεκταθεί για να χειριστεί πολλαπλές κατηγορίες και πολυκατηγορικά προβλήματα.
- **Αποτελεσματικότητα:** Παρουσιάζει καλή απόδοση σε προβλήματα με μικρό πλήθος δειγμάτων και μεγάλο αριθμό χαρακτηριστικών.

Εντούτοις, η αφελής υπόθεση ανεξαρτησίας μεταξύ των χαρακτηριστικών μπορεί να μην ισχύει στην πραγματικότητα, και αυτό μπορεί να επηρεάσει την ακρίβεια του μοντέλου. Παρ' όλα αυτά, ο Naive Bayes αποτελεί ένα χρήσιμο εργαλείο για προβλήματα κατηγοριοποίησης και αναγνώρισης προτύπων, και συχνά χρησιμοποιείται σε συστήματα αναγνώρισης κειμένου, φιλτραρίσματος spam, και άλλων παρόμοιων εφαρμογών.

Ένα παράδειγμα χρήσης της τεχνικής Naive Bayes δείχνει η εικόνα 11. Στο παράδειγμα αυτό υπάρχει ένα σύνολο δεδομένων με ορισμένες δυνατότητες Outlook, Temp, Humidity και Windy και ο στόχος εδώ είναι να προβλέψουμε εάν ένα άτομο ή μια ομάδα θα παίξει τένις ή όχι. Έτσι, αντιπροσωπεύουμε χαρακτηριστικά ως X όπως X1, X2 και ούτω καθεξής. Ομοίως, οι κλάσεις αντιπροσωπεύονται ως C1 και C2. Στο Naive Bayes για κάθε παρατήρηση, προσδιορίζουμε την πιθανότητα να ανήκει στην κατηγορία 1 ή στην κατηγορία 2.

Για παράδειγμα, εδώ πρώτα ανακαλύπτουμε την πιθανότητα να παίξει το άτομο δεδομένου ότι το Outlook είναι Sunny, η θερμοκρασία είναι ζεστή, η υγρασία είναι υψηλή και δεν φυσάει όπως φαίνεται παρακάτω. Αργότερα, μπορεί να υπολογιστεί επίσης η πιθανότητα το άτομο να μην παίξει με τις ίδιες συνθήκες. Αυτό επαναλαμβάνεται για όλες τις σειρές. Έτσι, κατά κάποιον τρόπο υπολογίζεται η υπό όρους πιθανότητα, όπου προσπαθούμε να προβλέψουμε την κλάση με βάση τις συνθήκες ή τα χαρακτηριστικά εδώ.



Outlook	Temp	Humidity	Windy	Play
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	FALSE	yes
rainy	cool	normal	TRUE	no
overcast	cool	normal	TRUE	yes
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	mild	normal	TRUE	yes
overcast	mild	high	TRUE	yes
overcast	hot	normal	FALSE	yes
rainy	mild	high	TRUE	no

Εικόνα 11 - *παράδειγμα εφαρμογής της τεχνικής* Naive *Bayes.*

3.7 Τεχνητά Νευρωνικά δίκτυα

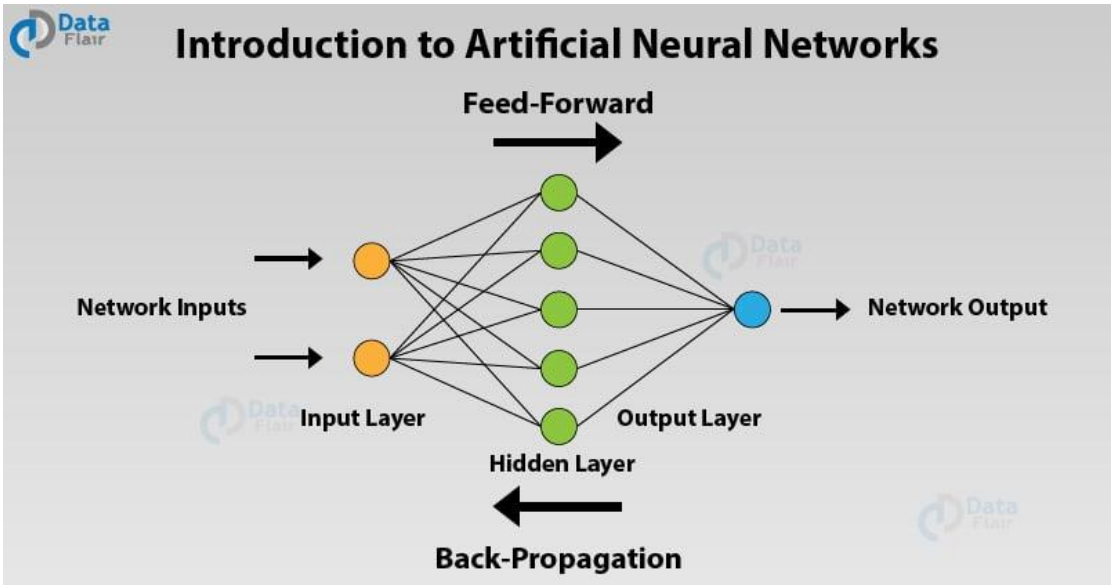
Τα τεχνητά νευρωνικά δίκτυα είναι οι πιο δημοφιλείς αλγόριθμοι μηχανικής μάθησης σήμερα. Η εφεύρεση αυτών των νευρωνικών δικτύων έλαβε χώρα τη δεκαετία του 1970, αλλά έχουν επιτύχει τεράστια δημοτικότητα λόγω της πρόσφατης αύξησης της υπολογιστικής ισχύος εξαιτίας της οποίας βρίσκονται πλέον σχεδόν παντού.

Τα τεχνητά νευρωνικά δίκτυα (ANN) έχουν επιτύχει μεγάλη επιτυχία στην εκτέλεση εργασιών μηχανικής μάθησης, όπως ταξινόμηση, παλινδρόμηση, πρόβλεψη, επεξεργασία εικόνας, αναγνώριση εικόνας κ.λπ., λόγω της εξαιρετικής εκπαίδευσης, εκμάθησης και οργάνωσης δεδομένων. Συμβατικά, ένας αλγόριθμος που βασίζεται σε κλίση, γνωστός ως back propagation (BP) χρησιμοποιείται συχνά για την εκπαίδευση της τιμής των παραμέτρων του ANN (εικόνα 12).

Ωστόσο, αυτή η μέθοδος έχει εγγενή μειονεκτήματα της αργής ταχύτητας σύγκλισης, της ευαισθησίας στις αρχικές λύσεις και της μεγάλης τάσης να παγιδεύεται σε τοπικό βέλτιστο λόγω της πολυπλοκότητας του τοπίου βελτιστοποίησης. Ως απάντηση σε αυτές τις προκλήσεις, έχουν αναπτυχθεί διάφορες τεχνικές και αλγόριθμοι για τη βελτίωση της απόδοσης των τεχνητών νευρωνικών δικτύων και την υπέρβαση των περιορισμών της παραδοσιακής οπισθοδρόμησης.

Εξέλιξη των τεχνητών νευρωνικών δικτύων

Τα τεχνητά νευρωνικά δίκτυα, εμπνευσμένα από τη δομή και τη λειτουργία του ανθρώπινου εγκεφάλου, έχουν αναδειχθεί ως ένα ισχυρό εργαλείο στη μηχανική μάθηση. Αποτελούνται από διασυνδεδεμένους κόμβους ή «νευρώνες», οργανωμένους σε στρώματα, συμπεριλαμβανομένων των επιπέδων εισόδου, κρυφού και εξόδου. Τα ANN είναι γνωστά για την ικανότητά τους να μαθαίνουν πολύπλοκα μοτίβα και σχέσεις από δεδομένα, καθιστώντας τα κατάλληλα για ένα ευρύ φάσμα εφαρμογών. Διαπρέπουν σε εργασίες όπως η αναγνώριση μοτίβων σε εικόνες, η πραγματοποίηση προβλέψεων με βάση ιστορικά δεδομένα και η ταξινόμηση αντικειμένων σε διαφορετικές κατηγορίες.[29]



Εικόνα 12 - ***τρόπος λειτουργίας των τεχνητών νευρωνικών δικτύων.***

Η back-propagation, ο ακρογωνιαίος λίθος της εκπαίδευσης νευρωνικών δικτύων, στοχεύει στην ελαχιστοποίηση της διαφοράς μεταξύ των προβλεπόμενων και των πραγματικών εξόδων προσαρμόζοντας τις παραμέτρους του δικτύου μέσω της βελτιστοποίησης gradient descent. Λειτουργεί με τη διάδοση της διαβάθμισης σφάλματος προς τα πίσω μέσω του δικτύου, ενημερώνοντας επαναληπτικά τα βάρη και τις προκαταλήψεις κάθε νευρώνα για να ελαχιστοποιηθεί η συνάρτηση απώλειας. Αυτή η διαδικασία καθοδηγεί το δίκτυο προς την εκμάθηση των υποκείμενων προτύπων και σχέσεων μέσα στα δεδομένα.

3.8 Μετρικές

Σε αυτή την ενότητα θα αναφερθούν οι μετρικές που θα χρησιμοποιηθούν για την αξιολόγηση των μοντέλων.

3.8.1 Ακρίβεια (Accuracy)

Η ακρίβεια είναι η μετρική που αξιοποιείται ώστε να υπολογιστεί η ακρίβεια του μοντέλου κατηγοριοποίησης. Η ακρίβεια είναι ο λόγος των σωστών προβλέψεων (θετικές και αρνητικές) προς όλες τις περιπτώσεις [30]

$$Ακρίβεια = \frac{Θετικές\ Πρόβλεψεις + Αρνητικές\ Προβλέψεις}{Συνολικές\ Περιπτώσεις}$$

Εξίσωση 5

3.8.2 Ακρίβεια (Precision)

Η ακρίβεια (precision) είναι η μετρική που αξιολογεί την ποιότητα των προβλέψεων του μοντέλου κατηγοριοποίησης. Συγκεκριμένα εστιάζεται στην επίδοση του μοντέλου όταν προβλέπουν θετικές κλάσεις. [31]

$$Ακρίβεια = \frac{Θετικές\ Προβλέψεις}{Θετικές\ Προβέψεις + Λανθασμένες\ Θετικές\ Προβλέψεις}$$

Εξίσωση 6

3.8.3 Ανάκληση (Recall)

Είναι η μετρική που αλλιώς ονομάζεται και ευαισθησία (Sensitivity). Αξιολογεί την ικανότητα του μοντέλου αν αναγνωρίζει σωστά τις κατηγορίες. Είναι ιδιαίτερα χρήσιμη μετρική σε περιπτώσεις όπου οι λανθασμένες αρνητικές προβλέψεις πρέπει να ληφθούν παραπάνω υπόψιν από της λανθασμένες θετικές προβλέψεις. [32]

$$Ανάκληση = \frac{Θετικές\ Προβλέψεις}{Θετικές\ Προβλέψεις + Λανθασμένες\ Αρνητικές\ Προβλέψεις}$$

Εξίσωση 7

3.8.4 Βαθμολογία F1 (F1-Score)

Η βαθμολογία F1 (F1-Score) αποτελεί τον αρμονικό μέσο όρο της ακρίβειας (precision) και της ανάκλησης (recall), προσφέροντας έτσι μια ισορροπημένη μέτρηση μεταξύ των δύο. Είναι ιδιαίτερα χρήσιμη όταν υπάρχει ανισόμετρη κατανομή κλάσεων ή όταν οι λανθασμένες θετικές προβλέψεις και οι λανθασμένες αρνητικές προβλέψεις έχουν διάφορα κόστη. Η φόρμουλα για τον υπολογισμό της βαθμολογίας F1 είναι: [33]

$$F1 = 2 * (\frac{Ακρίβεια * Ανάκληση}{Ακρίβεια + Ανάκληση})$$

Εξίσωση 8

4 Πειραματικό Μέρος

Σκοπός αυτού του κεφαλαίου είναι να παρουσιαστούν οι εφαρμογές κάποιων αλγορίθμων μηχανικής μάθησης και μεθευρετικών αλγορίθμων, όπου αναπτύχθηκαν στα προηγούμενα κεφάλαια της εργασίας. Θα αναπτυχθεί ο μεθευρετικός αλγόριθμος της Προσομοιωμένης Ανόπτησης (*Simulated annealing*) και οι αλγόριθμοι μηχανικής μάθησης όπως K-NN και SVM. Με αποτέλεσμα ο αναγνώστης να μπορέσει να αποκτήσει μια ολοκληρωμένη εμπειρία με τους αλγόριθμους μηχανικής μάθησης και τους μεθευρετικούς αλγορίθμους.

4.1 Πληροφορίες Συνόλου Δεδομένων (Dataset)

Η βάση που θα χρησιμοποιηθεί για την εκπαίδευση και την αξιολόγηση των αλγορίθμων μηχανικής μάθησης είναι η 20 Newsgroups Dataset.

Το 20 Newsgroups Dataset είναι μια συλλογή κειμένων από 20 διαφορετικά newsgroups. Χρησιμοποιείται κυρίως για την ταξινόμηση κειμένου και έχει χρησιμοποιηθεί ευρέως στην κοινότητα της μηχανικής μάθησης για την εκπαίδευση και τον έλεγχο των αλγορίθμων.

4.1.1 Ιστορικό

Αυτή η βάση δεδομένων δημιουργήθηκε στα μέσα της δεκαετίας του '90 και αποτελείται από συλλογές μηνυμάτων από διαφορετικά newsgroups. Το "20 Newsgroups" προσφέρει μια μοναδική ευκαιρία για την ανάλυση των προτιμήσεων και των συμπεριφορών των χρηστών σε πλατφόρμες ενημέρωσης και συζήτησης, παρέχοντας έτσι ένα πλούσιο πεδίο για την εφαρμογή τεχνικών μηχανικής μάθησης.
Πολλές έρευνες στον τομέα της μηχανικής μάθησης και της πληροφορικής έχουν χρησιμοποιήσει το "20 Newsgroups" για πειράματα σχετικά με ταξινόμηση κειμένου, ομαδοποίηση και άλλες εφαρμογές όπως ανάλυση συναισθημάτων.

Η βάση προσφέρεται για την εφαρμογή τεχνικών επιλογής χαρακτηριστικών και μεθευρετικών μεθόδων. Πιο συγκεκριμένα, ο A. K. M. Bahalul Haque και συνεργάτες έχουν χρησιμοποιήσει το 20 Newsgroups Dataset σε διάφορες έρευνες, όπως το "A Comparative Study of Machine Learning Techniques for Text Classification"[1][2]. Άλλες

σημαντικές έρευνες που χρησιμοποιούν το 20 Newsgroups Dataset περιλαμβάνουν το "A Comparison of Machine Learning Techniques for Text Classification" από τον S. Kotsiantis [3] και το "A Study of Machine Learning Techniques for Spam Filtering" από τον S. Carrillo [4].

Οι αναλύσεις που βασίζονται στις 2 αυτές βάσεις μπορούν να προσφέρουν νέες προσεγγίσεις στον τομέα της επεξεργασίας φυσικής γλώσσας και της κατανόησης του περιεχομένου των κειμένων, βελτιώνοντας την εμπειρία των χρηστών και την αποτελεσματικότητα των πλατφορμών ενημέρωσης.

4.1.2 Χαρακτηριστικά του Dataset

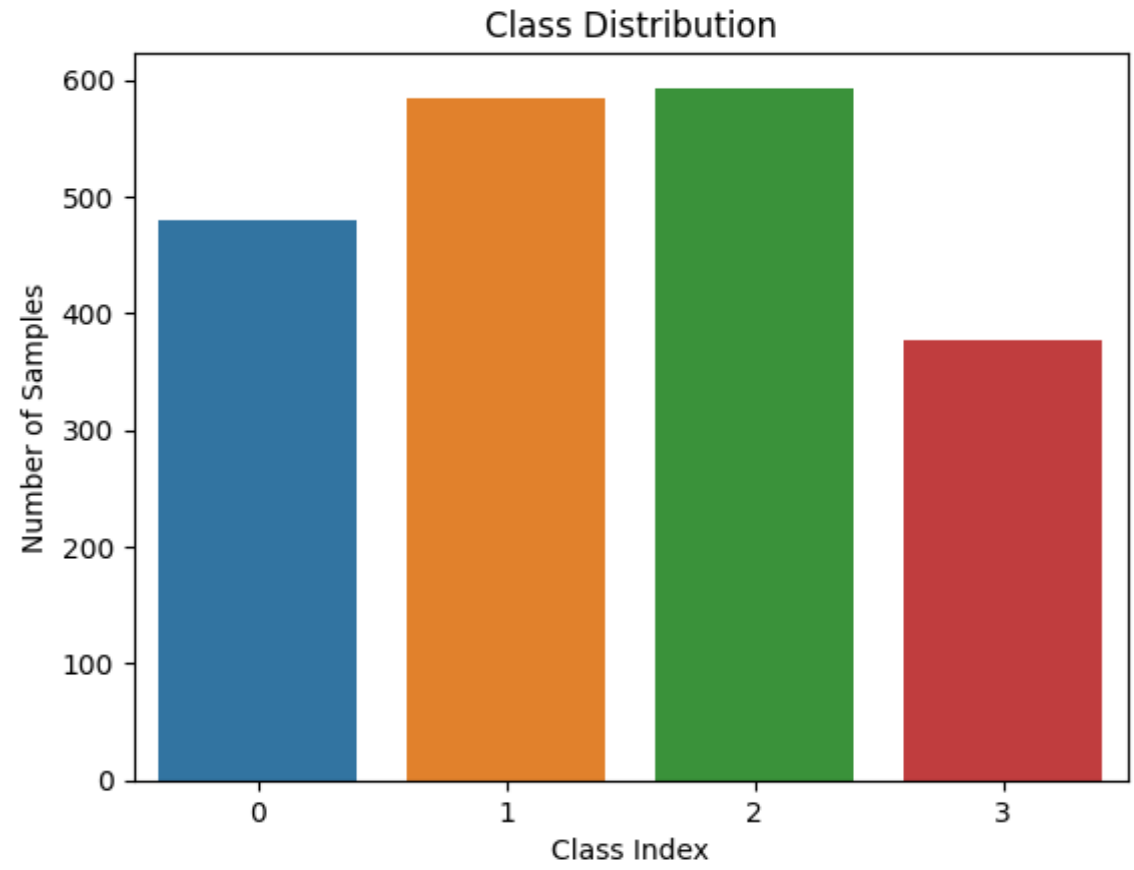
Το "20 Newsgroups" είναι μια συλλογή περίπου 18.000 εγγράφων από 20 διαφορετικά newsgroups, που έχει γίνει δημοφιλής για πειράματα σε εφαρμογές μηχανικής μάθησης, όπως η ταξινόμηση και η ομαδοποίηση κειμένου.

Τα δεδομένα οργανώνονται σε 21 αρχεία: ένα αρχείο list.csv, το οποίο περιέχει αναφορές στον αριθμό του document_id και το newsgroup με το οποίο σχετίζεται, καθώς και 20 αρχεία που περιέχουν τα έγγραφα, ένα για κάθε newsgroup.

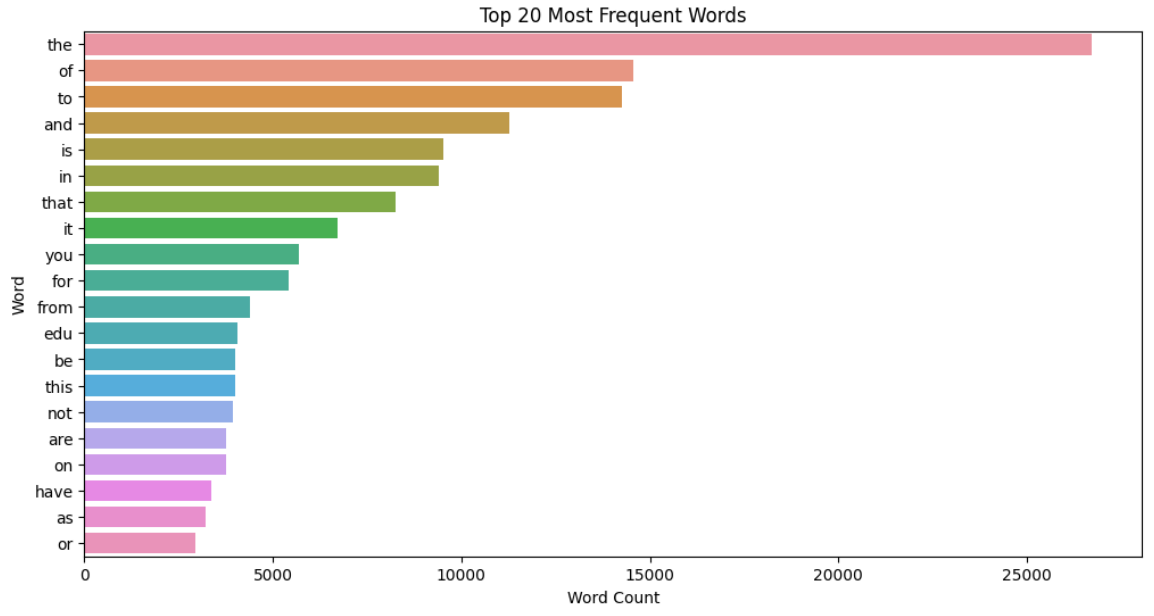
Στατιστικά στοιχεία:

Συνολικός αριθμός εγγράφων: 18,828

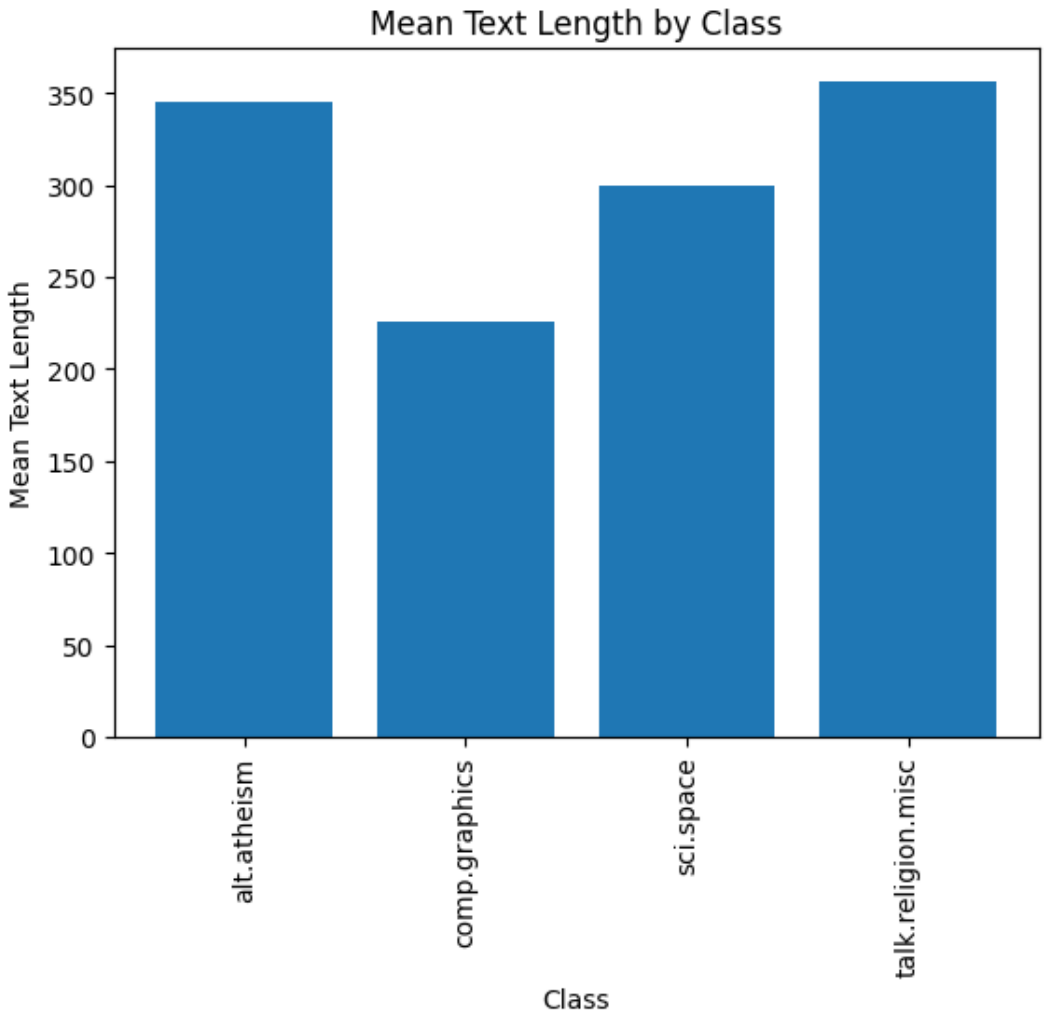
Αριθμός newsgroups: 20



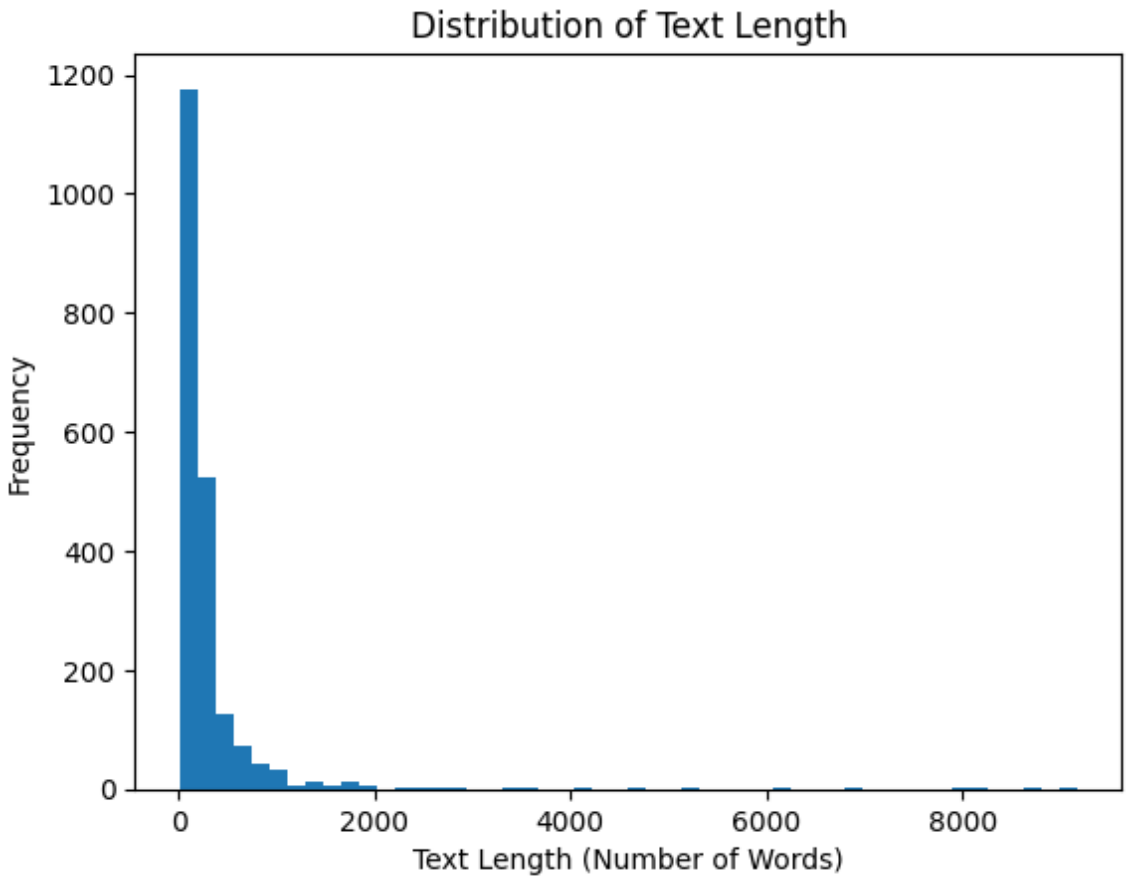
Εικόνα 13 – Κατανομή των δειγμάτων



Εικόνα 14 – Top 20 συχνότερες λέξεις



Εικόνα 15 – Μέσος όρος μεγέθους λέξεων που περιέχονται στις θεματικές κατηγορίες



Εικόνα 16 – Κατανομή μεγέθους λέξεων

4.2 Παρουσίαση Κώδικα

Ακολουθεί ανάπτυξη κώδικα, χρησιμοποιώντας το σύνολο δεδομένων που αναπτύξαμε στο προηγούμενο κεφάλαιο. Θα γίνει επεξήγηση κάθε βήματος του κώδικα, από τις βιβλιοθήκες που θα συμπεριληφθούν, την προετοιμασία των δεδομένων, την εκπαίδευση των μοντέλων και την αξιολόγησή τους.

Ο κώδικας θα αναπτυχθεί σε γλώσσα προγραμματισμού python, με τη βοήθεια του Jupiter. Το Jupiter μας δίνεται η δυνατότητα να χρησιμοποιήσουμε πολλές βιβλιοθήκες της python που είναι προ εγκατεστημένες και να εκτελέσουμε τον κώδικα μας σταδιακά. Ως αποτέλεσμα είναι πιο εύκολη η κατανόηση της εκτέλεσης του κώδικα.

Τέλος θα γίνει μια παρουσίαση των τρόπων όπου μπορούν να ενισχύσουν οι μεθευρετικοί αλγόριθμοι την επίδοση των αλγορίθμων μηχανικής μάθησης. Σε ποια σημεία μπορούν να εφαρμοστούν και για πιο σκοπό.

4.2.1 Υλοποίηση του Αλγόριθμου Προσομοιωμένης Ανόπτησης

Στην ενότητα αυτή θα γίνει η ανάπτυξη του αλγορίθμου σε γλώσσα προγραμματισμού Python και ο κώδικας έχει παρθεί από [22] .

Για αρχή ορίζουμε την αντικειμενική συνάρτηση και το εύρος του χώρου αναζήτησης. Σε αυτή την υλοποίηση για λόγους απλότητας και κατανόησης θα ορίσουμε μια απλή μονοδιάστατη εκθετική αντικειμενική συνάρτηση.

```
# objective function
def objective(x):
    return x**2.0

# define range for input
r_min, r_max = -5.0, 5.0
# sample input range uniformly at 0.1 increments
inputs = arange(r_min, r_max, 0.1)
```

Κώδικας 1 – Αντικειμενική συνάρτηση

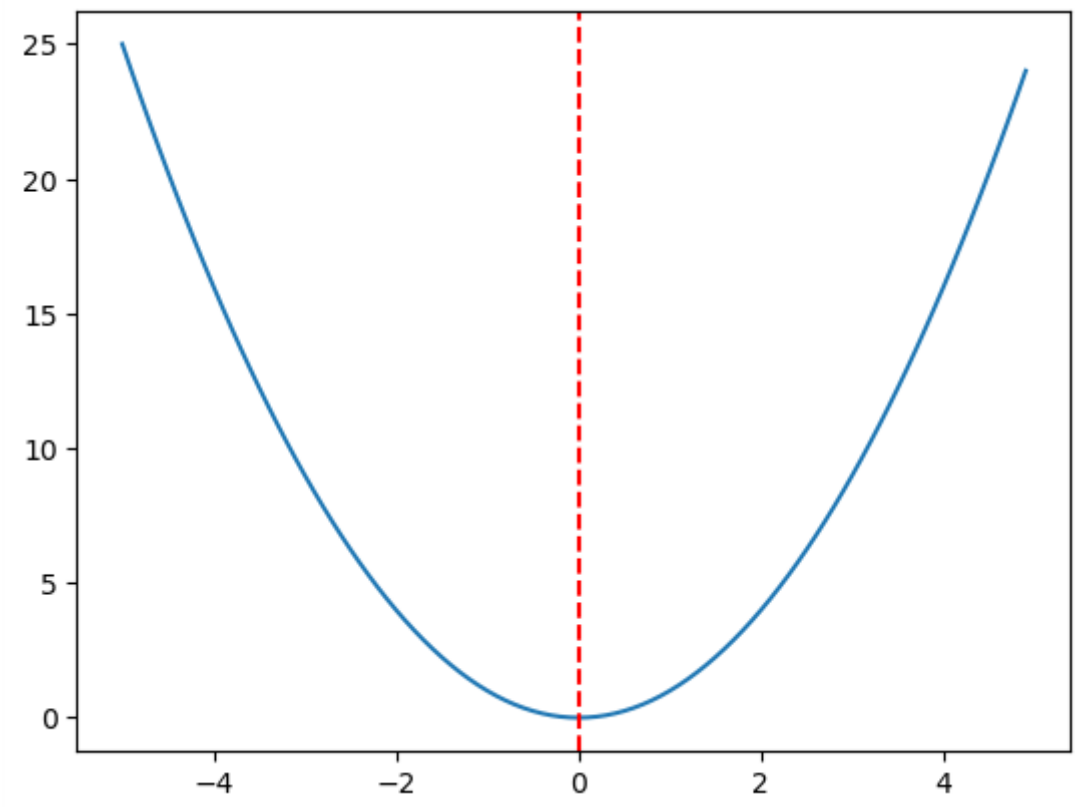
Λόγο της απλότητας της αντικειμενικής συνάρτησης γνωρίζουμε ότι η βέλτιστη λύση είναι 0. Για να μπορέσουμε να το αναπαράγουμε σε διάγραμμα, θα πρέπει να υπολογίσουμε την αντικειμενική συνάρτηση όλων των σημείων στο χώρο αναζήτησης.

```
# compute targets
results = [objective(x) for x in inputs]

# create a line plot of input vs result
pyplot.plot(inputs, results)
# define optimal input value
x_optima = 0.0
# draw a vertical line at the optimal input
pyplot.axvline(x=x_optima, ls='--', color='red')
# show the plot
pyplot.show()
```

Κώδικας 2 – Παραγωγή εκθετικής συνάρτησης

Ο κώδικας παράγει το ακόλουθο διάγραμμα. Όπου επιβιώνετε ότι η βέλτιστη λύση της αντικειμενικής συνάρτησης είναι το μηδέν.



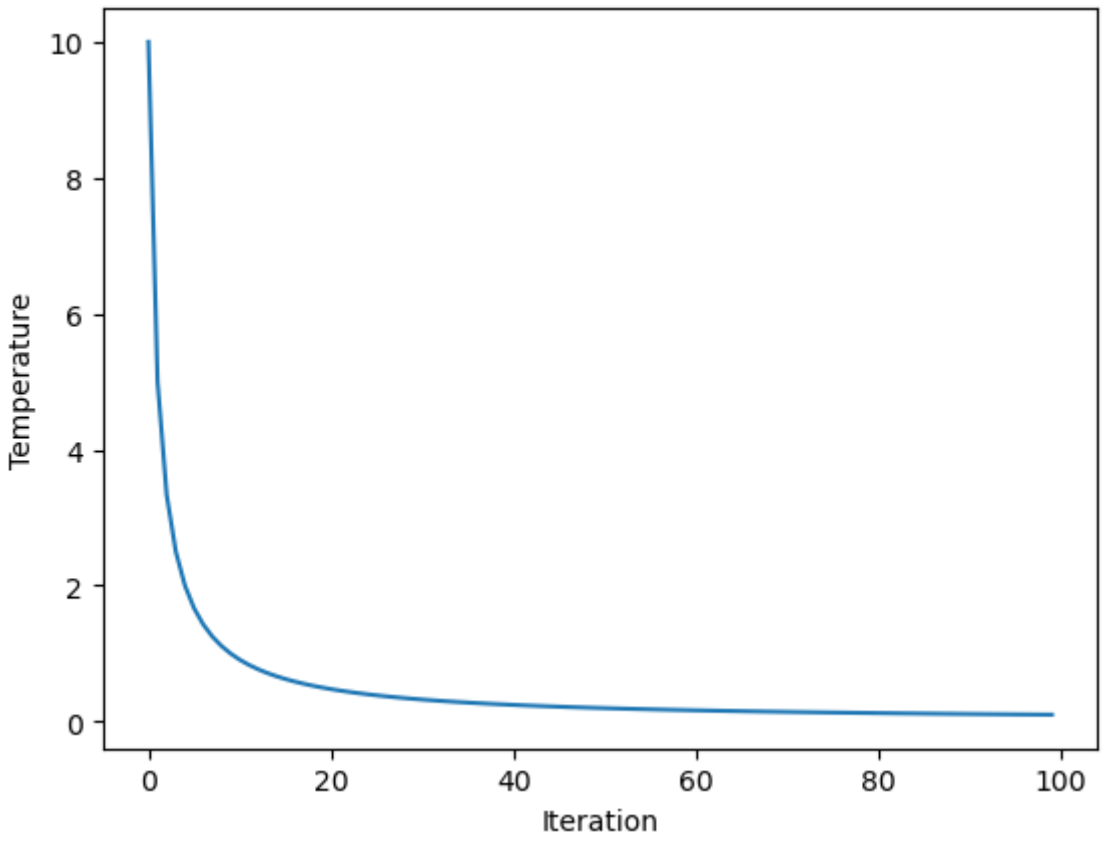
Εικόνα 17 – Εκθετική συνάρτηση

Για την καλύτερη κατανόηση του Αλγόριθμου Προσομοιωμένης Ανόπτησης, πριν την υλοποίηση του, θα μελετηθεί πως επηρεάζεται το κριτήριο αποδοχής (*metropolis*

acceptance) από τη θερμοκρασία και την διαφορά των τιμών μεταξύ της υποψήφιας λύσης και την τρέχουσας λύσης κατά τη διάρκεια των επαναλήψεων. Στα ακόλουθα διαγράμματα που θα παραχθούν, θα γίνει ξεκάθαρο πως η θερμοκρασία επηρεάζεται εκθετικά στο πέρασμα των επαναλήψεων, με αποτέλεσμα το κριτήριο αποδοχής (*metropolis acceptance)* να γίνεται αρκετά αυστηρό ώστε να δεχτεί μια νέα τιμή που να είναι χειρότερη από την τρέχον. Επιπλέον, θα αποκτήσει ο αναγνώστης μια πιο πλήρη εικόνα, στην επίδραση που έχει η διαφορά της υποψήφιας λύσης με την τρέχουσα, στο πέρασμα των επαναλήψεων.

```
# quick check how temperature is affected by iterations
# total iterations of algorithm
iterations = 100
# initial temperature
initial_temp = 10
# array of iterations from 0 to iterations -1
iterations = [i for i in range(iterations)]
# temperature for each iterations
temperatures = [initial_temp/float(i + 1) for i in iterations]
# plot iterations vs temperature
pyplot.plot(iterations, temperatures)
pyplot.xlabel('Iteration')
pyplot.ylabel('Temperature')
pyplot.show()
```

Κώδικας 3 – Παραγωγή διαγράμματος παρουσίασης Θερμοκρασίας και Επαναλήψεων



Εικόνα 18 – Διάγραμμα παρουσίασης Θερμοκρασίας και Επαναλήψεων

Παρατηρώντας το παραγόμενο διάγραμμα, γίνεται ξεκάθαρο πως μετά την 20 επανάληψη έχει πέσει εκθετικά η τιμή της θερμοκρασίας και κατ' επέκταση η ανεκτικότητα του αλγορίθμου σε χειρότερες τιμές.

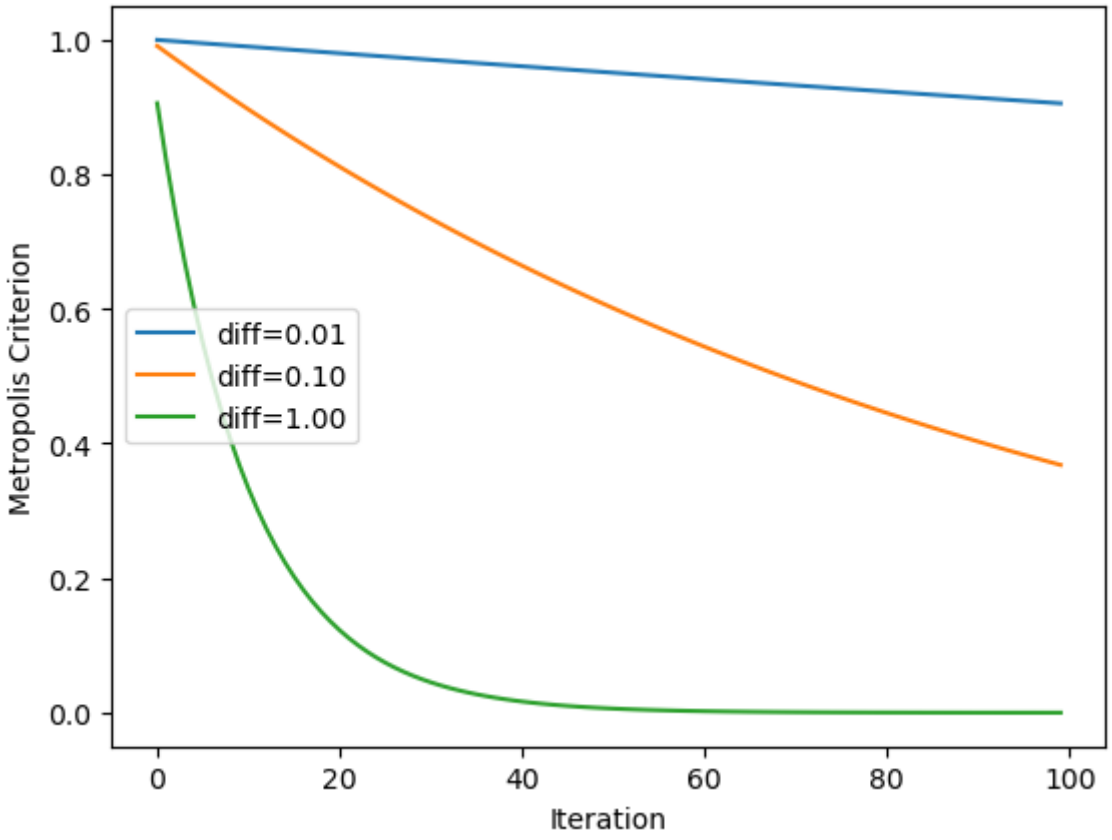
Ακολουθεί το διάγραμμα της διαφοράς μεταξύ της υποψήφιας λύσης με την τρέχουσα ως προς τις επαναλήψεις.

```
from math import exp
from matplotlib import pyplot
# total iterations of algorithm
iterations = 100
# initial temperature
initial_temp = 10
# array of iterations from 0 to iterations -1
iterations = [i for i in range(iterations)]
# temperature for each iterations
temperatures = [initial_temp/float(i + 1) for i in iterations]
# metropolis acceptance criterion
differences = [0.01, 0.1, 1.0]
```

```
for d in differences:
    metropolis = [exp(-d/t) for t in temperatures]
    # plot iterations vs metropolis
    label = 'diff=%.2f' % d
    pyplot.plot(iterations, metropolis, label=label)
# plot iterations vs temperature
pyplot.xlabel('Iteration')
pyplot.ylabel('Metropolis Criterion')
pyplot.legend()
pyplot.show()
```

Κώδικας 4 - Παραγωγής διαγράμματος metropolis acceptance ως προς των επαναλήψεων

Ο παραπάνω κώδικας παράγει το εξής διάγραμμα:



Εικόνα 19 – Διάγραμμα metropolis acceptance ως προς των επαναλήψεων

Σύμφωνα με το διάγραμμα γίνεται ξεκάθαρο πως η μεγάλη διαφορά των αντικειμενικών συναρτήσεων της νέας με της τρέχουσας τιμής στο πέρασμα των επαναλήψεων κάνει σχεδόν μηδενική πιθανότητα την αποδοχή μια χειρότερης λύσης.

Από τη στιγμή που έχει κατανοηθεί η συμπεριφορά του κριτήριου αποδοχής, θα παρουσιαστεί η υλοποίηση του αλγορίθμου.

```
# simulated annealing algorith
def simulated_annealing(objective, bounds, n_iterations, step_size, temp):
    # generate an initial point
    best = bounds[:, 0] + rand(len(bounds)) * (bounds[:, 1] - bounds[:,
0])
    # evaluate the initial point
    best_eval = objective(best)
    # current working solution
    curr, curr_eval = best, best_eval
    scores = list()
    # run the algorith
    for i in range(n_iterations):
        # take step
        candidate = curr + randn(len(bounds)) * step_size
        # evaluate candidate point
        candidate_eval = objective(candidate)
        # check for new best solution
        if candidate_eval < best_eval:
            # store new best point
            best, best_eval = candidate, candidate_eval
            # keep track of scores
            scores.append(best_eval)
            # report progress
            print('>%d f(%s) = %.5f' % (i, best, best_eval))
            # difference between candidate and current point evaluation
            diff = candidate_eval - curr_eval
            # calculate temperature for each epoch
            t = temp / float(i + 1)
            # calculate metropolis acceptance criterion
            metropolis = exp(-diff/t)
            # check if we should keep the new point
            if diff < 0 or rand() < metropolis:
                # store the new current point
                curr, curr_eval = candidate, candidate_eval
    return [best, best_eval, scores]
```

Κώδικας 5 – Κώδικας Προσομοιωμένης Ανόπτησης

Στην αρχή της διαδικασίας ορίζεται ένα τυχαίο σημείο μέσα στα όρια των τιμών εισόδων, ως την καλύτερη λύση.

```
# generate an initial point
best = bounds[:, 0] + rand(len(bounds)) * (bounds[:, 1] - bounds[:, 0])
# evaluate the initial point
best_eval = objective(best)
# current working solution
curr, curr_eval = best, best_eval
```

Κώδικας 6 – Αρχικοποίηση λύσης

Στην συνέχεια ξεκινάει την αναζήτηση της βέλτιστης τιμής. Σε κάθε επανάληψη ο αλγόριθμος επιλέγει μια υποψήφια λύση και υπολογίζει την αντικειμενική συνάρτηση της. Τέλος γίνεται η αποδοχή της νέας λύσης σύμφωνα με όσα αναπτύχθηκαν παραπάνω. Δηλαδή είτε η νέα τιμή να είναι καλύτερη από την προηγούμενη είτε να υπάρξει πιθανότητα να αποδεχτεί μια χειρότερη τιμή, όπου αυτή η πιθανότητα είναι γνωστή ως *metropolis acceptance*

```
# run the algorith
for i in range(n_iterations):
    # take step
    candidate = curr + randn(len(bounds)) * step_size
    # evaluate candidate point
    candidate_eval = objective(candidate)
    # check for new best solution
    if candidate_eval < best_eval:
        # store new best point
        best, best_eval = candidate, candidate_eval
        # keep track of scores
        scores.append(best_eval)
        # report progress
        print('>%d f(%s) = %.5f' % (i, best, best_eval))
        # difference between candidate and current point evaluation
        diff = candidate_eval - curr_eval
        # calculate temperature for each epoch
        t = temp / float(i + 1)
        # calculate metropolis acceptance criterion
        metropolis = exp(-diff/t)
        # check if we should keep the new point
        if diff < 0 or rand() < metropolis:
            # store the new current point
            curr, curr_eval = candidate, candidate_eval
```

Κώδικας 7 – Αναζήτηση καλύτερης τιμής

Την μεταβλητή *score* χρησιμοποιείτε ώστε στη συνέχεια να μπορέσει να παραχθεί το διάγραμμα, όπου φαίνεται η καλύτερη τιμή κάθε φορά που υπήρχε βελτίωση.

Τέλος πρέπει να οριστούν οι παράμετροί του αλγορίθμου και να γίνει η εκτέλεσή του.

```
# seed the pseudorandom number generator
seed(1)
# define range for input
bounds = asarray([[-5.0, 5.0]])
# define the total iterations
n_iterations = 1000
# define the maximum step size
step_size = 0.1
# initial temperature
temp = 10
```

Κώδικας 8 – Αρχικοποίηση παραμέτρων και τιμών

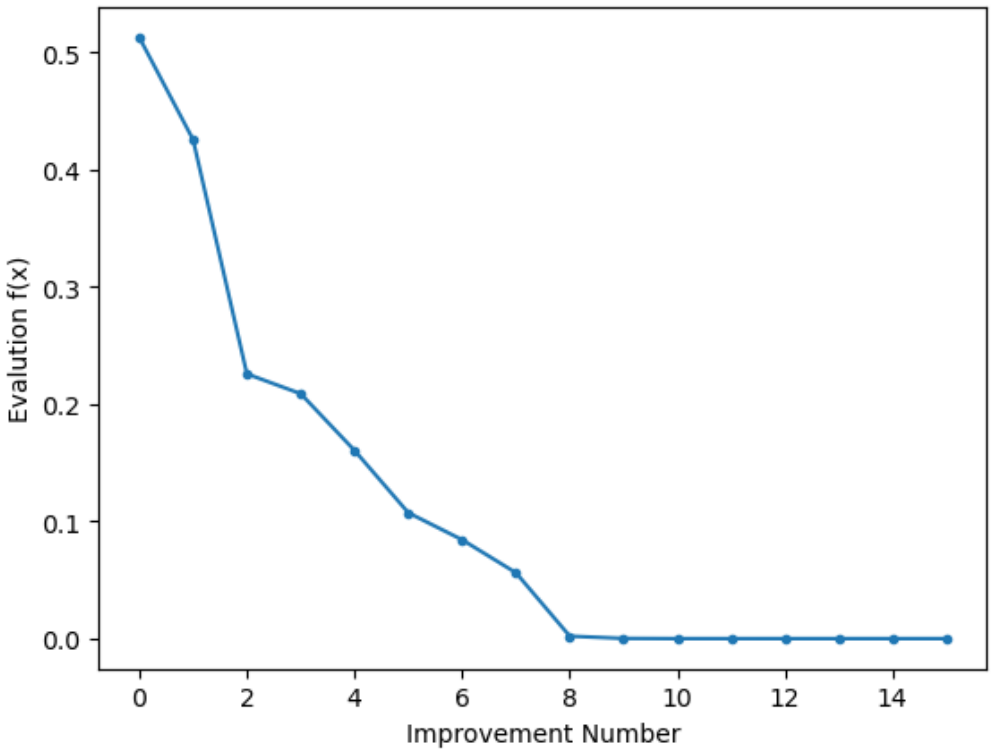
```
# perform the simulated annealing search
best, score, scores = simulated_annealing(objective, bounds, n_iterations,
step_size, temp)
print('Done')
print('f(%s) = %f' % (best, score))
# line plot of best scores
pyplot.plot(scores, '.-')
pyplot.xlabel('Improvement Number')
pyplot.ylabel('Evalution f(x)')
pyplot.show()
```

Κώδικας 9 – Εκτέλεση κώδικα

Με την εκτέλεση του αλγορίθμου έχουμε το εξής αποτέλεσμα:

```
>5 f([-0.71624542]) = 0.51301
>7 f([-0.65250924]) = 0.42577
>9 f([-0.47524847]) = 0.22586
>11 f([-0.45712705]) = 0.20897
>12 f([-0.40069256]) = 0.16055
>14 f([-0.327695]) = 0.10738
>15 f([-0.29039562]) = 0.08433
>16 f([-0.23701453]) = 0.05618
>18 f([-0.04563249]) = 0.00208
>19 f([-0.01255278]) = 0.00016
>25 f([0.00329871]) = 0.00001
>32 f([-0.00295045]) = 0.00001
>47 f([-0.00066185]) = 0.00000
>79 f([0.00065104]) = 0.00000
>251 f([0.00032916]) = 0.00000
>440 f([-0.00030124]) = 0.00000
Done
f([-0.00030124]) = 0.000000
```

Εικόνα 20 – Βέλτιστη τιμή κάθε επανάληψης



Εικόνα 21 – Διάγραμμα της εκτίμησης της αντικειμενικής συνάρτησης σε κάθε βελτίωση

Όπως συμπεραίνεται και από τις αναφορές αρκετά γρήγορα ο αλγόριθμος Προσομοιωμένης Ανόπτησης είναι κοντά στην βέλτιστη τιμή.

4.2.2 Ανάλυση ανάπτυξης κώδικα

Εισαγωγή βιβλιοθηκών

Ξεκινώντας το κώδικα πρέπει να συμπεριληφθούν οι βιβλιοθήκες που θα αξιοποιηθούν για την εκτέλεση του πειραματικού κώδικα.

Θα χρησιμοποιηθούν βασικές βιβλιοθήκες για διαχείριση δεδομένων και δημιουργία διαγραμμάτων στην python, όπως *numpy*, *pandas*, *matplotlib*.

```
# general imports
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Κώδικας 10 – Εισαγωγή βιβλιοθηκών

Στη συνέχεια θα εισάγουμε και τις βιβλιοθήκες που θα χρειαστούμε για την εισαγωγή το σύνολο δεδομένων (*dataset*), προετοιμασία των δεδομένων μας, το μοντέλο μηχανικής μάθησης και τα εργαλεία για την αξιολόγησή του.

Στο συγκεκριμένο κομμάτι κώδικα γίνεται εισαγωγή ο K-NN αλγόριθμος κατηγοριοποίησης.

```
# sklearn imports
from sklearn.datasets import fetch_20newsgroups
from sklearn.decomposition import TruncatedSVD
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import Normalizer

from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
from sklearn.metrics import ConfusionMatrixDisplay
```

Κώδικας 11 – Εισαγωγή Sklearn βιβλιοθηκών

Χρήση συνόλου δεδομένων και προετοιμασία δεδομένων

```
# Define categories of datasets
categories = ['alt.atheism', 'talk.religion.misc', 'comp.graphics',
'sci.space']

# Define train dataset
data_train = fetch_20newsgroups(subset='train', categories=categories,
random_state=42)

# Convert to TF-DF format
vectorizer = TfidfVectorizer(max_df=0.5, min_df=2, stop_words='english',
use_idf=True)
# Converted X_train subset
X_train = vectorizer.fit_transform(data_train.data)


# Define test dataset
data_test = fetch_20newsgroups(subset='test', categories=categories,
random_state=42)

# Define y_train, y_test subsets
y_train, y_test = data_train.target, data_test.target

# Define X_test subset
X_test = vectorizer.transform(data_test.data)
```

Κώδικας 12 – Προετοιμασία δεδομένων

Το επόμενο βήμα της διαδικασίας είναι η χρήση του συνόλου δεδομένων που θα εκπαιδευτεί το μοντέλο μηχανικής μάθησης. Θα πρέπει να δημιουργηθούν υποσύνολα των δεδομένων μας, για την εκπαίδευση και την αξιολόγηση του μοντέλου.

Η συνάρτηση *fetch_20newsgroups()* της *sklearn* βιβλιοθήκης δίνει την δυνατότητα εύκολης δημιουργίας υποσυνόλων για εκπαίδευση και ελέγχου των μοντέλων.

Στη συνέχεια, πρέπει να γίνει η προετοιμασία των δεδομένων. Όπως έχει αναπτυχθεί και στο κεφάλαιο #ref datasets, τα δεδομένα είναι λεκτικά, με αποτέλεσμα να πρέπει να γίνει η μετατροπή τους σε διανύσματα. Η επίτευξη αυτού, γίνεται εύκολα με τη χρήση της συνάρτησης *TfidfVectorizer()*.

Χρήση μοντέλου μηχανικής μάθησης

Από τη στιγμή που τα δεδομένα έχουν επεξεργαστεί και προετοιμαστεί κατάλληλα ώστε να μπορούν οι αλγόριθμοι μηχανικής μάθησης να κάνουν την εξόρυξη των δεδομένων, το επόμενο βήμα είναι να ορίσουμε τον επιθυμητό αλγόριθμο μηχανικής μάθησης και να εκπαιδευτεί στα δεδομένα.

Αξιολόγηση Μοντέλου

Τέλος πρέπει να γίνει η αξιολόγηση του μοντέλου ώστε να δούμε αν προβλέπει με ακρίβεια ή όχι. Ο τρόπος που γίνεται η αξιολόγηση είναι να γίνουν προβλέψεις *(y_pred)* με τα δεδομένα αξιολόγησης *(X_test)* και στη συνέχεια να συγκριθούν με τα πραγματικά αποτελέσματα *(y_test)*, χρησιμοποιώντας τις συνάρτηση της *sklearn accuracy_score()*, *confusion_matrix()* και *classification_report()*.

4.2.3 SVM

Χρήση μοντέλου μηχανικής μάθησης

```
# Define model
svm_clf = SVC()

# Train model
svm_clf.fit(X_train.toarray(), y_train)

# Model predictions for X_train
svm_pred = svm_clf.predict(X_train.toarray())
```

Κώδικας 13 – Χρήση SVM μοντέλου

Ο αλγόριθμος μηχανικής μάθησης που θα χρησιμοποιηθεί σε αυτό το σημείο είναι ο SVM. Όπως και στην προηγούμενη ενότητα, έτσι και εδώ ο αλγόριθμος εκπαιδεύεται με τα δεδομένα εκπαίδευσης (X_train), γίνονται προβλέψεις με αυτά, όπου και αξιολογούνται. Στη συνέχεια, θα γίνουν προβλέψεις με τα τεστ δεδομένα(X_test), όπου και αυτά με τη σειρά τους αξιολογηθούν.

Αξιολόγηση Μοντέλου

```
# Compute train accuracy score
train_score = accuracy_score(y_train, svm_pred) * 100
print(f"Train accuracy score: {train_score:.2f}%")

# Model predictions for X_test
y_pred = svm_clf.predict(X_test.toarray())

# Compute train accuracy score
test_score = accuracy_score(y_test, y_pred) * 100
print(f"Test accuracy score: {test_score:.2f}%")
```

Κώδικας 14 – Αξιολόγηση προβλέψεων του SVM

Με το Κώδικα 17 μπορούμε να υπολογίσουμε την ακρίβεια της επίδοσης του μοντέλου στα δεδομένα εκπαίδευσης και στα τεστ δεδομένα. Η εκτέλεση του κώδικα παράγει το ακόλουθο αποτέλεσμα.

```
Train accuracy score: 99.95%
Test accuracy score: 88.62%
```

Εικόνα 22 – Αποτέλεσμα αξιολόγησης του SVM στα δεδομένα εκπαίδευσης

Η αξιολόγηση των προβλέψεων με τα σύνολα δεδομένων εκπαίδευσης έχουν τόσο υψηλό ποσοστό, διότι όπως είναι προφανές το μοντέλο έχει εκπαιδευτεί με αυτά τα δεδομένα. Σε άλλη περίπτωση αυτό το υψηλό ποσοστό είναι ένδειξη ότι το μοντέλο είναι overfitting.

Στη συνέχεια, ώστε να παραχθεί το confusion matrix της αποτελεσματικότητας των προβλέψεων του μοντέλου (*y_pred*) με τις πραγματικές τιμές των δεδομένων εκπαίδευσης (*y_train*), εκτελείται η ακόλουθη εντολή.

```
cm = confusion_matrix(y_train, svm_pred)
cm = array([[479,   0,   0,   1],
       [  0, 584,   0,   0],
       [  0,   0, 593,   0],
       [  0,   0,   0, 377]], dtype=int64)
```

Κώδικας 15 – Ορισμός του confusion matrix των προβλέψεων με τις πραγματικές τιμές των δεδομένων εκπαίδευσης

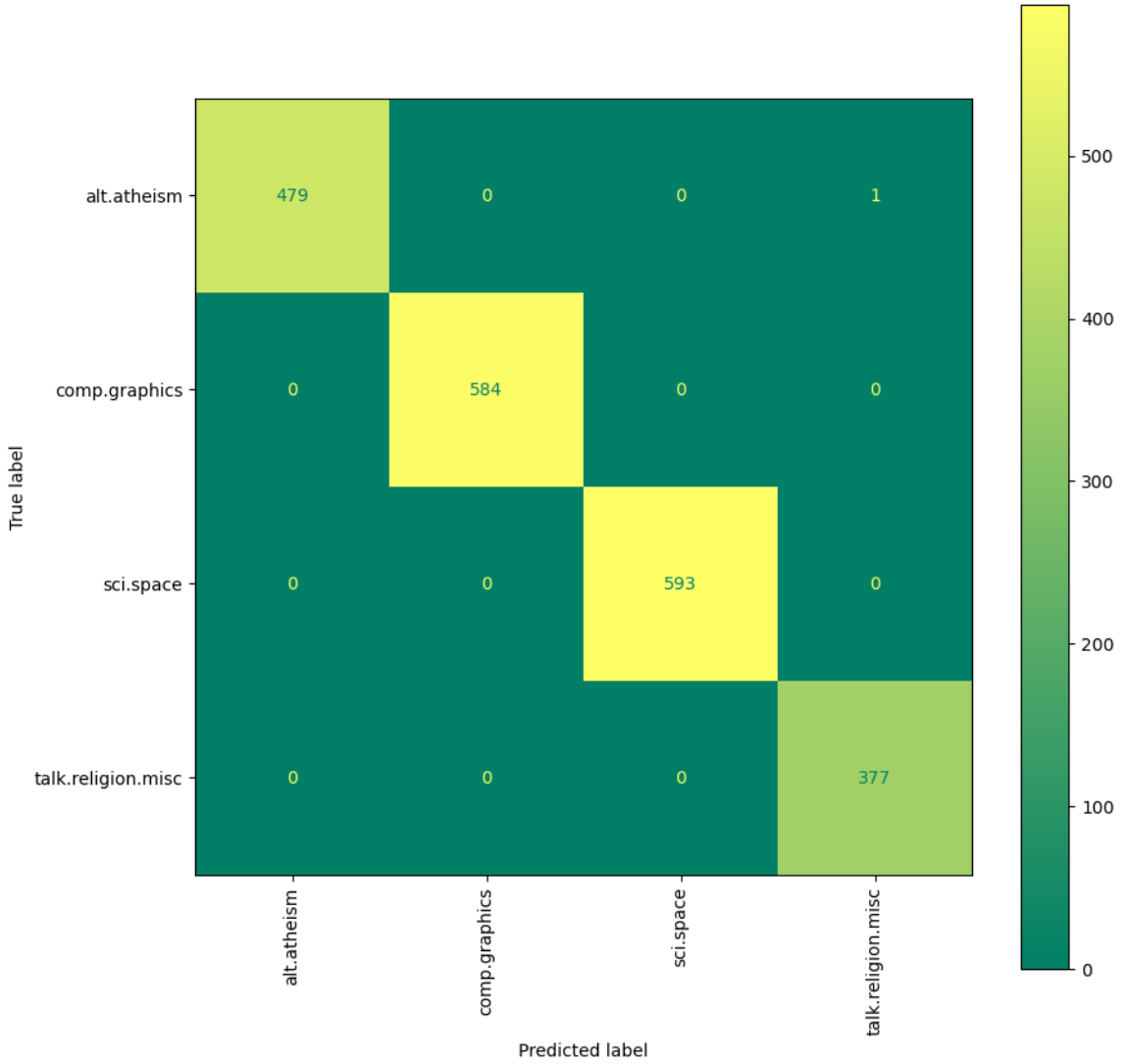
Από τη στιγμή που το έχουμε ορίσει το confusion matrix χρησιμοποιούμε την *ConfusionMatrixDisplay()* ώστε να παραχθεί το κατάλληλο διάγραμμα

```
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=data_train.target_names)
fig, ax = plt.subplots(figsize=(10,10))
disp = disp.plot(xticks_rotation='vertical', ax=ax, cmap='summer')

plt.show()
```

Κώδικας 16 – Κώδικας παραγωγής διαγράμματος Confusion Matrix των προβλέψεων με τις πραγματικές τιμές των δεδομένων εκπαίδευσης

Το αποτέλεσμα του κώδικα είναι



Εικόνα 23 – SVM Confusion matrix διάγραμμα των προβλέψεων με τις πραγματικές τιμές των δεδομένων εκπαίδευσης

Μελετώντας το διάγραμμα, παρατηρούμε ότι μόνο μια πρόβλεψη το μοντέλο είχε σφάλμα. Πρόβλεψε λάθος ότι η κατηγορία ήταν talk.religion.misc, ενώ ήταν alt.atheism. Όπου δικαιολογεί και το αποτέλεσμα του *accuracy_score*.

Επιπλέον με την εντολή

```
pd.DataFrame(classification_report(y_train, svm_pred, output_dict=True)).T
```

Κώδικας 17 – Παραγωγής Αναφοράς για τον SVM

Παράγεται ο εξής πίνακας:

	precision	recall	f1-score	support
alt.atheism	1.000000	0.997917	0.998957	480.000000
comp.graphics	1.000000	1.000000	1.000000	584.000000
sci.space	1.000000	1.000000	1.000000	593.000000
talk.religion.misc	0.997354	1.000000	0.998675	377.000000
accuracy	0.999508	0.999508	0.999508	0.999508
macro avg	0.999339	0.999479	0.999408	2034.000000
weighted avg	0.999510	0.999508	0.999508	2034.000000

Εικόνα 24 – Αποτέλεσμα του classification_report() των προβλέψεων με τις πραγματικές τιμές των δεδομένων εκπαίδευσης

Στο αποτέλεσμα του classification_report των συνόλων δεδομένων εκπαίδευσης, Φαίνεται ότι στην κατηγορία alt.atheism έχουμε σίγουρα μια πρόβλεψη όπου ήταν λάθος ως αρνητική, λόγο ότι η μετρική *recall* αυτής της κατηγορίας είναι 0.99. Αντίστοιχα η κατηγορία talk.religion.misc έχει στην μετρική *precision* ποσοστό 0.99. Με αυτά τα αποτελέσματα συμπεραίνουμε ότι μια πρόβλεψη ήταν λάθος ως κατηγορία talk.religion.misc ενώ κανονικά θα έπρεπε να ταξινομηθεί ως κατηγορία alt.atheism.

Ακολουθείτε η ίδια διαδικασία για την αξιολόγηση των προβλέψεων του μοντέλου (*svm_pred*) με τα δεδομένα ελέγχου (y_test). Πιο συγκεκριμένα ο κώδικας θα έχει την εξής δομή:

```
cm = confusion_matrix(y_test, y_pred)
cm = array([[259,  15,   7,  38],
       [  1, 384,   3,   1],
       [  1,  27, 365,   1],
       [ 37,  13,  10, 191]], dtype=int64)
```

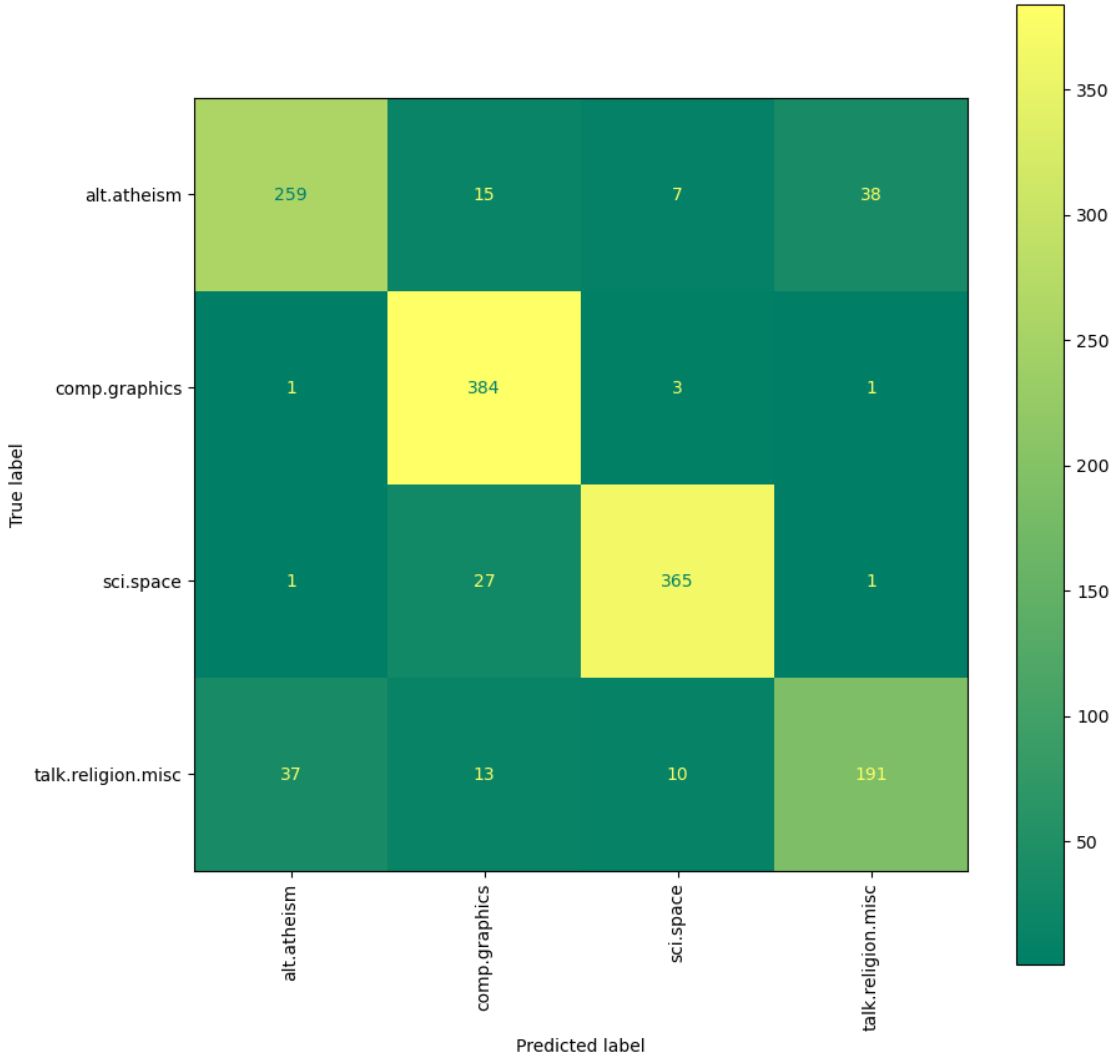
Κώδικας 18 - Ορισμός του confusion matrix των προβλέψεων με τις πραγματικές τιμές των δεδομένων ελέγχου

Όπως και προηγουμένως έτσι και τώρα για να μπορέσει να παραχθεί το διάγραμμα θα πρέπει να εκτελεστούν οι ακόλουθες εντολές:

```
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=data_train.target_names)
fig, ax = plt.subplots(figsize=(10,10))
disp = disp.plot(xticks_rotation='vertical', ax=ax, cmap='summer')
plt.show()
```

Κώδικας 19 - Κώδικας παραγωγής διαγράμματος Confusion Matrix των προβλέψεων με τις πραγματικές τιμές των δεδομένων ελέγχου

Το αποτέλεσμα που παράγεται είναι :



Εικόνα 25 - SVM Confusion matrix διάγραμμα των προβλέψεων με τις πραγματικές τιμές των δεδομένων ελέγχου

Και για να παραχθεί ο αντίστοιχος πίνακας με την αναφορά της κατηγοριοποίησης εκτελείτε ο κώδικας :

```
pd.DataFrame(classification_report(y_test, y_pred, output_dict=True)).T
```

Κώδικας 20 - Αποτέλεσμα του classification_report() των προβλέψεων με τις πραγματικές τιμές των δεδομένων ελέγχου

Όπου έχει το αποτέλεσμα:

	precision	recall	f1-score	support
alt.atheism	0.869128	0.811912	0.839546	319.000000
comp.graphics	0.874715	0.987147	0.927536	389.000000
sci.space	0.948052	0.926396	0.937099	394.000000
talk.religion.misc	0.826840	0.760956	0.792531	251.000000
accuracy	0.886179	0.886179	0.886179	0.886179
macro avg	0.879684	0.871603	0.874178	1353.000000
weighted avg	0.885872	0.886179	0.884530	1353.000000

Εικόνα 26 - Αποτέλεσμα του classification_report() των προβλέψεων με τις πραγματικές τιμές των δεδομένων ελέγχου

Μελετώντας τα αποτελέσματα των αξιολογήσεων των δεδομένων συνόλου ελέγχου, διακρίνεται ότι το ποσοστό είναι ικανοποιητικά ακριβές κατά μέσο όρο των προβλέψεών του. Παρατηρείται παρόλα αυτά μια αδυναμία, συγκεκριμένα στη κατηγορία talk.religion.misc. Όπου η μετρική recall έχει το μικρότερο ποσοστό 0.76 από όλες τις υπόλοιπες κατηγορίες. Επί της ουσίας, η κατηγορία talk.religion.misc έχει το μεγαλύτερο ποσοστό λανθασμένης απόρριψης ως πρόβλεψη. Αντιθέτως η κατηγορία sci.space έχει το μεγαλύτερο θετικό ποσοστό προβλέψεων.

4.2.4 K-NN

Χρήση μοντέλου μηχανικής μάθησης

```
# Define model
knn_model = KNeighborsClassifier(n_neighbors=1, metric='minkowski')

# Train model
knn_model.fit(X_train, y_train)

# Model predictions
y_pred = knn_model.predict(X_test)
```

Κώδικας 21 – Χρήση Μοντέλου

Σε αυτό το κομμάτι κώδικα ορίζουμε τον αλγόριθμο μηχανικής μάθησης που θα χρησιμοποιηθεί. Ακολουθεί η εκπαίδευσή του και τέλος η πρόβλεψη των δεδομένων αξιολόγησης.

Αξιολόγηση Μοντέλου

```


# Accuracy score
train_score = accuracy_score(y_test, y_pred) * 100

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)

# Display statistics
cr = classification_report(y_test, y_pred)
```

Κώδικας 22 – Αξιολόγηση Μοντέλου

Τέλος με τη χρήση των συναρτήσεων που αναγράφονται παραπάνω μπορούμε να έχουμε μια συνολική εικόνα για την επίδοση του αλγορίθμου μηχανικής μάθησης. Τα αποτελέσματα που έχουμε από τις συναρτήσεις είναι τα εξής:

```
train_score = 85.88322246858833
```

Εικόνα 27 – Αποτέλεσμα της συνάρτησης accuracy_score

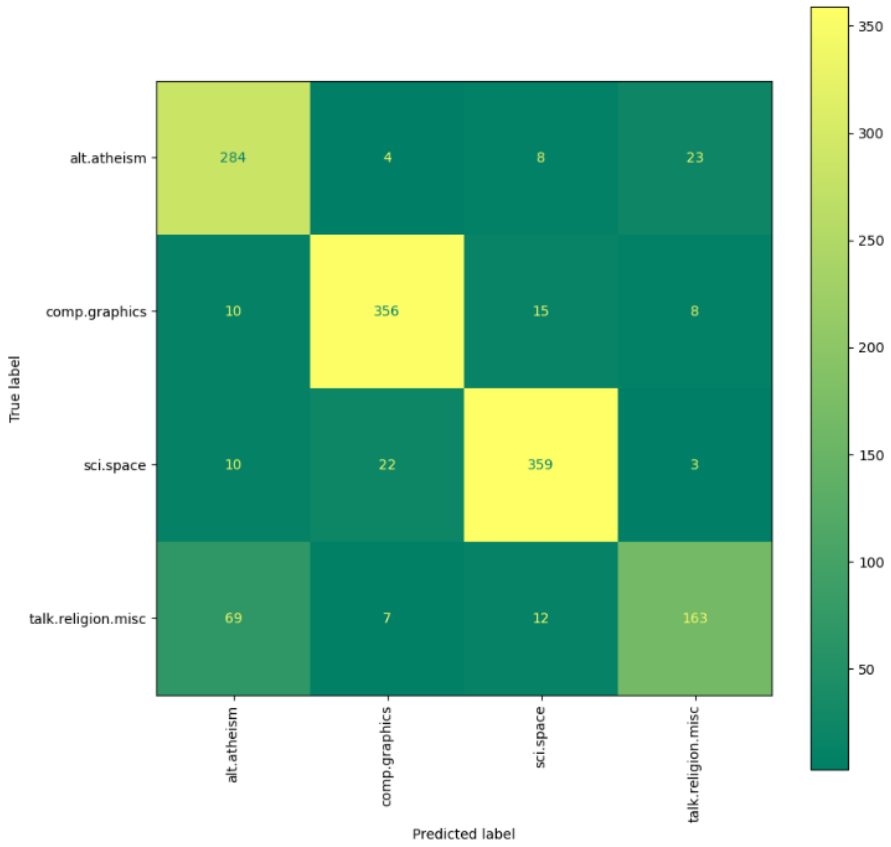
Η ακρίβεια του μοντέλου είναι 85%

```
cm = array([[258,  11,   9,  41],
       [ 14, 329,  30,  16],
       [ 11,  31, 350,   2],
       [ 47,  12,  14, 178]], dtype=int64)

# Display Confusion Matrix
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=data_train.target_names)

fig, ax = plt.subplots(figsize=(10,10))
disp = disp.plot(xticks_rotation='vertical', ax=ax, cmap='summer')
plt.show()
```

Κώδικας 23 – Δημιουργία του Confusion matrix



Εικόνα 28 – Confusion Matrix

Σε αυτό το διάγραμμα γίνεται πιο ξεκάθαρη η κατανομή των αποτελεσμάτων. Για παράδειγμα, φαίνεται ότι η κατηγορία *comp.grapshics* έχει πολύ μεγαλύτερο ποσοστό επιτυχίας από ότι η κατηγορία *talk.religion.misc*.

```
cr =  precision    recall  f1-score   support

       alt.atheism       0.76      0.89      0.82       319
     comp.graphics       0.92      0.92      0.92       389
         sci.space       0.91      0.91      0.91       394
talk.religion.misc       0.83      0.65      0.73       251

          accuracy                           0.86      1353
         macro avg       0.85      0.84      0.84      1353
      weighted avg       0.86      0.86      0.86      1353
```

Εικόνα 29 – Αποτέλεσμα της συνάρτησης classification_report()

Σύμφωνα με την αναφορά των μετρικών, μπορούμε να συμπεράνουμε ότι τα περισσότερα λάθη έγιναν μεταξύ των κατηγοριών alt.atheism και talk.religion.misc. Όπου λανθασμένα το μοντέλο ταξινόμησε την πρόβλεψη ως κατηγορία alt.atheism αντί κατηγορία talk.religion.misc.

4.3 Συνδυασμός Μεθόδων

Σε αυτήν την ενότητα θα γίνει η χρήση του αλγορίθμου Προσομοιωμένης Ανόπτησης (Simulated Annealing) για να βρεθεί η βέλτιστη τιμή της υπερ-παραμέτρου n_neighbors του αλγορίθμου k-NN, ώστε να έχουμε την καλύτερη δυνατή ακρίβεια στην επίδοση του μοντέλου. Κανονικά η επιλογή αυτής της μεταβλητής γίνεται εμπειρικά και με συνεχές δοκιμές από τους μηχανικούς που διαχειρίζονται το μοντέλο. Επομένως με τη χρήση του αλγορίθμου Προσομοιωμένης Ανόπτησης, βελτιστοποιείται η διαδικασία επιλογής αυτής της υπερ-παραμέτρου.

Ώστε να μπορέσει να χρησιμοποιηθεί ο αλγόριθμος Προσομοιωμένης Ανόπτησης στην βελτιστοποίηση του μοντέλου θα πρέπει να προσαρμοστεί ο κώδικας κατάλληλα, σύμφωνα με τις ανάγκες του κώδικα.

Βασική αλλαγή που πρέπει να γίνει στον αλγόριθμο της Προσομοιωμένης Ανόπτησης είναι ότι οι υποψήφιες λύσεις πρέπει πλέον να είναι ακέραιος, διότι η μεταβλητή n_neighbors είναι ακέραιος αριθμός.

Μια ακόμα απαραίτητη προσαρμογή στον αλγόριθμό είναι η αντικειμενική συνάρτηση. Πλέον η αντικειμενική συνάρτηση θα περιλαμβάνει την εκπαίδευση του μοντέλου με την υποψήφια λύση, όπου είναι ο αριθμός των γειτονιών που θα χειριστεί ο αλγόριθμος k-NN. Το αποτέλεσμα που θα επιστρέφει η αντικειμενική συνάρτηση είναι η μονάδα μείον την ακρίβεια του μοντέλου. Η αιτία που επιστρέφει με αυτόν τον τρόπο το κόστος της αντικειμενικής συνάρτησης είναι ότι το συγκεκριμένο πρόβλημα αντιμετωπίζεται ως πρόβλημα ελαχιστοποίησης. Αν χρειαζόταν να αντιμετωπιστεί ως πρόβλημα μεγιστοποίησης τότε, η αντικειμενική συνάρτηση θα επέστρεφε απλά τον αριθμό της ακρίβειας του μοντέλου και θα αλλάζαμε τον τελεστή της συνθήκης *candidate_eval < best_eval σε candidate_eval > best_eval.*

4.3.1 Κώδικας

Αρχικά πρέπει να ορίσουμε τις βιβλιοθήκες που θα χρησιμοποιήσουμε για την υλοποίηση του κώδικα, όπως φαίνεται και στον ακόλουθο κώδικα.

```
from numpy import asarray
from numpy import exp
from numpy.random import randn
from numpy.random import rand
from numpy.random import seed

# general imports
import pandas as pd
import matplotlib.pyplot as plt

# sklearn imports
from sklearn.datasets import fetch_20newsgroups
from sklearn.decomposition import TruncatedSVD
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import Normalizer

from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
from sklearn.metrics import ConfusionMatrixDisplay
```

Κώδικας 24 – Εισαγωγή Βιβλιοθηκών

Χρησιμοποιούμε τις συναρτήσεις του numpy κυρίως ώστε να μπορέσουμε να παράγουμε τυχαίους αριθμούς. Απαραίτητη διαδικασία για την αρχικοποίηση της υποψήφιας τιμής την αντικειμενικής συνάρτησης και στον υπολογισμό της επόμενης υποψήφιας τιμής.

Στο επόμενο κομμάτι κώδικα ορίζουμε τη συνάρτηση που θα χρησιμοποιηθεί, ώστε να προετοιμάσει τα σύνολα εκπαίδευσης και ελέγχου.

```
def DataPreperation(categories):
    data_train = fetch_20newsgroups(subset='train', categories=categories,
random_state=42)
    # Convert to TF-DF format
    vectorizer = TfidfVectorizer(max_df=0.5, min_df=2,
stop_words='english', use_idf=True)
    X_train = vectorizer.fit_transform(data_train.data)
    # Reduce dimensions
    n_components = 5
    svd = TruncatedSVD(n_components)
    normalizer = Normalizer(copy=False)
    y_train = data_train.target

    data_test = fetch_20newsgroups(subset='test', categories=categories,
random_state=42)
    X_test = vectorizer.transform(data_test.data)
    y_test = data_test.target
    return (X_train, y_train, X_test, y_test)
```

Κώδικας 25 – Προετοιμασία Δεδομένων

Επιπλέον θα ορίσουμε και την αντικειμενική συνάρτηση.

```
# objective function
def objective(X_train, y_train, X_test, y_test, candidate):
    knn_model = KNeighborsClassifier(n_neighbors=candidate,
metric='minkowski')
    knn_model.fit(X_train, y_train)
    y_pred = knn_model.predict(X_test)
    train_score = (1 - accuracy_score(y_test, y_pred)) * 100
    return train_score
```

Κώδικας 26 – Αντικειμενική συνάρτηση

Τέλος ορίζουμε και την συνάρτηση της Προσομοιωμένης Ανόπτησης.

```
# simulated annealing algorith
def simulated_annealing(objective, bounds, n_iterations, step_size, temp):
    # generate an initial point
    # best = 5
    best = bounds[:, 0] + rand(len(bounds)) * (bounds[:, 1] - bounds[:,
0])
    best = int(round(best[0]))
    # init categories of dataset
    categories = ['alt.atheism', 'talk.religion.misc', 'comp.graphics',
'sci.space']
    # init data
    X_train, y_train, X_test, y_test = DataPreperation(categories)
    # evaluate the initial point
    best_eval = objective(X_train, y_train, X_test, y_test, best)
    # current working solution
    curr, curr_eval = best, best_eval
    candidate = curr
    scores = list()

    # run the algorith
    for i in range(n_iterations):
        # take step
        random_value = randn(len(bounds))
        candidate = curr + random_value * step_size
        candidate = int(round(candidate[0]))
        # evaluate candidate point
        candidate_eval = objective(X_train, y_train, X_test, y_test,
candidate)
        # check for new best solution
        if candidate_eval < best_eval:
            # store new best point
            best, best_eval = candidate, candidate_eval
            # keep track of scores
            scores.append(best_eval)
            # report progress
            print('>%d f(%s) = %.5f' % (i, best, best_eval))
            # difference between candidate and current point evaluation
            diff = candidate_eval - curr_eval
            # calculate temperature for each epoch
            t = temp / float(i + 1)
            # calculate metropolis acceptance criterion
            metropolis = exp(-diff/t)
            # check if we should keep the new point
            if diff < 0 or rand() < metropolis:
```

```
                # store the new current point
                curr, curr_eval = candidate, candidate_eval
    return [best, best_eval, scores]
```

Κώδικας 27 – Επεξεργασμένος Αλγόριθμος Προσομοιωμένης Ανόπτησης

Όπως σχολιάστηκε στην εισαγωγή, στον παραπάνω κώδικα έχουν γίνει αλλαγές στον τύπο των υποψήφιων τιμών, όπου είναι πλέον ακέραιοι αριθμοί. Επιπλέον έχει αλλάξει η αντικειμενική συνάρτηση και τα ορίσματά της.

Στη συνέχεια, ορίζονται το εύρος των διαθέσιμων τιμών των υποψήφιων τιμών, ο αριθμός των επαναλήψεων που θα γίνουν μέσα στον αλγόριθμο Προσομοιωμένης Ανόπτησης, το μέγεθος του βήματος που θα γίνεται σε κάθε επανάληψη και η αρχική θερμοκρασία.

```
# Init Variables
# seed the pseudorandom number generator
seed(1)
# define range for input
bounds = asarray([[1,100]], dtype=int)
# define the total iterations
n_iterations = 150
# n_iterations = 600
# define the maximum step size
step_size = 3.6
# initial temperature
temp = 10
```

Κώδικας 28 – Αρχικοποίηση παραμέτρων του αλγορίθμου Προσομοιωμένης Ανόπτησης

Σε αυτό το σημείο να σημειωθεί ότι έγιναν πολλές δοκιμές, ώστε να οριστεί το μέγεθος του βήματος της υποψήφιας τιμής. Διότι αν η τιμή είναι μικρή ή πολύ μεγάλη τότε ο μεθευρετικός αλγόριθμος εγκλωβίζεται σε τοπικό βέλτιστο.

Αυτό που μένει να γίνει είναι να εκτελεστεί ο κώδικας, ώστε να πάρουμε τη βέλτιστη τιμή για τη μεταβλητή k_neighbor του αλγορίθμου k-NN.

```
# perform the simulated annealing search
best, score, scores = simulated_annealing(objective, bounds, n_iterations,
step_size, temp)
```

Κώδικας 29 – Εκτέλεση Αλγορίθμου Προσομοιωμένης Ανόπτησης

Αφ' όσον έχει ολοκληρωθεί η εκτέλεση του αλγορίθμου έχουμε πλέον τις βέλτιστες τιμές.

```
best, score, scores = (30,
 13.8950480413895,
 [14.486326681448636, 14.116777531411673, 13.8950480413895])
```

Κώδικας 30 – Εμφάνιση Βέλτιστων τιμών

Σύμφωνα με τα αποτελέσματα που έχουμε, η πιο αποτελεσματική τιμή για την υπερ-παράμετρο n_neighbors είναι ίση με 30. Και το αποτέλεσμα της ακρίβειας είναι :

$$accuracy = 100 - error\_rate$$

Όπου σε αυτή περίπτωση το error_rate είναι ίσο με 13.8950480413895. Άρα η ακρίβεια είναι ίση με 86.10%.

Ώστε να επιβεβαιωθεί το αποτέλεσμα της ακρίβειας θα πρέπει πρώτα να εκπαιδεύσουμε το μοντέλο k-NN με την τιμή n_neighbors = 30.

```
categories = ['alt.atheism', 'talk.religion.misc', 'comp.graphics',
'sci.space']
# init data
X_train, y_train, X_test, y_test = DataPreperation(categories)
knn_model = KNeighborsClassifier(n_neighbors=best, metric='minkowski')
knn_model.fit(X_train, y_train)
y_pred = knn_model.predict(X_test)
```

Κώδικας 31 – Εκπαίδευση k-NN με τη βέλτιστη τιμή

Από τη στιγμή που έχει ολοκληρωθεί η εκτέλεση το παραπάνω κώδικα, θα μπορέσει να παραχθεί μια αναφορά σχετικά με την επίδοση του μοντέλου. Όπως αναφέρθηκε και σε προηγούμενες ενότητες, ένας τρόπος για να παραχθεί η αναφορά του μοντέλου είναι με την εξής εντολή:

```
pd.DataFrame(classification_report(y_test, y_pred, output_dict=True)).T
```

Κώδικας 32 – Παραγωγή Αναφοράς Μοντέλου k-NN

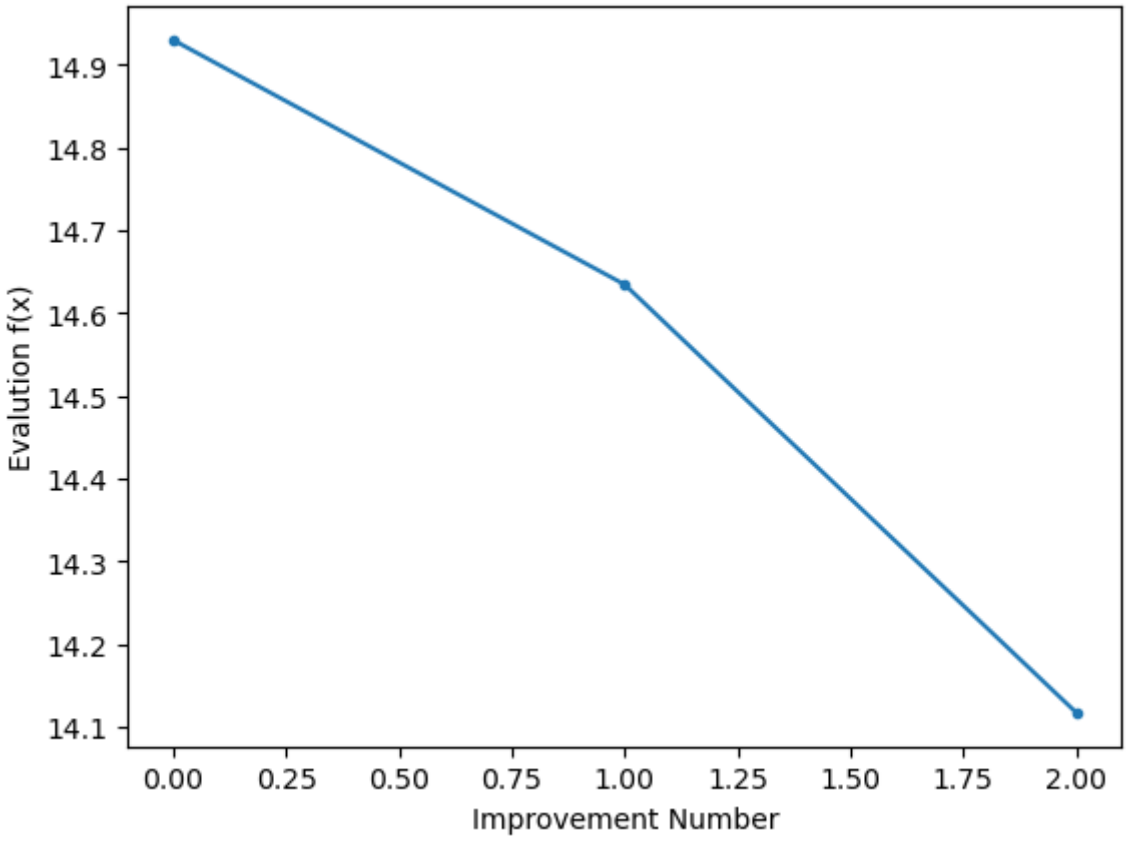
Όπου το αποτέλεσμα της εντολής είναι :

	precision	recall	f1-score	support
alt.atheism	0.765499	0.890282	0.823188	319.00000
talk.religion.misc	0.917313	0.912596	0.914948	389.00000
comp.graphics	0.907268	0.918782	0.912989	394.00000
sci.space	0.836735	0.653386	0.733781	251.00000
accuracy	0.861050	0.861050	0.861050	0.86105
macro avg	0.856704	0.843762	0.846227	1353.00000
weighted avg	0.863646	0.861050	0.859134	1353.00000

Εικόνα 30 – Αποτέλεσμα αναφοράς Μοντέλου k-NN

Όπως φαίνεται και στα αποτελέσματα της αναφοράς η ακρίβεια είναι όσο υπολογίστηκε παραπάνω.

Επιπλέον παράγοντας το διάγραμμα στην Εικόνα 31, είναι ξεκάθαρο το πόσο γρήγορα βρήκε ο αλγόριθμος τη βέλτιστη τιμή.



Εικόνα 31 – Βελτιστοποίηση τιμής ανά επανάληψη

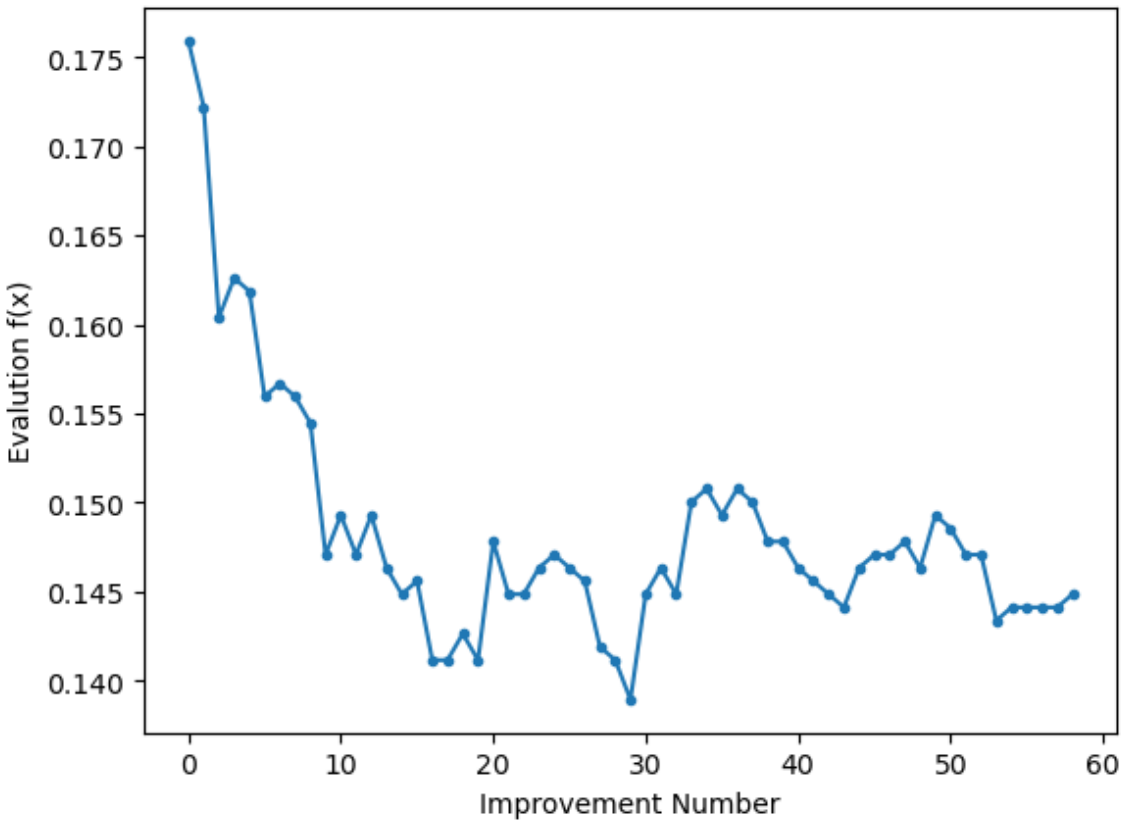
Σε αυτό το σημείο θα πρέπει να επιβεβαιωθεί ότι όντως ο αλγόριθμος Προσομοιωμένης Ανόπτησης επιστρέφει τη βέλτιστη τιμή που μπορεί πριν τη ολοκλήρωση των συνολικών επαναλήψεων.

Ο τρόπος που θα υλοποιηθεί ώστε να ελεγχθεί η επίδοση του μεθευρετικού αλγορίθμου είναι απλός. Θα γίνει μια προσπέλαση σε όλες τις ακέραιες τιμές της υπερ-παραμέτρου n_neighbors από το 1 μέχρι το 60 και απλά θα επιστραφεί η μικρότερη τιμή. Ο λόγος που έχει επιλεγεί το εύρος των τιμών να είναι μεταξύ 1 και 60, είναι ότι είναι πολύ λεπτή η γραμμή αν είναι αποδοτικό να αυξήσουμε τις γειτονίες πάνω από το 50, διότι αυξάνεται πάρα πολύ η περιπλοκότητα σε σημείο να υπάρχει το δίλημμα αν θέλουμε να έχουμε καλύτερη επίδοση. Προφανώς ανάλογα το πρόβλημα αλλάζει και το κριτήριο της επιλογής. Στη ζητούμενη περίπτωση δεν θεωρείται πως χρειάζεται να αυξηθεί περισσότερο η περιπλοκότητα του μοντέλου. Οπότε η σύνταξη του κώδικα είναι η εξής :

```
reports = []
best = 1
knn_model = KNeighborsClassifier(n_neighbors=best, metric='minkowski')
knn_model.fit(X_train, y_train)
y_pred = knn_model.predict(X_test)
best_eval = 1- accuracy_score(y_test, y_pred)
for k in range(1,60):
    knn_model = KNeighborsClassifier(n_neighbors=k, metric='minkowski')
    knn_model.fit(X_train, y_train)
    y_pred = knn_model.predict(X_test)
    accuracy = 1- accuracy_score(y_test, y_pred)
    reports.append(accuracy)
    if accuracy < best_eval:
        best, best_eval = k, accuracy
```

Κώδικας 33 – Αξιολόγηση αποτελέσματος

Όπου το best ισούται με 30. Τέλος για να είναι ακόμα πιο ξεκάθαρο, παράγεται το ακόλουθο διάγραμμα με τον εξής κώδικα:



Εικόνα 32 – Διάγραμμα ποιότητας ως προς n_neighbors

Όπως γίνεται ακόμα πιο φανερό από το διάγραμμα η βέλτιστη τιμή της υπερ-παραμέτρου σωστά βρέθηκε από το μεθευρετικό αλγόριθμο Προσομοιωμένης Ανώπτησης.

5 Συμπεράσματα

Μελετήθηκαν σε αυτή την εργασία οι βασικές κατηγορίες στους αλγόριθμους μηχανικής μάθησης. Όπου είναι οι επιβλεπόμενοι αλγόριθμοι, οι μη-επιβλεπόμενοι και οι ενισχυμένης διδασκαλίας. Επιπλέον αναφέρονται οι τρόποι όπου μπορεί να βελτιστοποιηθεί ένας αλγόριθμος μηχανικής μάθησης σε προβλήματα κατηγοριοποίησης με τη χρήση των μεθευρετικών αλγορίθμων.

Στη συνέχεια αναπτύχθηκε πειραματικός κώδικας, όπου έγινε ανάπτυξη του αλγορίθμου της Προσομοιωμένης Ανόπτησης (*Simulated Annealing*) σε ένα απλό παράδειγμα και έγινε η απαραίτητη επεξεργασία των δεδομένων και εκπαίδευση του SVM και k-NN μοντέλου.

Τέλος με τη βοήθεια της Προσομοιωμένης Ανόπτησης βρέθηκε η βέλτιστη τιμή της υπερ-παραμέτρου n_neighbors του k-NN αλγορίθμου μηχανικής μάθησης, ώστε να μοντέλο να είναι όσο πιο αποτελεσματικό γίνεται.

Μελλοντικές Εργασίες

Η εργασία αυτή ανοίγει πολλές δυνατότητες για μελλοντικές έρευνες. Μερικές πιθανές κατευθύνσεις περιλαμβάνουν:

1. **Επέκταση σε Άλλα Αλγοριθμικά Μοντέλα**: Μπορεί να εξεταστεί η εφαρμογή μεθευρετικών αλγορίθμων σε άλλα αλγοριθμικά μοντέλα για την ταξινόμηση δεδομένων και τη βελτίωση της απόδοσης.

2. **Βελτίωση της Αποτελεσματικότητας**: Επιπλέον ερευνητική εργασία μπορεί να επικεντρωθεί στη βελτίωση της αποτελεσματικότητας και της επίδοσης των μεθευρετικών αλγορίθμων.

3. **Εφαρμογές σε Πραγματικά Προβλήματα**: Η μελλοντική έρευνα μπορεί να εστιαστεί στην εφαρμογή των αλγορίθμων σε πραγματικά προβλήματα και σενάρια επιχειρηματικής εφαρμογής.

4. **Εξέλιξη των Μεθευρετικών**: Οι μεθευρετικοί αλγόριθμοι εξελίσσονται συνεχώς. Μπορεί να εξεταστεί η χρήση πιο προηγμένων εκδοχών αυτών των αλγορίθμων ή η ανάπτυξη νέων μεθοδολογιών.

Αυτές οι προτάσεις ανοίγουν νέες προοπτικές για την έρευνα στον τομέα της βελτιστοποίησης της ταξινόμησης δεδομένων με χρήση μεθευρετικών αλγορίθμων και αποτελούν τη βάση για μελλοντική επέκταση και βελτίωση της έρευνας σε αυτόν τον σημαντικό τομέα.

6 Βιβλιογραφία

[1] A. H. Gandomi, X.-S. Yang, S. Talatahari, and A. H. Alavi, "Metaheuristic algorithms in modeling and optimization," *Metaheuristic applications in structures and infrastructures*, vol. 1, pp. 1–24, 2013.

[2] S. S. Alaoui, Y. Labsiv, and B. Aksasse, "Classification algorithms in data mining," *Int. J. Tomogr. Simul*, vol. 31, no. 4, pp. 34–44, 2018.

[3] T. Taslim, E. Sabna, and K. W. Ningsih, "Optimization of K Value at K Nearest Neighbor for Classification and Prediction of Healing in Covid-19 Patient," *J Pharm Negat Results*, vol. 13, no. 4, pp. 1699–1708, 2022.

[4] S. Sharma and V. Kumar, "A comprehensive review on multi-objective optimization techniques: Past, present and future," *Archives of Computational Methods in Engineering*, vol. 29, no. 7, pp. 5605–5633, 2022.

[5] S. M. Almufti, A. A. Shaban, Z. A. Ali, R. I. Ali, and J. A. Dela Fuente, "Overview of Metaheuristic Algorithms," *Polaris Global Journal of Scholarly Research and Trends*, vol. 2, no. 2, pp. 10–32, 2023.

[6] M. Affi, H. Derbel, B. Jarboui, and P. Siarry, "A Skewed General Variable Neighborhood Search Approach for Solving the Battery Swap Station Location-

Routing Problem with Capacitated Electric Vehicles," *Green Transportation and New Advances in Vehicle Routing Problems*, pp. 75–89, 2020.

[7] N. Pourkhodabakhsh, M. M. Mamoudan, and A. Bozorgi-Amiri, "Effective machine learning, Meta-heuristic algorithms and multi-criteria decision making to minimizing human resource turnover," *Applied Intelligence*, vol. 53, no. 12, pp. 16309–16331, 2023.

[8] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*, vol. 4, no. 4. Springer, 2006.

[9] M. Bansal, A. Goyal, and A. Choudhary, "A comparative analysis of K-nearest neighbor, genetic, support vector machine, decision tree, and long short term memory algorithms in machine learning," *Decision Analytics Journal*, vol. 3, p. 100071, 2022.

[10] M. Karimi-Mamaghan, M. Mohammadi, P. Meyer, A. M. Karimi-Mamaghan, and E.-G. Talbi, "Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: A state-of-the-art," *Eur J Oper Res*, vol. 296, no. 2, pp. 393–422, 2022.

[11] V. Nasteski, "An overview of the supervised machine learning methods," *Horizons. b*, vol. 4, pp. 51–62, 2017.

[12] A. Viloria, N. A. L. Zelaya, and N. Varela, "Unsupervised learning algorithms applied to grouping problems," *Procedia Comput Sci*, vol. 175, pp. 677–682, 2020.

[13] E. Kuiper, E. Constantinides, S. A. de Vries, R. F. Marinescu-Muster, and F. Metzner, "A framework of unsupervised machine learning algorithms for user profiling," in *48th Annual European Marketing Academy (EMAC) Conference*, 2019.

[14] B. Smith, R. Abay, J. Abbey, S. Balage, M. Brown, and R. Boyce, "Propulsionless planar phasing of multiple satellites using deep reinforcement learning," *Advances in Space Research*, vol. 67, no. 11, pp. 3667–3682, 2021.

[15] K. Gourav and A. Kaur, "A study of reinforcement learning applications & its algorithms," *International Journal of Scientific & Technology Research*, vol. 9, no. 3, pp. 4223–4228, 2020.

[16] B. Jang, M. Kim, G. Harerimana, and J. W. Kim, "Q-learning algorithms: A comprehensive classification and applications," *IEEE access*, vol. 7, pp. 133653–133667, 2019.

[17] G. C. Okwuibe, J. Bhalodia, A. S. Gazafroudi, T. Brenner, P. Tzscheutschler, and T. Hamacher, “Intelligent Bidding Strategies for Prosumers in Local Energy Markets Based on Reinforcement Learning,” *IEEE Access*, vol. 10, pp. 113275–113293, 2022.

[18] S. Li, X. Yuan, and J. Niu, “Robotic Peg-in-Hole Assembly Strategy Research Based on Reinforcement Learning Algorithm,” *Applied Sciences*, vol. 12, no. 21, p. 11149, 2022.

[19] H. Jiang, R. Gui, Z. Chen, L. Wu, J. Dang, and J. Zhou, “An Improved Sarsa (\lambda) Reinforcement Learning Algorithm for Wireless Communication Systems,” *IEEE Access*, vol. 7, pp. 115418–115427, 2019.

[20] L. Moro, M. G. A. Paris, M. Restelli, and E. Prati, “Quantum compiling by deep reinforcement learning,” *Commun Phys*, vol. 4, no. 1, p. 178, 2021.

[21] C. R. Reeves, “Genetic algorithms,” *Handbook of metaheuristics*, pp. 109–139, 2010.

[22] J. Brownlee, *Optimization for Machine Learning*. Machine Learning Mastery, 2021. [Online]. Available: https://books.google.gr/books?id=tW1HEAAAQBAJ

[23] M. Aci, C. Inan, and M. Avci, “A hybrid classification method of k nearest neighbor, Bayesian methods and genetic algorithm,” *Expert Syst Appl*, vol. 37, no. 7, pp. 5061–5067, 2010.

[24] Z. F. Hussain *et al.*, “A new model for iris data set classification based on linear support vector machine parameter’s optimization,” *International Journal of Electrical and Computer Engineering*, vol. 10, no. 1, p. 1079, 2020.

[25] R. S. Olson and J. H. Moore, “TPOT: A tree-based pipeline optimization tool for automating machine learning,” in *Workshop on automatic machine learning*, 2016, pp. 66–74.

[26] G. Guo, H. Wang, D. Bell, Y. Bi, and K. Greer, “KNN model-based approach in classification,” in *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2003, Catania, Sicily, Italy, November 3-7, 2003. Proceedings*, 2003, pp. 986–996.

[27] K. Taunk, S. De, S. Verma, and A. Swetapadma, “A brief review of nearest neighbor algorithm for learning and classification,” in *2019 international conference on intelligent computing and control systems (ICCS)*, 2019, pp. 1255–1260.

[28] S. Shekhar, K. Kartikey, and A. Arya, “Integrating decision trees with metaheuristic search optimization algorithm for a student’s performance prediction,” in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2020, pp. 655–661.

[29] J. Fan, L. Fang, J. Wu, Y. Guo, and Q. Dai, “From brain science to artificial intelligence,” *Engineering*, vol. 6, no. 3, pp. 248–252, 2020.

[30] “Accuracy Score,” https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html#:~:text=sklearn.metrics.accuracy_score%28y_true%2C%20y_pred%2C%20,more%20in%20the%20User%20Guide.

[31] “Precision,” https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_score.html.

[32] “Recall,” https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html.

[33] “F1-Score,” https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html.