ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΔΙΑΤΜΗΜΑΤΙΚΟ ΜΕΤΑΠΤΥΧΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΠΛΗΡΟΦΟΡΙΚΗ ΚΑΙ ΥΠΟΛΟΓΙΣΤΙΚΗ ΒΙΟΙΑΤΡΙΚΗ
ΚΑΤΕΥΘΥΝΣΗ

«ΠΛΗΡΟΦΟΡΙΚΗ ΜΕ ΕΦΑΡΜΟΓΕΣ ΣΤΗΝ ΑΣΦΑΛΕΙΑ, ΔΙΑΧΕΙΡΙΣΗ ΜΕΓΑΛΟΥ ΟΓΚΟΥ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΠΡΟΣΟΜΟΙΩΣΗ»

Ευφυής διαχείριση ερωτημάτων σε κατανεμημένους κόμβους με χρήση τεχνικών Μηχανικής Μάθησης

ΓΕΩΡΓΙΟΣ ΡΑΧΟΥΤΗΣ , Α.Μ:00744

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Επιβλέπων Καθηγητής: κ. Κωνσταντίνος Κολομβάτσος

Λαμία, Ιούνιος 2023

UNIVERSITY OF THESSALY

SCHOOL OF SCIENCE

INFORMATICS AND COMPUTATIONAL BIOMEDICINE

Intelligent query management in distributed nodes using machine learning techniques

GEORGIOS RACHOUTIS

Master thesis

Name of Supervisor: Mr. Konstantinos Kolomvatsos

Lamia, June 2023

«Υπεύθυνη Δήλωση μη λογοκλοπής και ανάληψης προσωπικής ευθύνης»

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, και γνωρίζοντας τις συνέπειες της λογοκλοπής, δηλώνω υπεύθυνα και ενυπογράφως ότι η παρούσα εργασία με τίτλο «Ευφυής διαχείριση ερωτημάτων σε κατανεμημένους κόμβους με χρήση τεχνικών Βαθιάς Μάθησης» αποτελεί προϊόν αυστηρά προσωπικής εργασίας και όλες οι πηγές από τις οποίες χρησιμοποίησα δεδομένα, ιδέες, φράσεις, προτάσεις ή λέξεις, είτε επακριβώς (όπως υπάρχουν στο πρωτότυπο ή μεταφρασμένες) είτε με παράφραση, έχουν δηλωθεί κατάλληλα και ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή. Αναλαμβάνω πλήρως, ατομικά και προσωπικά, όλες τις νομικές και διοικητικές συνέπειες που δύναται να προκύψουν στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής.

Ο ΔΗΛΩΝ

Ημερομηνία 07/06/2023

Υπογραφή

Περιεχόμενα

Η παρούσα διπλωματική εργασία εκπονήθηκε στο πλαίσιο των απαιτήσεων του Μεταπτυχιακού Προγράμματος Σπουδών του τμήματος Πληροφορικής με Εφαρμογές στη Βιοϊατρική της σχολής Θετικών Επιστημών του Πανεπιστήμιου Θεσσαλίας. Η έγκρισή της δεν υποδηλώνει απαραιτήτως και την αποδοχή των απόψεων του συγγραφέα εκ μέρους του Πανεπιστήμιου Θεσσαλίας.

ΠΡΟΛΟΓΟΣ

Η παρούσα διπλωματική εργασία εκπονήθηκε στο πλαίσιο των απαιτήσεων του Μεταπτυχιακού Προγράμματος Σπουδών του τμήματος Πληροφορικής με Εφαρμογές στη Βιοϊατρική της σχολής Θετικών Επιστημών του Πανεπιστήμιου Θεσσαλίας. Έχει τίτλο "Ευφυής διαχείριση ερωτημάτων σε κατανεμημένους κόμβους με χρήση τεχνικών Βαθιάς Μάθησης". Η εργασία μου ανατέθηκε από τον Καθηγητή κ. Κολομβάτσο ο οποίος ήταν υπεύθυνος για την παρακολούθηση της προόδου.

Η εργασία αυτή αποτελεί μία προσέγγιση ως προς τη διαχείριση ερωτημάτων σε κατανεμημένους κόμβους μέσω της χρήσης κατάλληλου λογισμικού, το οποίο έχει αναπτυχθεί στο πλαίσιο της εργασίας, και το οποίο κάνει χρήση τεχνικών και αλγορίθμων μηχανικής μάθησης.

Στην προσπάθεια δημιουργίας του κατάλληλου αλγορίθμου και εύρεσης του αντίστοιχου υλικού για την εργασία θα ήθελα να ευχαριστήσω από καρδιάς τον υπεύθυνο καθηγητή κ. Κωνσταντίνο Κολομβάτσο για τη πολύτιμη καθοδήγησή του. Επιπλέον, θα ήθελα να ευχαριστήσω την οικογένεια μου για την κατανόηση και υπομονή που έδειξαν καθώς και για την στήριξη τους στη προσπάθεια μου αυτή.

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΠΕΡΙΛΗΨΗ

Το Διαδίκτυο των Πραγμάτων (Internet of Things, IoT) δημιουργεί τεράστιους όγκους δεδομένων από εκατομμύρια συσκευές. Η μηχανική μάθηση τροφοδοτείται από δεδομένα και δημιουργεί πληροφορίες από αυτά. Η μηχανική μάθηση χρησιμοποιεί τη συμπεριφορά του παρελθόντος για τον εντοπισμό προτύπων και δημιουργεί μοντέλα που βοηθούν στην πρόβλεψη μελλοντικής συμπεριφοράς και γεγονότων. Η μηχανική μάθηση μπορεί να βοηθήσει στην αποκάλυψη των κρυφών μοτίβων στα δεδομένα του IoT αναλύοντας τεράστιους όγκους δεδομένων χρησιμοποιώντας εξελιγμένους αλγόριθμους. Αυτό έχει ως αποτέλεσμα, να μπορούμε να συμπληρώσουμε ή να αντικαταστήσουμε τις μη αυτόματες διαδικασίες με αυτοματοποιημένα συστήματα που εκτελούν ενέργειες προερχόμενες από στατιστικά σε κρίσιμες διαδικασίες.

Σκοπός της παρούσας εργασίας είναι η δημιουργία ενός αλγορίθμου ο οποίος με τη χρήση κατάλληλων τεχνικών μηχανικής μάθησης να διαχειρίζεται με ευφυή τρόπο ερωτήματα τα οποία εκτελούνται σε κατανεμημένους κόμβους ενός συστήματος. Ταυτόχρονα προσπαθεί να εισάγει τον αναγνώστη στο κατάλληλο θεωρητικό υπόβαθρο που απαιτείται για τη κατανόηση του αλγορίθμου. Στο πρώτο κεφάλαιο γίνεται μία εισαγωγή στο IoT, τη διαχείριση των δεδομένων τους και των ερωτημάτων επί αυτών. Στο δεύτερο κεφάλαιο γίνεται αναφορά στο τομέα της Τεχνητής Νοημοσύνης (Artificial Intelligence, AI), της Μηχανικής Μάθησης και των αλγορίθμων της. Στο τρίτο κεφάλαιο περιγράφεται το περιβάλλον ανάπτυξης, η βιβλιοθήκη μηχανικής μάθησης που χρησιμοποιήθηκε, οι διαθέσιμοι αλγόριθμοι και οι δυνατότητές τους. Τέλος, στο τέταρτο κεφάλαιο περιγράφονται οι τεχνικές και αλγόριθμοι που χρησιμοποιήθηκαν για την ανάπτυξη του λογισμικού αναλύονται οι τεχνικές προεκτάσεις και παράμετροι του λογισμικού που αναπτύχθηκε καθώς και τα αποτελέσματα από τη χρήση του.

Το **υλικό** που χρησιμοποιήθηκε για την συγγραφή της παρούσας εργασίας βασίστηκε σε διεθνή και ελληνική βιβλιογραφία κυρίως μέσω της μηχανής αναζήτησης Google Scholar και του επιστημονικού δικτύου ResearchGate.

Τα **αποτελέσματα** της εργασίας έδειξαν ότι καθώς βρισκόμαστε ακόμα στα πολύ αρχικά στάδια της σχέσης μεταξύ μεγάλων δεδομένων, IoT και AI, υπάρχουν ακόμη πολλά να προκύψουν από αυτήν την τεχνολογική τριάδα. Επιτελείται συνεχής πρόοδος στο συγκεκριμένο τομέα που θα ωφελήσει τόσο τις επιχειρήσεις όσο και την καθημερινότητά μας.

ΚΕΦΑΛΑΙΟ 1

1.1 Εισαγωγή

Η ανάπτυξη του IoT στις επιχειρήσεις περιλαμβάνει τη χρήση αισθητήρων IoT σε βαριές συσκευές και μηχανήματα. Οι συσκευές IoT μπορούν να δημιουργήσουν έναν τεράστιο όγκο δεδομένων που μπορούν να έχουν χρήσιμες και ουσιαστικές ερμηνείες, αν αξιοποιηθούν και επεξεργαστούν σωστά. Αυτά τα δεδομένα είναι που βοηθούν τη τεχνολογία του IoT να αλλάξει ενεργά τον τρόπο λειτουργίας των οργανισμών.

Ο 21ος αιώνας έχει να κάνει με Big Data και την ανάλυση τους. Υπάρχουν πολλά δεδομένα στα ανθρώπινα χέρια τα οποία είναι ακατανόητα στο μυαλό μας. Οι αναδυόμενες τεχνολογίες του σήμερα, όπως η τεχνητή νοημοσύνη και η μηχανική μάθηση, έχουν καταστήσει δυνατή τη συσσώρευση, την κατανόηση και την αναγνώριση προτύπων στα δεδομένα που συλλέγονται.

Χρησιμοποιώντας τη Μηχανική Μάθηση και τις αναλύσεις Μεγάλων Δεδομένων, ένας υπεύθυνος παραγωγής μπορεί να τρέξει ολόκληρη την εγκατάσταση παραγωγής στα υψηλότερα επίπεδα παραγωγικότητας και αποδοτικότητας. Ωστόσο, στην πορεία, υπάρχει επίσης ένας τόνος δεδομένων στη διάθεσή τους. Η ανάλυση των Μεγάλων Δεδομένων μπορεί να παρέχει τις καλύτερες ερμηνείες για αυτό και να κάνουν το έργο του υπεύθυνου παραγωγής ακόμα πιο εύκολο μέσω της πρόβλεψης προτύπων και της αυτοματοποίησης των λειτουργιών.

1.2 Αποθετήρια δεδομένων

Ένα αποθετήριο δεδομένων που συχνά ονομάζεται αρχείο δεδομένων ή βιβλιοθήκη, είναι μια γενική ορολογία που αναφέρεται σε ένα τμηματοποιημένο σύνολο δεδομένων που χρησιμοποιείται για αναφορά ή ανάλυση. Είναι μια τεράστια υποδομή βάσεων δεδομένων που συγκεντρώνει, διαχειρίζεται και αποθηκεύει διάφορα σύνολα δεδομένων για ανάλυση, διανομή και αναφορά [1].
Σύμφωνα με το Μητρώο Αποθετηρίων Δεδομένων Έρευνας (re3data.org) [2] ένα αποθετήριο δεδομένων είναι ένας

"*Υποτύπος μιας βιώσιμης υποδομής πληροφοριών που παρέχει μακροπρόθεσμη αποθήκευση και πρόσβαση σε ερευνητικά δεδομένα που αποτελεί τη βάση για μια επιστημονική δημοσίευση. Δεδομένα έρευνας είναι αντικείμενα πληροφοριών που δημιουργούνται από επιστημονικά έργα, για παράδειγμα μέσω πειραμάτων, μετρήσεων, ερευνών ή συνεντεύξεων*".

Με άλλα λόγια, ένα αποθετήριο δεδομένων παρέχει μακροπρόθεσμη αποθήκευση στα δεδομένα που υποστηρίζουν διαφόρων ειδών επεξεργασίες (π.χ. υποστήριξη

πειραμάτων για επιστημονικές δημοσιεύσεις, εξαγωγή αναλύσεων, κ.λπ.). Τα αποθετήρια δεδομένων είναι προσπάθειες για την αειφόρο διατήρηση των δεδομένων που δημιουργούνται από τους ερευνητές. Τα αποθετήρια δεδομένων χρησιμεύουν για να διασφαλίσουν ότι τα ερευνητικά δεδομένα είναι προσβάσιμα πέρα από τη διάρκεια μιας επιχορήγησης, ερευνητικού έργου ή μεμονωμένης σταδιοδρομίας.

Τύποι αποθετηρίων δεδομένων είναι οι σχεσιακές Βάσεις Δεδομένων (RDBMS), Αποθήκες Δεδομένων (Data Warehouse), «Λίμνες Δεδομένων» (Data Lake), «Πρατήρια Δεδομένων» (Data Mart) και Λειτουργική Αποθήκευση Δεδομένων (ODS). Ενώ όλα αυτά τα αποθετήρια δεδομένων στον πυρήνα τους μοιράζονται ένα κοινό θέμα - αποθηκεύουν αποτελεσματικά δομημένα και μη δομημένα δεδομένα για υποστήριξη αναφορών και ανάλυσης - διαφέρουν ως προς τον σκοπό τους, τον τύπο των δεδομένων που αποθηκεύουν και τον τρόπο πρόσβασης στα δεδομένα.

1.3 Κατανεμημένα συστήματα

Επίσης, ένα κατανεμημένο σύστημα είναι μια συλλογή ανεξάρτητων στοιχείων που βρίσκονται σε διαφορετικά μηχανήματα και μοιράζονται μηνύματα μεταξύ τους προκειμένου να επιτύχουν κοινούς στόχους. Οι μηχανές που αποτελούν μέρος ενός κατανεμημένου συστήματος μπορεί να είναι υπολογιστές, φυσικοί διακομιστές, εικονικές μηχανές, συσκευές IoT ή οποιοσδήποτε άλλος κόμβος που μπορεί να συνδεθεί στο δίκτυο, να έχει τοπική μνήμη και να επικοινωνεί ανταλλάσσοντας μηνύματα έχοντας ένα καθορισμένο σύνολο ρόλων και εργασιών.

Πλεονεκτήματα των κατανεμημένων συστημάτων είναι η δυνατότητα οριζόντιας επεκτασιμότητας, η αξιοπιστία και η απόδοση. Δεδομένου ότι ο υπολογισμός συμβαίνει ανεξάρτητα σε κάθε κόμβο, είναι εύκολο και γενικά φθηνό να προσθέσουμε επιπλέον κόμβους και λειτουργικότητα, όπως απαιτείται. Τα περισσότερα κατανεμημένα συστήματα είναι ανεκτικά σε σφάλματα καθώς μπορούν να αποτελούνται από εκατοντάδες κόμβους που συνεργάζονται. Το σύστημα γενικά δεν αντιμετωπίζει καμία διακοπή εάν αποτύχει ένα μόνο μηχάνημα. Τα κατανεμημένα συστήματα είναι εξαιρετικά αποδοτικά επειδή ο φόρτος εργασίας μπορεί να διασπαστεί και να σταλεί σε πολλά μηχανήματα.

Προκλήσεις που αντιμετωπίζει η αρχιτεκτονική αυτή σχετίζονται με τον προγραμματισμό των εργασιών, τη καθυστέρηση και τη παρακολούθηση της κατάστασης τους. Ένα κατανεμημένο σύστημα πρέπει να αποφασίζει ποιες εργασίες πρέπει να εκτελούνται, πότε και που πρέπει να εκτελούνται συνυπολογίζοντας περιορισμούς που μπορεί να υπάρχουν και που μπορεί να οδηγήσουν σε υλικό που δεν εκμεταλλεύεται σωστά και απρόβλεπτους χρόνους εκτέλεσης. Όσο πιο ευρέως διανέμεται ένα σύστημά, τόσο μεγαλύτερη καθυστέρηση μπορεί να αντιμετωπίσουμε στις επικοινωνίες. Αυτό συχνά οδηγεί σε αντισταθμίσεις μεταξύ διαθεσιμότητας,

συνέπειας και καθυστέρησης. Η συγκέντρωση, η επεξεργασία, η παρουσίαση και η παρακολούθηση των μετρήσεων που αφορούν τη χρήση του υλικού για πολλούς κόμβους αποτελεί μια σημαντική πρόκληση.

1.4 Διαχείριση δεδομένων

Τις τελευταίες δύο δεκαετίες, η συνεχής αύξηση της υπολογιστικής ισχύς έχει δημιουργήσει μια συντριπτική ροή δεδομένων που απαιτούν αλλαγή στην υπολογιστική αρχιτεκτονική και σε μεγάλης κλίμακας μηχανισμούς επεξεργασίας δεδομένων. Γενικά, οι τρέχουσες αρχιτεκτονικές υπολογιστών παρουσιάζουν μία ολοένα αυξανόμενη ανισορροπία όπου το χάσμα καθυστέρησης μεταξύ των πολυπύρηνων CPU και των μηχανικών σκληρών δίσκων αυξάνονται κάθε χρόνο καθιστώντας δυσκολότερες τις προκλήσεις που σχετίζονται με επεξεργασία δεδομένων υψηλής υπολογιστικής έντασης. Ωστόσο, αυτό το κενό δεν είναι το μοναδικό πρόβλημα να αντιμετωπιστεί. Εφαρμογές που χειρίζονται TeraBytes και PetaBytes κατανεμημένων δεδομένων πρέπει να έχουν πρόσβαση σε δίκτυα με εγγυημένη ποιότητα υπηρεσιών (QoS). Αν οι μηχανισμοί δικτύου παραμελούνται, οι εφαρμογές απλώς θα έχουν πρόσβαση σε μια υπηρεσία καλύτερης προσπάθειας (best effort), η οποία αρκετές φορές δεν συμβαδίζει με τις απαιτήσεις. Ως εκ τούτου, υπάρχει μια κρίσιμη ανάγκη για μια συστηματική και γενική προσέγγιση για την αντιμετώπιση αυτών των προβλημάτων με μια αρχιτεκτονική που να μπορεί να κλιμακωθεί στο μέλλον [3].

Επί του παρόντος, ο όγκος των δεδομένων για εφαρμογές αυξάνεται με απίστευτους ρυθμούς. Στην πραγματικότητα, η πρόοδος στην τεχνολογία αισθητήρων, η αύξηση του διαθέσιμου εύρους ζώνης και η δημοτικότητα των φορητών συσκευών που μπορούν να συνδεθούν στο Διαδίκτυο δημιούργησε ένα περιβάλλον όπου ακόμη και εφαρμογές μικρής κλίμακας πρέπει να αποθηκεύουν μεγάλα σύνολα δεδομένων. Ένα terabyte δεδομένων, που κάποτε ήταν μία ανήκουστη ποσότητα πληροφοριών, είναι πλέον συνηθισμένη. Με τα σύνολα δεδομένων να αυξάνονται πέρα από μερικές εκατοντάδες terabytes, δεν υπάρχουν έτοιμες λύσεις που να μπορούν εύκολα να διαχειριστούν και να αναλύσουν αυτό τον όγκο δεδομένων.

1.5 Διανομή και ανακατανομή δεδομένων

Παρατηρώντας τις αλλαγές που έχουν συμβεί τα τελευταία είκοσι χρόνια στη διαχείριση κατανεμημένων δεδομένων βλέπουμε ότι οι θεμελιώδες αρχές εξακολουθούν να ισχύουν και η διαχείριση κατανεμημένων δεδομένων μπορεί να χαρακτηριστεί από τρεις διαστάσεις: κατανομή, ετερογένεια και αυτονομία των πηγών δεδομένων. Αυτό που άλλαξε πολύ κατά τη διάρκεια των χρόνων και έκανε τα προβλήματα πολύ πιο δύσκολα, είναι η κλίμακα των διαστάσεων: κατανομή πολύ μεγάλης κλίμακας (cluster, P2P, web και Cloud), πολύ υψηλή ετερογένεια (web), και υψηλή αυτονομία (web, P2P).

Επίσης είναι ενδιαφέρον να σημειωθεί είναι ότι οι θεμελιώδεις αρχές του κατακερματισμού ή κατάτμησης, της βάσης δεδομένων όπως η ενοποίηση δεδομένων, η διαχείριση των συναλλαγών, η αναπαραγωγή (replication) και η σχεσιακή επεξεργασία ερωτημάτων έχει αντέξει στη δοκιμασία του χρόνου. Ειδικότερα, νέες τεχνικές και αλγόριθμοι θα μπορούσαν να παρουσιαστούν ως επεκτάσεις προηγούμενων, χρησιμοποιώντας σχεσιακές έννοιες. Σήμερα, για την υποστήριξη των απαιτήσεων σημαντικών εφαρμογών μεγάλου όγκου δεδομένων (π.χ. κοινωνικά δίκτυα, αναλύσεις δεδομένων ιστού), νέες τεχνικές διαχείρισης κατανεμημένων δεδομένων (π.χ. MapReduce, Hadoop, Peanut, Pig latin, SciDB) εμφανίζονται και λαμβάνουν μεγάλη προσοχή από την ερευνητική κοινότητα. Αν και τα πάνε καλά όσον αφορά τη συνέπεια, την ευελιξία και την απόδοση παρουσιάζουν συμβιβασμούς για συγκεκριμένες εφαρμογές, και μπορεί να βλάψουν τη διαλειτουργικότητα των δεδομένων [4].

Προκειμένου να υποστηριχθεί ένας επεκτάσιμος αριθμός εφαρμογών που δημιουργεί και αναφέρεται σε διάφορες δομές διαφορετικών τύπου δεδομένων και δεδομένων που είναι διασκορπισμένα σε διαφορετικές τοποθεσίες, απαιτούνται μηχανισμοί για την επικοινωνία και την κοινή χρήση δεδομένων. Το περιβάλλον αυτό πρέπει να υποστηρίζει την ικανότητα τόσο για εφαρμογές όσο και υπηρεσίες να αποθηκεύουν, ανακαλύπτουν και να ανακτούν δεδομένα με ένα γενικό και ευέλικτο τρόπο. Με τέτοιους μηχανισμούς, πολύπλοκες εφαρμογές μπορούν να υλοποιηθούν από χαλαρά συζευγμένα στοιχεία, υπηρεσίες ή εφαρμογές. Η κοινή χρήση συνόλων δεδομένων μεταξύ σχετικών εφαρμογών μπορεί επίσης να είναι χρήσιμη για την αποτελεσματική εκτέλεση φόρτου εργασίας πολλαπλών ερωτημάτων [5].

1.6 Βελτιστοποίηση ερωτημάτων (Query Optimization)

Μεγάλα δεδομένα σημαίνει μεγάλα ή σύνθετα σύνολα δεδομένων που δεν μπορούν να τα χειριστούν τα υπάρχοντα συστήματα σχεσιακής επεξεργασίας δεδομένων και απαιτείται η εφαρμογή άλλων εργαλείων και τεχνικών. Οι προκλήσεις που αντιμετωπίζει ο χειρισμός τους περιλαμβάνει τη λήψη, αποθήκευση, αναζήτηση, ανάλυση, την κοινή χρήση, μεταφορά, και την οπτικοποίηση των πληροφοριών.
Τα τεράστια δεδομένα που δημιουργούνται σε πραγματικό χρόνο χρειάζονται γρήγορες και αποτελεσματικές μεθόδους για τον χειρισμό διαδραστικών ερωτημάτων σε πραγματικό χρόνο. Αρκετά ερευνητικά και βιομηχανικά έργα επικεντρώνονται σε μεθόδους επεξεργασίας και βελτιστοποίησης ερωτημάτων επί των δεδομένων αυτών. Η επεξεργασία και η βελτιστοποίηση ερωτημάτων Big Data βρίσκεται ακόμη σε αρχικό στάδιο. Είναι τεράστιο το πεδίο των εφαρμογών που χρειάζονται γρήγορη και ακριβή απάντηση για ερωτήματα που εκτελούνται σε Big Data.

1.7 Πρόβλημα

Τα δίκτυα IoT βιομηχανικά και μη χρησιμοποιούνται συνήθως για συστήματα παρακολούθησης και υποστήριξης βρόχων ελέγχου, καθώς και για συστήματα ανίχνευσης κίνησης, έλεγχο διεργασιών και αυτοματισμό εργοστασίων. Για το σκοπό αυτό, τα δεδομένα που παράγονται από την παρακολούθηση συσκευών IoT συλλέγονται, επεξεργάζονται και αποστέλλονται σε ελεγκτές, ενεργοποιητές ή αποθετήρια δεδομένων. Η δρομολόγηση και η επεξεργασία των δεδομένων αυτών είναι αναπόσπαστο μέρος του οποιοδήποτε βιομηχανικού δικτύου μεγάλης κλίμακας καθιστώντας κρίσιμη τη διατήρηση της όποιας καθυστέρησης σε χαμηλά επίπεδα. Συνήθως χρησιμοποιούνται κεντρικά σχήματα, με τα οποία τα δεδομένα μεταφέρονται σε ένα κεντρικό ελεγκτή του δικτύου, από όπου είναι εύκολα προσβάσιμα από οποιονδήποτε άλλο κόμβο τα απαιτεί. Αυτό μπορεί να οδηγήσει σε σημαντικά αυξημένα έξοδα όσο και σε κατανάλωση πόρων του δικτύου με μη βέλτιστο τρόπο [6].

1.8 Κίνητρο

Το κίνητρο για την εκπόνηση της εργασίας αυτής και του λογισμικού που τη συνοδεύει είναι η αυξημένη ζήτηση για ανακάλυψη γνώσης μέσα από Μεγάλα Δεδομένα. Τα Μεγάλα Δεδομένα ενσωματώνουν δεδομένα από πολλές πηγές. Όταν πρόκειται για μη δομημένα δεδομένα, η πολυπλοκότητα τους γίνεται ένα τεράστιο εμπόδιο στη διαδικασία της ανάλυσης τους.

Παρόλο που έχουν προταθεί πολλά εργαλεία και τεχνικές, η εκθετική ανάπτυξη των Μεγάλων Δεδομένων απαιτεί πιο καινοτόμες προσεγγίσεις για την απάντηση σε έξυπνα και σύνθετα ερωτήματα. Ως εκ τούτου, η επεξεργασία ερωτημάτων είναι μια αναπτυσσόμενη και ενδιαφέρουσα ερευνητική περιοχή [7].

ΚΕΦΑΛΑΙΟ 2

2.1 Τεχνητή νοημοσύνη

Τεχνητή νοημοσύνη είναι η ικανότητα ενός υπολογιστή ή ενός ρομπότ που ελέγχεται από υπολογιστή να εκτελεί εργασίες που σχετίζονται συνήθως με ευφυή όντα. Η ορολογία αυτή συνήθως χρησιμοποιείται στην ανάπτυξη συστημάτων προικισμένων με τις πνευματικές διεργασίες που χαρακτηρίζουν τον άνθρωπο, όπως η ικανότητα λογικής, ανακάλυψης του νοήματος, της γενίκευσης ή της εκμάθησης από την εμπειρία του παρελθόντος. Ήδη από τη δεκαετία του 1940 όταν ξεκίνησαν να αναπτύσσονται οι υπολογιστές έχει παρατηρηθεί ότι με τον κατάλληλο προγραμματισμό είναι σε θέση να εκτελούν διεργασίες με μεγάλο βαθμό πολυπλοκότητας σε ένα ευρύ πεδίο από τα

μαθηματικά μέχρι τα παιχνίδια στρατηγικής. Παρόλο τη αυξανόμενη εξέλιξη των χαρακτηριστικών τους όμως, που σχετίζονται με την μνήμη και την επεξεργαστική ισχύ αυτών, απουσιάζει το ανθρώπινο χαρακτηριστικό της ευελιξίας ή των καθημερινών γνώσεων σε διάφορα λογισμικά. Υπάρχουν όμως λογισμικά που έχουν προσεγγίσει σε μεγάλο βαθμό, σε συγκεκριμένες διεργασίες απόδοση που θα συναντούσαμε σε ειδικούς του εκάστοτε κλάδου με αποτέλεσμα η τεχνητή νοημοσύνη να βρει εφαρμογή σε πεδία ετερογενή μεταξύ τους, από την επιστήμη της ιατρικής εως και τις μηχανές αναζήτησης [8].

2.2 Μηχανική μάθηση

Η μηχανική μάθηση είναι ένα εργαλείο που χρησιμοποιείται για την αυτοματοποίηση της λήψης αποφάσεων με βάση τα μοτίβα που βρίσκονται στα διαθέσιμα δεδομένα. Η μηχανική μάθηση είναι ιδιαίτερα χρήσιμη σε τομείς στους οποίους υπάρχει μεγάλος όγκος δεδομένων τόσο επειδή τα διαθέσιμα δεδομένα χρησιμοποιούνται για την εκπαίδευση των μοντέλων, στα οποία περισσότερα δεδομένα ορίζουν πιο ακριβή μοντέλα, και επειδή τα προβλήματα με μεγάλο όγκο δεδομένων είναι δύσκολο να επιλυθούν χωρίς τη βοήθεια αυτοματοποιημένων εργαλείων. Στη μηχανική μάθηση η γνώση αποκτάται από τα διαθέσιμα δεδομένα και γενικεύεται, προκειμένου να ληφθούν αποφάσεις και προβλέψεις για νέα δεδομένα και μελλοντικά γεγονότα.

Η μηχανική μάθηση είναι ένα επιτυχημένο εργαλείο για τους γνωστούς τύπους προβλημάτων όπως η αναζήτηση. Οι μηχανές αναζήτησης παίρνουν αυτοματοποιημένες αποφάσεις σχετικά με τους όρους αναζήτησης προκειμένου να εντοπιστούν οι περισσότερο πιθανές σχετικές πληροφορίες. Τα φίλτρα ανεπιθύμητης αλληλογραφίας αναλύουν αυτόματα e-mail για να τα χαρακτηρίσουν ως ανεπιθύμητα ή μη ανεπιθύμητα. Η αναγνώρισης ομιλίας, που χρησιμοποιείται ευρέως σε τηλέφωνα και τηλεφωνικά κέντρα, βελτιώνεται συνεχώς από την πρόοδο που επιτυγχάνεται στη μηχανική μάθηση. Η μηχανική όραση (Computer Vision) εξελίσσεται με ταχείς ρυθμούς, από τον εντοπισμό ατόμων στα μέσα κοινωνικής δικτύωσης έως τα συστήματα πλοήγησης σε αυτοκινούμενα οχήματα. Η μηχανική μάθηση είναι το θεμέλιο για όλες αυτές τις τεχνολογίες.

Η μηχανική μάθηση συνδέεται στενά με τη στατιστική, όπου τα μοντέλα εντοπίζουν μοτίβα και περιγράφουν τη δομή των δεδομένων. Οι αλγόριθμοι μηχανικής μάθησης προσπαθούν να ελαχιστοποιήσουν το σφάλμα στη περιγραφή των δεδομένα, και ως εκ τούτου αναζητούν μια βέλτιστη τιμή για τις παραμέτρους. Δύο γενικές κατηγορίες μηχανικής μάθησης περιλαμβάνουν την εποπτευόμενη και τη μη εποπτευόμενη μάθηση.

Η εποπτευόμενη μάθηση περιλαμβάνει την εκπαίδευση ενός μοντέλου με ένα γνωστό αποτέλεσμα ή επισημασμένα δεδομένα. Η γραμμική παλινδρόμηση είναι μια βασική

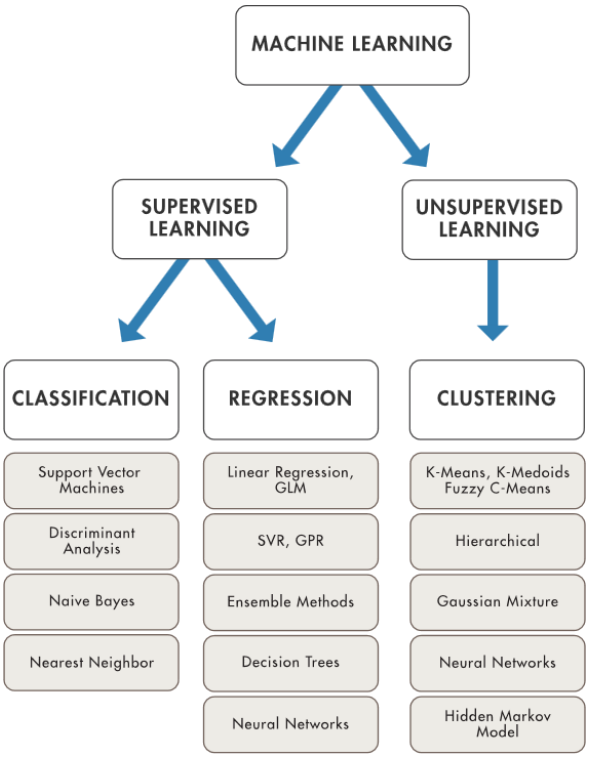
μορφή εποπτευόμενης μάθησης χρησιμοποιώντας συνεχή δεδομένα. Για παράδειγμα, οι τιμές των κατοικιών μπορεί να προβλεφθούν με βάση ιστορικά δεδομένα συμπεριλαμβανομένου του μεγέθους και της θέσης τους. Κάποια γραμμική προσαρμογή στα δεδομένα εκπαίδευσης θα επιτρέψει μια εκτίμηση της τιμής της κατοικίας σε ένα σύνολο δοκιμών που βασίζεται σε συνδυασμό μεγέθους και θέσης που δεν περιέχονται στο δεδομένα που χρησιμοποιηθήκαν για την εκπαίδευση του μοντέλου. Για βέλτιστες προβλέψεις, τα δεδομένα εκπαίδευσης πρέπει περιέχουν μεγάλο αριθμό δεδομένων που εκτείνονται στο εύρος των δυνατών τιμών για το μέγεθος και την τοποθεσία. Κατά τη διάρκεια της εκπαίδευσης ο αλγόριθμος αναζητά μια βέλτιστη τιμή που ελαχιστοποιεί το σφάλμα μεταξύ της πρόβλεψης χρησιμοποιώντας τη γραμμική εξίσωση και τη γνωστή παράμετρο, για παράδειγμα, η τιμή του σπιτιού. Η κατάβαση κλίσης (Gradient descent) είναι ένας κοινός αλγόριθμος για την ελαχιστοποίηση του σφάλματος, στον οποίο ο αλγόριθμος επαναλαμβάνεται συνεχώς κάνοντας βήματα κατά μήκος της κλίσης για να ελαχιστοποιηθεί το σφάλμα. Η λογιστική παλινδρόμηση (Logistic regression) είναι μία γραμμική μορφή της εποπτευόμενης μάθησης που χρησιμοποιείται με διακριτά δεδομένα, και μπορεί να χρησιμοποιηθεί για ταξινόμηση. Η ταξινόμηση μπορεί να είναι δυαδική, όπως ανεπιθύμητη ή μη ανεπιθύμητη ή μπορεί να διαιρέσει τα δεδομένα σε πολλαπλές κλάσεις, όπως αυτόματη ανίχνευση ψηφίων 0 έως 9.

Η χωρίς επίβλεψη μάθηση περιλαμβάνει μάθηση από δεδομένα χωρίς μια προκαθορισμένη ετικέτα. Αυτό μπορεί να περιλαμβάνει τον καθορισμό ομάδων δεδομένων χωρίς προηγούμενη γνώση της κατανομής ή του σχήματος της ομάδας των δεδομένων. Η μάθηση χωρίς επίβλεψη μπορεί να είναι χρήσιμη για την ανακάλυψη άγνωστων συσχετισμών. Τα δεδομένα μπορούν να ομαδοποιηθούν με βάση κάποια παρόμοια χαρακτηριστικά. Η ανακάλυψη μοτίβων σε εικόνες για αναγνώριση είναι μια σημαντικό πεδίο έρευνας [9].

2.2.1 Κατηγορίες μηχανικής μάθησης

Η μηχανική μάθηση ταξινομείται σε γενικές γραμμές ως εποπτευόμενη, χωρίς επίβλεψη, ημι-εποπτευόμενη και ενισχυτική μάθηση. Ένα εποπτευόμενο μοντέλο μάθησης έχει δύο κύριες εργασίες προς εκτέλεση, τη ταξινόμηση (classification) και την παλινδρόμηση (regression). Η ταξινόμηση αφορά την πρόβλεψη μιας ονομαστικής ετικέτας κλάσης, ενώ η παλινδρόμηση αφορά την πρόβλεψη για την αριθμητική τιμή για την ετικέτα κλάσης. Από τη σκοπιά των μαθηματικών, η δημιουργία ενός μοντέλου παλινδρόμησης έχει να κάνει με τον προσδιορισμό της σχέσης μεταξύ της ετικέτας κλάσης και των μεταβλητών εισόδου που χρησιμοποιούνται για τη πρόβλεψη τα οποία ονομάζονται επίσης χαρακτηριστικά. Με όρους στατιστικής τα χαρακτηριστικά αυτά ονομάζονται ανεξάρτητες μεταβλητές, ενώ η ετικέτα κλάσης καλείται εξαρτημένη μεταβλητή. Ένα μοντέλο παλινδρόμησης είναι μια αναπαράσταση αυτής της σχέσης μεταξύ εξαρτημένων και ανεξάρτητων μεταβλητών. Κατά τη διάρκεια της φάσης

εκπαίδευσης, τυχόν νέα δεδομένα συνδέονται στην καμπύλη σχέσεων για να καταλήξουν στη πρόβλεψη. Αυτό μειώνει το πρόβλημα της μηχανικής μάθησης στην επίλυση μιας μαθηματικής εξίσωσης [10]. Η ευρεία ταξινόμηση των τεχνικών μηχανικής μάθησης απεικονίζεται στην Εικόνα 2.1.



Εικόνα 2.1 Τεχνικές μηχανικής μάθησης [11]

2.2.2 Εποπτευόμενη μάθηση

Η εποπτευόμενη μάθηση είναι ένα μοντέλο μάθησης που έχει δημιουργηθεί για να κάνει προβλέψεις, δεδομένης μίας απρόβλεπτης παρουσίας στην είσοδο. Ένας εποπτευόμενος αλγόριθμος μάθησης παίρνει ένα γνωστό σύνολο συνόλου δεδομένων εισόδου και τις γνωστές αποκρίσεις του στα δεδομένα (έξοδος) για να μάθει το μοντέλο παλινδρόμησης / ταξινόμησης. Ένας αλγόριθμος μάθησης στη συνέχεια εκπαιδεύει ένα μοντέλο ώστε να δημιουργεί μια πρόβλεψη ως απόκριση σε νέα δεδομένα ή στο σύνολο των δεδομένων της δοκιμής. Η εποπτευόμενη μάθηση χρησιμοποιεί αλγόριθμους ταξινόμησης και τεχνικές παλινδρόμησης για να αναπτύξει προγνωστικά μοντέλα. Οι αλγόριθμοι περιλαμβάνουν γραμμική παλινδρόμηση, λογιστική παλινδρόμηση και νευρωνικά δίκτυα επίσης, καθώς και δέντρα αποφάσεων. Η εργασία της ταξινόμησης προβλέπει διακριτές αποκρίσεις. Συνιστάται εάν είναι εφικτό τα δεδομένα να μπορούν να κατηγοριοποιηθούν, να επισημανθούν ή να χωριστούν σε συγκεκριμένες ομάδες ή

κλάσεις. Τα μοντέλα ταξινόμησης ταξινομούν τα δεδομένα εισόδου σε κατηγορίες. Δημοφιλές ή σημαντικές εφαρμογές ταξινόμησης περιλαμβάνουν βαθμολόγηση πιστωτικών πιστώσεων, ιατρική απεικόνιση, και αναγνώριση λόγου. Επίσης, η αναγνώριση χειρόγραφου χρησιμοποιεί ταξινόμηση για να αναγνωρίζει γράμματα και αριθμούς, για να ελέγξει εάν ένα μήνυμα ηλεκτρονικού ταχυδρομείου είναι γνήσιο ή ανεπιθύμητο, ή ακόμα και για να ανιχνευθεί εάν ένας όγκος είναι καλοήθης ή καρκινικός. Οι τεχνικές παλινδρόμησης προβλέπουν συνεχείς αποκρίσεις. Μια γραμμική παλινδρόμηση επιχειρεί να μοντελοποιήσει τη σχέση μεταξύ δύο μεταβλητών προσαρμόζοντας γραμμική εξίσωση στα παρατηρούμενα δεδομένα [10].

2.2.3 Μη εποπτευόμενη μάθηση

Στην εποπτευόμενη μάθηση, ο στόχος είναι να μάθουμε την αντιστοίχιση από την είσοδο σε μια έξοδο των οποίων οι σωστές τιμές παρέχονται από κάποιον που εποπτεύει. Αλλά, στη μη εποπτευόμενη μάθηση, ο στόχος είναι να βρεθούν τέτοια χαρακτηριστικά στην εισαγωγή, έτσι ώστε να εντοπιστεί ότι ορισμένα μοτίβα εμφανίζονται συχνότερα από άλλα και να μάθουμε να παρατηρούμε τι συμβαίνει γενικά και τι όχι. Παραδείγματα που σχετίζονται με την αναγνώριση ομιλίας, την ομαδοποίηση εγγράφων και τη συμπίεση εικόνας ταιριάζουν με τη μάθηση χωρίς επίβλεψη.

2.3 Τεχνικές μηχανικής μάθησης

2.3.1 Παλινδρόμηση

Οι μέθοδοι παλινδρόμησης χρησιμοποιούνται για την εκπαίδευση στην εποπτευόμενη μάθηση. Ο στόχος των τεχνικών παλινδρόμησης είναι τυπικά να εξηγήσουν ή να προβλέψουν μια συγκεκριμένη αριθμητική τιμή χρησιμοποιώντας ένα προηγούμενο σύνολο δεδομένων. Για παράδειγμα, οι μέθοδοι παλινδρόμησης μπορούν να λάβουν ιστορικά δεδομένα τιμολόγησης και στη συνέχεια να προβλέψουν την τιμή ενός παρόμοιου ακινήτου. Η γραμμική παλινδρόμηση θεωρείται η απλούστερη και βασικότερη μέθοδος. Σε αυτήν την περίπτωση, ένα σύνολο δεδομένων μοντελοποιείται χρησιμοποιώντας την ακόλουθη εξίσωση: ($y = m * x + b$)

Είναι δυνατή η εκπαίδευση ενός μοντέλου παλινδρόμησης με πολλαπλά ζεύγη δεδομένων, όπως x, y. Για να γίνει αυτό, πρέπει να ορίσουμε μια θέση, καθώς και την κλίση της γραμμής, με μία ελάχιστη απόσταση από όλα τα γνωστά σημεία δεδομένων. Αυτή είναι η γραμμή που προσεγγίζει καλύτερα τις παρατηρήσεις στα δεδομένα και μπορεί να βοηθήσει να κάνουμε προβλέψεις για νέα δεδομένα [11].

2.3.2 Κατηγοριοποίηση (Classification)

Οι αλγόριθμοι κατηγοριοποίησης μπορούν να εξηγήσουν ή να προβλέψουν μια τιμή κλάσης. Η κατηγοριοποίηση είναι ένα βασικό συστατικό για πολλές εφαρμογές τεχνητής νοημοσύνης, αλλά είναι ιδιαίτερα χρήσιμη για εφαρμογές ηλεκτρονικού εμπορίου. Για παράδειγμα, οι αλγόριθμοι κατηγοριοποίησης μπορούν να βοηθήσουν στην πρόβλεψη εάν ένας πελάτης θα αγοράσει ένα προϊόν ή όχι. Οι δύο κατηγορίες σε αυτή την περίπτωση είναι "ναι" και "όχι". Οι αλγόριθμοι κατηγοριοποίησης δεν περιορίζονται σε δύο κατηγορίες και μπορούν να χρησιμοποιηθούν για την κατηγοριοποίηση στοιχείων σε μεγάλο αριθμό κατηγοριών.

2.3.3 Ομαδοποίηση (Clustering)

Οι αλγόριθμοι ομαδοποίησης είναι μέθοδοι μάθησης χωρίς επίβλεψη. Μερικοί συνηθισμένοι αλγόριθμοι ομαδοποίησης είναι οι K-means, η μέση μετατόπιση και η μεγιστοποίηση προσδοκιών. Οι αλγόριθμοι αυτοί ομαδοποιούν σημεία δεδομένων σύμφωνα με παρόμοια ή κοινά χαρακτηριστικά.

Οι τεχνικές ομαδοποίησης είναι ιδιαίτερα χρήσιμες σε επιχειρηματικές εφαρμογές όταν υπάρχει ανάγκη για τμηματοποίηση ή κατηγοριοποίηση μεγάλου όγκου δεδομένων. Τα παραδείγματα περιλαμβάνουν την τμηματοποίηση των πελατών με διαφορετικά χαρακτηριστικά για καλύτερη στόχευση καμπάνιες μάρκετινγκ και τη πρόταση ειδησεογραφικών άρθρων που θα ικανοποιήσουν ορισμένους αναγνώστες. Η ομαδοποίηση είναι επίσης αποτελεσματική στην ανακάλυψη προτύπων σε πολύπλοκα σύνολα δεδομένων που μπορεί να μην είναι προφανή στο ανθρώπινο μάτι [11].

2.4 Αλγόριθμοι μηχανικής μάθησης

2.4.1 Naive Bayes

Θα ήταν δύσκολο και πρακτικά αδύνατο να κατηγοριοποιήσουμε μια ιστοσελίδα, ένα έγγραφο, ένα email ή οποιαδήποτε άλλη μακροσκελή σημείωση κειμένου με μη αυτόματο τρόπο. Εδώ βοηθάει ο αλγόριθμος μηχανικής εκμάθησης Naïve Bayes Classifier. Ένας κατηγοριοποιητής είναι μια συνάρτηση που κατανέμει την τιμή ενός στοιχείου του πληθυσμού από μία από τις διαθέσιμες κατηγορίες. Για παράδειγμα, το φιλτράρισμα ανεπιθύμητων μηνυμάτων και η πρόγνωση του καιρού είναι μερικές από τις δημοφιλείς εφαρμογές του αλγόριθμου Naïve Bayes. Το φίλτρο ανεπιθύμητης αλληλογραφίας είναι ένας ταξινομητής που αποδίδει μια ετικέτα "Ανεπιθύμητο" ή "Μη ανεπιθύμητο" σε όλα τα μηνύματα ηλεκτρονικού ταχυδρομείου.

Ο Naïve Bayes Classifier είναι ένας από τους πιο δημοφιλείς τρόπους μάθησης που ομαδοποιεί με βάση την ομοιότητα, και λειτουργεί χρησιμοποιώντας το δημοφιλές θεώρημα πιθανοτήτων Bayes για την κατασκευή μοντέλων μηχανικής μάθησης και ιδιαίτερα για την πρόβλεψη ασθενειών και την κατηγοριοποίηση εγγράφων. Είναι μια απλή κατηγοριοποίηση λέξεων που βασίζεται στο θεώρημα πιθανοτήτων Bayes για

υποκειμενική ανάλυση περιεχομένου. Αυτός ο αλγόριθμος κατηγοριοποίησης χρησιμοποιεί πιθανότητες χρησιμοποιώντας το θεώρημα Bayes. Η βασική υπόθεση για αυτούς αλγόριθμους είναι ότι όλα τα χαρακτηριστικά θεωρούνται ανεξάρτητα το ένα από το άλλο. Είναι ένας πολύ απλός αλγόριθμος και είναι εύκολο να εφαρμοστεί. Είναι ιδιαίτερα χρήσιμος για μεγάλα σύνολα δεδομένων και μπορεί να εφαρμοστεί για σύνολα δεδομένων τύπου κειμένου.

2.4.2 K Means

Το K-means είναι ένας δημοφιλής αλγόριθμος μη εποπτευόμενης μηχανικής μάθησης για ανάλυση συστάδων. Το K-means είναι μια μη ντετερμινιστική και επαναληπτική μέθοδος. Ο αλγόριθμος λειτουργεί σε ένα δεδομένο σύνολο δεδομένων μέσω ενός προκαθορισμένου αριθμού συστάδων, k. Η έξοδος του αλγορίθμου K-means είναι k συστάδες με δεδομένα εισόδου κατανεμημένα μεταξύ των συστάδων.

Για οποιοδήποτε νέο εισερχόμενο σημείο δεδομένων, το σημείο δεδομένων ταξινομείται ανάλογα με την εγγύτητά του στις κοντινές κλάσεις. Τα σημεία δεδομένων μέσα σε μία συστάδα θα εμφανίζουν παρόμοια χαρακτηριστικά ενώ οι άλλες συστάδες θα έχουν διαφορετικές ιδιότητες. Το βασικό παράδειγμα ομαδοποίησης θα ήταν η ομαδοποίηση του ίδιου είδους πελατών σε μια συγκεκριμένη κατηγορία για κάθε είδους καμπάνια μάρκετινγκ. Είναι επίσης ένας χρήσιμος αλγόριθμος για την ομαδοποίηση εγγράφων.

2.4.3 Support Vector Machines

Ο αλγόριθμος Support Vector Machines είναι ένας εποπτευόμενος αλγόριθμος μηχανικής εκμάθησης για προβλήματα ταξινόμησης ή παλινδρόμησης όπου το σύνολο δεδομένων διδάσκει το SVM (Support Vector Machine) για τις κλάσεις, έτσι ώστε το SVM να μπορεί να ταξινομήσει τυχόν νέα δεδομένα. Λειτουργεί με την ταξινόμηση των δεδομένων σε διαφορετικές κλάσεις με την εύρεση μιας γραμμής που χωρίζει τα δεδομένα εκπαίδευσης σε τάξεις. Καθώς υπάρχουν πολλά τέτοια γραμμές, ο αλγόριθμος SVM προσπαθεί να μεγιστοποιήσει την απόσταση μεταξύ των διαφόρων κλάσεων που εμπλέκονται και αυτό αναφέρεται ως μεγιστοποίηση περιθωρίου. Εάν προσδιοριστεί η γραμμή που μεγιστοποιεί την απόσταση μεταξύ των κλάσεων, αυξάνεται η πιθανότητα να γενικευτεί καλά σε δεδομένα που εμφανίζονται για πρώτη φορά.

2.4.4 Linear Regression

Ο αλγόριθμος γραμμικής παλινδρόμησης δείχνει τη σχέση μεταξύ 2 μεταβλητών και πώς η αλλαγή στη μία μεταβλητή επηρεάζει την άλλη. Ο αλγόριθμος δείχνει την επίδραση στην εξαρτημένη μεταβλητή στην αλλαγή της ανεξάρτητης μεταβλητής. Οι

ανεξάρτητες μεταβλητές αναφέρονται ως επεξηγηματικές μεταβλητές, καθώς εξηγούν τους παράγοντες που επηρεάζουν την εξαρτώμενη μεταβλητή. Η εξαρτημένη μεταβλητή αναφέρεται συχνά ως παράγοντας ενδιαφέροντος ή προγνωστικός παράγοντας. Η γραμμική παλινδρόμηση χρησιμοποιείται για την εκτίμηση πραγματικών συνεχών τιμών. Τα πιο συνηθισμένα παραδείγματα γραμμικής παλινδρόμησης είναι οι προβλέψεις τιμών κατοικίας, οι προβλέψεις πωλήσεων, οι καιρικές προβλέψεις, οι εκτιμήσεις μισθών των εργαζομένων κ.λπ. Ο βασικός στόχος της γραμμικής παλινδρόμησης είναι να ταιριάξει τη καλύτερη γραμμή μεταξύ των προβλέψεων. Η εξίσωση για γραμμική παλινδρόμηση είναι Y = a*x+b όπου y είναι η εξαρτημένη μεταβλητή και x το σύνολο ανεξάρτητων μεταβλητών. a είναι η κλίση και b είναι η τομή.

2.4.5 Decision Tree

Ένα δέντρο αποφάσεων είναι μια γραφική αναπαράσταση που κάνει χρήση της μεθοδολογίας διακλάδωσης ώστε να αποτελέσουν πρότυπα όλα τα πιθανά αποτελέσματα μιας απόφασης, βάσει ορισμένων συνθηκών. Σε ένα δέντρο αποφάσεων, ο εσωτερικός κόμβος αντιπροσωπεύει μια δοκιμή στο χαρακτηριστικό, κάθε κλαδί του δέντρου αντιπροσωπεύει το αποτέλεσμα της δοκιμής και ο κόμβος φύλλου αντιπροσωπεύει μια συγκεκριμένη ετικέτα κλάσης, δηλαδή την απόφαση που λαμβάνεται μετά τον υπολογισμό όλων των χαρακτηριστικών. Οι κανόνες ταξινόμησης αναπαρίστανται μέσω της διαδρομής από τη ρίζα στον κόμβο φύλλου.

Ο αλγόριθμος του δέντρου αποφάσεων ταξινομεί τα αντικείμενα απαντώντας σε «ερωτήσεις» σχετικά με τα χαρακτηριστικά τους που βρίσκονται στα κομβικά σημεία. Ανάλογα με την απάντηση, επιλέγεται ένας από τους κλάδους και στην επόμενη διασταύρωση, τίθεται μια άλλη ερώτηση, έως ότου ο αλγόριθμος φτάσει στο «φύλλο» του δέντρου, το οποίο υποδεικνύει την τελική απάντηση.

Οι εφαρμογές του δέντρου αποφάσεων περιλαμβάνουν πλατφόρμες διαχείρισης γνώσεων για την εξυπηρέτηση πελατών, προβλέψιμη τιμολόγηση και σχεδιασμό προϊόντων. Ένα τυπικό παράδειγμα δέντρων αποφάσεων είναι ο προσδιορισμός του ασφαλίστρου που πρέπει να χρεωθεί με βάση την κατάσταση ενός ατόμου. Το δέντρο αποφάσεων μπορεί να ορίσει έναν περίπλοκο χάρτη κριτηρίων όπως η τοποθεσία, τα είδη των ασφαλισμένων συμβάντων, οι περιβαλλοντικές συνθήκες κ.λπ., και να καθορίσει τις κατηγορίες κινδύνου με βάση τις υποβληθείσες απαιτήσεις και τα δαπανημένα ποσά. Το σύστημα μπορεί στη συνέχεια να αξιολογήσει νέες απαιτήσεις για ασφαλιστική κάλυψη, κατηγοριοποιώντας τις ανά κατηγορία κινδύνου και πιθανές οικονομικές ζημίες.

2.4.6 Artificial Neural Networks

Τα νευρωνικά δίκτυα μιμούνται τη δομή του εγκεφάλου: κάθε τεχνητός νευρώνας συνδέεται με αρκετούς άλλους νευρώνες και μαζί εκατομμύρια νευρώνες δημιουργούν μια περίπλοκη γνωστική δομή. Τα νευρωνικά δίκτυα έχουν μια πολυεπίπεδη δομή: οι νευρώνες σε ένα στρώμα μεταδίδουν δεδομένα σε πολλούς νευρώνες στο επόμενο και ούτω καθεξής. Τελικά, τα δεδομένα φτάνουν στο επίπεδο εξόδου, όπου το δίκτυο αποφασίζει πώς να λύσει ένα πρόβλημα, να ταξινομήσει ένα αντικείμενο κ.λπ. Λόγω της πολυεπίπεδης φύσης των νευρωνικών δικτύων, το πεδίο μελέτης μεγάλων και πολύπλοκων νευρωνικών δικτύων είναι γνωστό ως "deep learning" [12].

Τα νευρωνικά δίκτυα χρησιμοποιούνται για μεγάλη ποικιλία επιχειρηματικών εφαρμογών. Στον τομέα της υγείας χρησιμοποιούνται στην ανάλυση ιατρικών εικόνων, για την επιτάχυνση των διαγνωστικών διαδικασιών και την αναζήτηση φαρμάκων. Στις βιομηχανίες τηλεπικοινωνιών και πολυμέσων, τα νευρωνικά δίκτυα μπορούν να χρησιμοποιηθούν για μηχανική μετάφραση, εντοπισμό απάτης και υπηρεσίες εικονικού βοηθού. Η χρηματοπιστωτική βιομηχανία τα χρησιμοποιεί για τον εντοπισμό απάτης, τη διαχείριση χαρτοφυλακίου και την ανάλυση κινδύνων [11].

2.4.7 K-Nearest Neighbors

Ο KNN είναι ο απλούστερος αλγόριθμος ταξινόμησης. Χρησιμοποιείται επίσης για την πρόβλεψη συνεχών τιμών όπως η παλινδρόμηση. Στο K Nearest Neighbours χρησιμοποιούνται μετρήσεις που βασίζονται στην απόσταση για να έχουμε την πλησιέστερη σωστή πρόβλεψη. Η τελική τιμή πρόβλεψης επιλέγεται βάσει των k γειτόνων. Οι διάφορες μετρικές απόστασης που χρησιμοποιούνται είναι η Ευκλείδια απόσταση, Manhattan, Minkowski και Hamming. Οι τρεις πρώτες είναι συνεχείς συναρτήσεις ενώ η απόσταση Hamming χρησιμοποιείται για κατηγορικές μεταβλητές. Η επιλογή της τιμής του K είναι η πιο σημαντική εργασία σε αυτόν τον αλγόριθμο. Συχνά αναφέρεται ως αλγόριθμος τεμπέλης εκπαιδευόμενου.

2.5 Decision trees (Δέντρα αποφάσεων)

2.5.1 Περιγραφή

Ένα δέντρο απόφασης είναι ένας ταξινομητής που εκφράζεται ως αναδρομική κατάτμηση του χώρου της παρουσίας. Το δέντρο απόφασης αποτελείται από κόμβους που σχηματίζουν ένα δέντρο με ρίζες, που σημαίνει ότι είναι ένα κατευθυνόμενο δέντρο με έναν κόμβο που ονομάζεται "ρίζα" και δεν έχει εισερχόμενα άκρα. Όλοι οι άλλοι κόμβοι έχουν ακριβώς ένα εισερχόμενο άκρο. Ένας κόμβος με εξερχόμενο άκρο ονομάζεται εσωτερικός ή δοκιμαστικός κόμβος. Όλοι οι άλλοι κόμβοι ονομάζονται φύλλα (επίσης γνωστοί ως τερματικοί κόμβοι ή απόφασης). Σε ένα δέντρο αποφάσεων, κάθε εσωτερικός κόμβος χωρίζει το διάστημα παρουσίας σε δύο ή περισσότερα

υποδιαστήματα σύμφωνα με μία διακριτή συνάρτηση των τιμών των χαρακτηριστικών εισόδου. Στην πιο απλή περίπτωση, κάθε δοκιμή εξετάζει ένα μοναδικό χαρακτηριστικό, ώστε ο χώρος παρουσίας να χωρίζεται σύμφωνα με την τιμή του χαρακτηριστικού. Στην περίπτωση αριθμητικών χαρακτηριστικών, η συνθήκη αναφέρεται σε ένα εύρος.

Κάθε φύλλο έχει ανατεθεί σε μία κατηγορία που αντιπροσωπεύει τη καταλληλότερη τιμή. Εναλλακτικά, το φύλλο μπορεί να έχει ένα διάνυσμα πιθανότητας υποδεικνύοντας την πιθανότητα το χαρακτηριστικό να έχει μια συγκεκριμένη τιμή. Οι περιπτώσεις ταξινομούνται κατά τη διαδρομή τους από τη ρίζα του δέντρου προς τα κάτω σε ένα φύλλο, σύμφωνα με το αποτέλεσμα των δοκιμών κατά μήκος της διαδρομής. Συνήθως η πολυπλοκότητα ενός δέντρου μετριέται με ένα από τα παρακάτω χαρακτηριστικά: ο συνολικός αριθμός κόμβων ,ο συνολικός αριθμός φύλλων ,το βάθος του δέντρου και αριθμός των χαρακτηριστικών που χρησιμοποιούνται. Κάθε διαδρομή από τη ρίζα ενός δέντρου απόφασης σε ένα από τα φύλλα του μπορεί να είναι μετατρέπεται σε κανόνα απλά συνδυάζοντας τις δοκιμές κατά μήκος της διαδρομής, και λαμβάνοντας την πρόβλεψη της κλάσης του φύλλου ως τιμή της κλάσης.

Τα δέντρα αποφάσεων είναι αλγόριθμοι που κατασκευάζουν αυτόματα ένα δέντρο απόφασης από ένα δεδομένο σύνολο δεδομένων. Συνήθως ο στόχος είναι να βρεθεί το βέλτιστο δέντρο αποφάσεων ελαχιστοποιώντας το σφάλμα γενίκευσης [13].

2.5.2 Τύποι δέντρων αποφάσεων

Υπάρχουν δύο τύποι δέντρων αποφάσεων. Τα δέντρα κατηγοριοποίησης και τα δέντρα παλινδρόμησης. Τα δέντρα κατηγοριοποίησης θεωρούνται ως το προεπιλεγμένο είδος δέντρων αποφάσεων που χρησιμοποιούνται για τον διαχωρισμό ενός συνόλου δεδομένων σε διαφορετικές κλάσεις, με βάση τη μεταβλητή απόκρισης. Αυτά χρησιμοποιούνται γενικά όταν η μεταβλητή απόκρισης είναι κατηγορικής φύσης, δηλαδή μπορεί να λάβει έναν από έναν περιορισμένο και συνήθως σταθερό αριθμό πιθανών τιμών.

Τα δέντρα παλινδρόμησης χρησιμοποιούνται όταν η μεταβλητή απόκρισης ή μεταβλητή στόχος είναι συνεχής ή αριθμητική. Αυτά χρησιμοποιούνται γενικά σε προγνωστικά είδη προβλημάτων σε σύγκριση με την ταξινόμηση. Τα δέντρα αποφάσεων μπορούν επίσης να ταξινομηθούν σε δύο τύπους, με βάση τον τύπο της μεταβλητής. Δέντρα αποφάσεων συνεχούς μεταβλητής και δέντρα αποφάσεων δυαδικής μεταβλητής. Είναι η μεταβλητή που βοηθά να αποφασιστεί τι είδους δέντρο αποφάσεων θα απαιτηθεί για ένα συγκεκριμένο πρόβλημα [14].

2.5.3 Χρήση αλγορίθμων μηχανικής μάθησης δέντρων αποφάσεων

Αυτοί οι αλγόριθμοι μηχανικής μάθησης βοηθούν στη λήψη αποφάσεων όταν υπάρχει αβεβαιότητα και μας βοηθούν να βελτιώσουμε την επικοινωνία, καθώς παρουσιάζουν μια οπτική αναπαράσταση μιας κατάστασης απόφασης. Επίσης διευκολύνουν να συλλάβουμε την εικόνα του πως η λήψη μιας διαφορετικής απόφαση, θα άλλαζε έντονα μία κατάσταση ή ένα μοντέλο. Οι αλγόριθμοι των δέντρου αποφάσεων βοηθούν στη λήψη βέλτιστων αποφάσεων επιτρέποντας να διασχίσουμε διαδρομές υπολογισμού προς τα εμπρός και προς τα πίσω.

Τα δέντρα αποφάσεων είναι ανθεκτικά σε λάθη και εάν τα δεδομένα εκπαίδευσης περιέχουν σφάλματα οι αλγόριθμοι δέντρων αποφάσεων θα είναι οι πλέον κατάλληλοι για την αντιμετώπιση τέτοιων προβλημάτων. Τα δέντρα απόφασης είναι τα πλέον κατάλληλα για προβλήματα όπου τα διάφορα στιγμιότυπα αντιπροσωπεύονται από ζεύγη τιμών των χαρακτηριστικών. Εάν τα δεδομένα εκπαίδευσης έχουν τιμές που λείπουν, τα δέντρα αποφάσεων μπορούν να χειριστούν τις τιμές αυτές κοιτάζοντας τα δεδομένα σε άλλες στήλες. Τα δέντρα απόφασης ταιριάζουν καλύτερα όταν η συνάρτηση που χρησιμοποιείται έχει διακριτές τιμές εξόδου [14].

2.5.4 Πλεονεκτήματα

Τα δέντρα αποφάσεων είναι πολύ ενστικτώδη και μπορούν να εξηγηθούν σε οποιονδήποτε με ευκολία. Άτομα από μη τεχνικό υπόβαθρο, μπορούν επίσης να αποκρυπτογραφήσουν την υπόθεση που προέρχεται από ένα δέντρο αποφάσεων, καθώς μπορούν να κατανοηθούν με ευκολία θεωρώντας τη αυτονόητη. Όταν χρησιμοποιείτε αλγόριθμους μηχανικής εκμάθησης δέντρων αποφάσεων, ο τύπος δεδομένων δεν αποτελεί περιορισμό καθώς μπορούν να χειριστούν τόσο κατηγορικές όσο και αριθμητικές μεταβλητές. Οι αλγόριθμοι μηχανικής εκμάθησης δένδρων αποφάσεων δεν απαιτούν καμία υπόθεση σχετικά με τη γραμμικότητα των δεδομένων και ως εκ τούτου μπορούν να χρησιμοποιηθούν σε περιπτώσεις όπου οι παράμετροι δεν σχετίζονται γραμμικά. Αυτοί οι αλγόριθμοι μηχανικής μάθησης δεν κάνουν υποθέσεις σχετικά με τη δομή του ταξινομητή και την κατανομή του χώρου.

Αυτοί οι αλγόριθμοι είναι χρήσιμοι στην εξερεύνηση δεδομένων. Τα δέντρα αποφάσεων εκτελούν σιωπηρά την επιλογή χαρακτηριστικών, η οποία είναι πολύ σημαντική στην ανάλυση προγνωστικά. Όταν ένα δέντρο αποφάσεων ταιριάζει σε ένα σύνολο δεδομένων εκπαίδευσης, οι κόμβοι στο επάνω μέρος στο οποίο χωρίζεται το δέντρο απόφασης, θεωρούνται ως σημαντικές μεταβλητές σε ένα δεδομένο σύνολο δεδομένων και η επιλογή χαρακτηριστικών ολοκληρώνεται από προεπιλογή. Τα δέντρα αποφάσεων συμβάλλουν στην εξοικονόμηση χρόνου που αφορά τη προετοιμασία των δεδομένων, καθώς δεν είναι ευαίσθητα στις τιμές που λείπουν και στις υπερβολικές τιμές. Οι τιμές που λείπουν δεν θα εμποδίσουν τον διαχωρισμό των δεδομένων για τη δημιουργία ενός δέντρου αποφάσεων. Οι ακραίες τιμές δεν θα επηρεάσουν επίσης τα

δέντρα αποφάσεων, καθώς ο διαχωρισμός δεδομένων συμβαίνει με βάση ορισμένα δείγματα εντός του εύρους διαχωρισμού και όχι με βάση τις απόλυτες τιμές [14].

2.5.5 Μειονεκτήματα

Όσο μεγαλύτερος είναι ο αριθμός των αποφάσεων σε ένα δέντρο, τόσο μικρότερη είναι η ακρίβεια οποιουδήποτε αναμενόμενου αποτελέσματος. Ένα σημαντικό μειονέκτημα των αλγορίθμων μηχανικής μάθησης δέντρων αποφάσεων είναι ότι τα αποτελέσματα μπορεί να βασίζονται στις προσδοκίες. Όταν οι αποφάσεις λαμβάνονται σε πραγματικό χρόνο, οι αποδόσεις και τα αποτελέσματα που προκύπτουν μπορεί να μην είναι τα ίδια με τα αναμενόμενα ή τα προγραμματισμένα. Υπάρχει πιθανότητα αυτό να οδηγήσει σε μη ρεαλιστικά δέντρα αποφάσεων που οδηγούν σε κακή λήψη αποφάσεων. Οποιεσδήποτε παράλογες προσδοκίες θα μπορούσαν να οδηγήσουν σε μεγάλα λάθη και ελαττώματα στην ανάλυση του δέντρου αποφάσεων, καθώς δεν είναι πάντα δυνατό να προγραμματιστεί για όλα τα ενδεχόμενα που μπορεί να προκύψουν από μια απόφαση. Τα Δέντρα Απόφασης δεν ταιριάζουν καλά για συνεχείς μεταβλητές και οδηγούν σε αστάθεια και προβλήματα στη ταξινόμηση. Τα δέντρα αποφάσεων είναι εύκολα στη χρήση σε σύγκριση με άλλα μοντέλα λήψης αποφάσεων, αλλά η δημιουργία μεγάλων δέντρων αποφάσεων που περιέχουν πολλούς κλάδους είναι μια σύνθετη και χρονοβόρα εργασία. Οι αλγόριθμοι μηχανικής εκμάθησης δέντρων αποφάσεων λαμβάνουν υπόψη μόνο ένα χαρακτηριστικό κάθε φορά και ενδέχεται να μην είναι οι πλέον κατάλληλοι για πραγματικά δεδομένα στο χώρο απόφασης. Τα δέντρα αποφάσεων μεγάλου μεγέθους με πολλαπλούς κλάδους δεν είναι κατανοητά και δημιουργούν αρκετές δυσκολίες στη παρουσίαση τους [14].

2.5.6 Εφαρμογές

Τα δέντρα αποφάσεων συγκαταλέγονται στους δημοφιλείς αλγόριθμους μηχανικής μάθησης και βρίσκουν μεγάλη χρήση στον χρηματοοικονομικό τομέα. Η τηλεπισκόπηση είναι μια περιοχή εφαρμογής για αναγνώριση προτύπων που βασίζεται σε δέντρα αποφάσεων. Οι αλγόριθμοι του δέντρου αποφάσεων χρησιμοποιούνται από τις τράπεζες για να ταξινομήσουν τους αιτούντες δάνειο με βάση την πιθανότητα αθέτησης πληρωμών. Η Gerber Products, μια δημοφιλής εταιρεία βρεφικών προϊόντων, χρησιμοποίησε αλγόριθμο μηχανικής εκμάθησης δέντρων αποφάσεων για να αποφασίσει εάν θα πρέπει να συνεχίσουν να χρησιμοποιούν το πλαστικό PVC (χλωριούχο πολυβινύλιο) στα προϊόντα τους. Το Ιατρικό Κέντρο του Πανεπιστημίου Rush έχει αναπτύξει ένα εργαλείο που ονομάζεται Guardian που χρησιμοποιεί έναν αλγόριθμο μηχανικής εκμάθησης δέντρων αποφάσεων για τον εντοπισμό ασθενών σε κίνδυνο και τις τάσεις ασθενειών [14].

2.6 Regression trees (Δέντρα παλινδρόμησης)

Τα δέντρα παλινδρόμησης είναι πολύ παρόμοια με τα τυπικά δέντρα αποφάσεων. Το μοντέλο που παράγεται από τον αλγόριθμο του δέντρου παλινδρόμησης έχει ακριβώς την ίδια δομή με ένα δέντρο αποφάσεων. Κάθε κόμβος ελέγχει την τιμή ενός χαρακτηριστικού για να προσδιορίσει ποιο κλάδο θα ακολουθήσει το δέντρο. Ωστόσο, ο στόχος των δέντρων παλινδρόμησης είναι να προβλέψουν έναν πραγματικό αριθμό και όχι μια διακριτή ετικέτα κλάσης. Για το σκοπό αυτό, κάθε φύλλο σε ένα δέντρο παλινδρόμησης συνοψίζει τους αριθμούς-στόχους που αντιστοιχούν σε αυτό το τμήμα των δεδομένων. Αυτή είναι συνήθως είτε η μέση είτε η διάμεση τιμή στόχου αυτού του τμήματος.
Όπως τα δέντρα για ταξινόμηση, τα δέντρα παλινδρόμησης είναι μια δημοφιλής τεχνική για την ανακάλυψη γνώσης, επειδή το δέντρο που παράγεται είναι πολύ εύκολο να ερμηνευτεί από τον άνθρωπο. Αν και δεν είναι πολύ ακριβή από μόνα τους, όπως τα δέντρα ταξινόμησης, η ακρίβειά τους μπορεί να βελτιωθεί μαθαίνοντας ένα σύνολο δέντρων και συνδυάζοντάς τα μαζί.
Η διαδικασία εκμάθησης δέντρων παλινδρόμησης είναι πολύ παρόμοια με αυτή της εκμάθησης δέντρων ταξινόμησης. Η κύρια διαφορά έγκειται στον τρόπο μέτρησης της «καθαρότητας» ενός τμήματος. Αντί να προσπαθούν να βρουν τμήματα με την ισχυρή πλειοψηφία μιας διακριτής ετικέτας κλάσης, τα δέντρα παλινδρόμησης αναζητούν να βρουν τμήματα όπου η διακύμανση των τιμών στόχου είναι χαμηλή. Για το σκοπό αυτό, τείνουν να χρησιμοποιούν διαχωριστικές μετρήσεις που βασίζονται στη μείωση της διακύμανσης [15].

2.7 Τεχνική Gradient boosting

Οι αλγόριθμοι Gradient boosting είναι μια οικογένεια ισχυρών τεχνικών μηχανικής μάθησης που έχουν δείξει σημαντική επιτυχία σε ένα ευρύ φάσμα πρακτικών εφαρμογών. Είναι ιδιαίτερα προσαρμόσιμες στις ιδιαίτερες ανάγκες της εκάστοτε εφαρμογής, όπως η εκμάθηση σε σχέση με διαφορετικές λειτουργίες απώλειας.
Σε αυτούς τους αλγόριθμους η διαδικασία εκμάθησης συνδυάζει διαδοχικά νέα μοντέλα για να παρέχει μια ακριβέστερη εκτίμηση της μεταβλητής απόκρισης. Η βασική ιδέα πίσω από αυτόν τον αλγόριθμο είναι να κατασκευάσουν νέους «μαθητευόμενους» που θα συσχετίζονται στο μέγιστο με την αρνητική κλίση της συνάρτησης απώλειας, που σχετίζεται με ολόκληρο το σύνολο. Οι εφαρμοζόμενες συναρτήσεις απώλειας μπορεί να είναι αυθαίρετες, αλλά σαν παράδειγμα, εάν η συνάρτηση σφάλματος είναι η κλασική απώλεια τετραγωνικού σφάλματος, η διαδικασία εκμάθησης θα είχε ως αποτέλεσμα διαδοχική προσαρμογή σφάλματος. Σε γενικές γραμμές, η επιλογή της συνάρτησης απώλειας εναπόκειται στον ερευνητή, με μια πλούσια ποικιλία συναρτήσεων απώλειας που έχουν προκύψει μέχρι τώρα και με τη δυνατότητα χρήσης εξειδικευμένης ανάλογα με την εργασία [16].

Η τεχνική Gradient boosting περιλαμβάνει τρία στοιχεία. Μια συνάρτηση απώλειας που πρέπει να βελτιστοποιηθεί. Έναν αδύναμο «μαθητευόμενου» που να κάνει προβλέψεις. Ένα πρόσθετο μοντέλο για την προσθήκη αδύναμων «μαθητευόμενων» για την ελαχιστοποίηση της συνάρτησης απώλειας.
Η συνάρτηση απώλειας που χρησιμοποιείται εξαρτάται από τον τύπο του προβλήματος που επιλύεται. Πρέπει να είναι διαφοροποιήσιμο, αλλά υποστηρίζονται πολλές τυπικές λειτουργίες απώλειας και μπορείτε να ορίσετε τη δική σας. Για παράδειγμα, η παλινδρόμηση μπορεί να χρησιμοποιεί ένα τετραγωνικό σφάλμα και η ταξινόμηση μπορεί να χρησιμοποιήσει λογαριθμική απώλεια. Ένα πλεονέκτημα της τεχνική Gradient boosting είναι ότι είναι ότι για οποιοδήποτε αλγόριθμο μπορεί να χρησιμοποιήσει οποιαδήποτε διαφοροποιήσιμη συνάρτηση απώλειας.
Τα δέντρα αποφάσεων χρησιμοποιούνται ως αδύναμοι «μαθητευόμενοι» στην τεχνική Gradient boosting. Χρησιμοποιούνται συγκεκριμένα δέντρα παλινδρόμησης που παράγουν πραγματικές τιμές για διασπάσεις και των οποίων η έξοδος μπορεί να προστεθεί μαζί, επιτρέποντας έξοδοι διαδοχικών μοντέλων να προστίθενται και να διορθώσουν ατέλειες στις προβλέψεις. Τα δέντρα κατασκευάζονται με άπληστο τρόπο, επιλέγοντας τα καλύτερα σημεία διάσπασης για να ελαχιστοποιήσουν την απώλεια. Αρχικά, χρησιμοποιήθηκαν πολύ σύντομα δέντρα αποφάσεων που είχαν μόνο μία διάσπαση, που ονομάζεται αποκοπή απόφασης. Μεγαλύτερα δέντρα μπορούν να χρησιμοποιηθούν γενικά με 4 έως 8 επίπεδα. Είναι σύνηθες να περιορίζουμε τους αδύναμους «μαθητευόμενους» με συγκεκριμένους τρόπους, όπως ένα μέγιστο αριθμό στρωμάτων, κόμβων, διαχωρισμών ή κόμβων φύλλων. Αυτό γίνεται για να διασφαλιστεί ότι οι μαθητευόμενοι παραμένουν αδύναμοι, αλλά μπορούν ακόμα να κατασκευαστούν με άπληστο τρόπο.
Τα δέντρα προστίθενται ένα κάθε φορά και τα υπάρχοντα δέντρα στο μοντέλο δεν αλλάζουν. Μια διαδικασία χρησιμοποιείται για την ελαχιστοποίηση της απώλειας κατά την προσθήκη δέντρων. Η διαδικασία αυτή χρησιμοποιείται για την ελαχιστοποίηση ενός συνόλου παραμέτρων, όπως οι συντελεστές σε εξίσωση παλινδρόμησης ή τα βάρη σε νευρωνικό δίκτυο. Μετά τον υπολογισμό του σφάλματος ή της απώλειας, τα βάρη ενημερώνονται για να ελαχιστοποιηθεί αυτό το σφάλμα. Αντί για παραμέτρους, έχουμε υπο-μοντέλα αδύναμους «μαθητευόμενων» ή πιο συγκεκριμένα δέντρα αποφάσεων. Μετά τον υπολογισμό της απώλειας, για να εκτελέσουμε τη διαδικασία, πρέπει να προσθέσουμε ένα δέντρο στο μοντέλο που μειώνει την απώλεια. Αυτό το κάνουμε παραμετροποιώντας το δέντρο, μετά τροποποιούμε τις παραμέτρους του δέντρου και κινούμαστε προς τη σωστή κατεύθυνση μειώνοντας την υπολειπόμενη απώλεια [17].

2.8 Μοντέλο Multiple Additive Regression Trees (MART)

Τα Multiple Additive Regression Trees (MART), ένα σύνολο μοντέλων ενισχυμένων δέντρων παλινδρόμησης, που είναι γνωστό ότι προσφέρουν υψηλή ακρίβεια πρόβλεψης για διάφορες εργασίες και χρησιμοποιείται ευρέως στην πράξη. Ωστόσο, αντιμετωπίζουν ένα ζήτημα που ονομάζουμε υπερ-εξειδίκευση (over-specialization),

όπου τα δέντρα που προστίθενται σε μεταγενέστερες επαναλήψεις τείνουν να επηρεάζουν την πρόβλεψη μόνο μερικών περιπτώσεων και συμβάλλουν αμελητέα στις υπόλοιπες περιπτώσεις. Αυτό επηρεάζει αρνητικά την απόδοση του μοντέλου σε δεδομένα που δεν έχει αντιμετωπίσει και καθιστά επίσης το μοντέλο υπερβολικά ευαίσθητο στις συνεισφορές των λίγων δέντρων που προστίθενται αρχικά [18].

2.9 Αλγόριθμος FastTree

Το FastTree είναι μια αποτελεσματική εφαρμογή του αλγορίθμου MART που χρησιμοποιεί τη τεχνική Gradient Boosting. Η τεχνική Gradient Boosting είναι μια τεχνική μηχανικής μάθησης για προβλήματα παλινδρόμησης. Χτίζει κάθε δέντρο παλινδρόμησης με σταδιακό τρόπο, χρησιμοποιώντας μια προκαθορισμένη συνάρτηση απώλειας για να μετρήσει το σφάλμα για κάθε βήμα και να το διορθώσει στο επόμενο. Αυτό το μοντέλο πρόβλεψης είναι στην πραγματικότητα ένα σύνολο πιο αδύναμων μοντέλων πρόβλεψης. Σε προβλήματα παλινδρόμησης, δημιουργεί μια σειρά τέτοιων δέντρων βήμα βήμα και στη συνέχεια επιλέγει το βέλτιστο δέντρο χρησιμοποιώντας μια αυθαίρετη συνάρτηση απώλειας.
Ο αλγόριθμος MART μαθαίνει ένα σύνολο δέντρων παλινδρόμησης, το οποίο είναι ένα δέντρο αποφάσεων με κλιμακωτές τιμές στα φύλλα του. Ένα δέντρο απόφασης (ή παλινδρόμησης) είναι ένα δυαδικό διάγραμμα ροής που μοιάζει με δέντρο, όπου σε κάθε εσωτερικό κόμβο αποφασίζει ποιος από τους δύο θυγατρικούς κόμβους θα συνεχίσει να βασίζεται σε μία από τις τιμές χαρακτηριστικών από την είσοδο. Σε κάθε κόμβο φύλλου, επιστρέφεται μια τιμή. Στους εσωτερικούς κόμβους, η απόφαση βασίζεται στη δοκιμή x <= v όπου x είναι η τιμή του χαρακτηριστικού στο δείγμα εισόδου και v είναι μία από τις πιθανές τιμές αυτού του χαρακτηριστικού. Οι συναρτήσεις που μπορούν να παραχθούν από ένα δέντρο παλινδρόμησης είναι όλες οι συνεχείς συναρτήσεις ανά κομμάτι.
Το σύνολο των δέντρων παράγεται με υπολογισμό, σε κάθε βήμα, ενός δέντρου παλινδρόμησης που προσεγγίζει την κλίση της συνάρτησης απώλειας και την προσθήκη στο προηγούμενο δέντρο με συντελεστές που ελαχιστοποιούν την απώλεια του νέου δέντρου. Η έξοδος του συνόλου που παράγεται από τον MART σε μια δεδομένη περίπτωση είναι το άθροισμα των εξόδων του δέντρου. Σε περίπτωση δυαδικού προβλήματος κατηγοριοποίησης, η έξοδος μετατρέπεται σε πιθανότητα χρησιμοποιώντας κάποια μορφή βαθμονόμησης. Σε περίπτωση προβλήματος παλινδρόμησης, η έξοδος είναι η τιμή της συνάρτησης που έχει προβλεφθεί. Σε περίπτωση προβλήματος κατάταξης, οι περιπτώσεις ταξινομούνται βάσει της τιμής εξόδου του συνόλου [19].

ΚΕΦΑΛΑΙΟ 3

3.1 Περιβάλλον ανάπτυξης

Το Microsoft Visual Studio είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης (Integrated Development Environment, IDE) από τη Microsoft. Χρησιμοποιείται για την ανάπτυξη προγραμμάτων υπολογιστών, καθώς και ιστότοπων, εφαρμογών ιστού, υπηρεσιών ιστού και εφαρμογών για κινητά. Το Visual Studio χρησιμοποιεί πλατφόρμες ανάπτυξης λογισμικού της Microsoft, όπως Windows API, Windows Forms, Windows Presentation Foundation, Windows Store και Microsoft Silverlight. Μπορεί να παράγει τόσο εγγενή κώδικα (native code) όσο και διαχειριζόμενο κώδικα (managed code) [20].

Το Visual Studio περιλαμβάνει έναν επεξεργαστή κώδικα που υποστηρίζει το IntelliSense (το στοιχείο ολοκλήρωσης κώδικα) καθώς και αναδιαμόρφωση κώδικα. Ο ενσωματωμένος εντοπισμός σφαλμάτων λειτουργεί τόσο ως εντοπισμός σφαλμάτων σε πηγαίο κώδικα όσο και ως εντοπισμός σφαλμάτων σε επίπεδο μηχανής. Άλλα ενσωματωμένα εργαλεία περιλαμβάνουν σχεδιαστές για την κατασκευή εφαρμογών GUI, ιστοσελίδων, για τη δημιουργία κλάσεων και σχήματος βάσης δεδομένων. Δέχεται προσθήκες που επεκτείνουν τη λειτουργικότητα σχεδόν σε κάθε επίπεδο- συμπεριλαμβανομένης της προσθήκης υποστήριξης για συστήματα ελέγχου κώδικα (source control) (όπως το Subversion και το Git) και την προσθήκη νέων συνόλων εργαλείων όπως συντάκτες και οπτικούς σχεδιαστές για συγκεκριμένες γλώσσες ή εργαλεία για άλλες πτυχές του κύκλος ανάπτυξης λογισμικού (όπως ο πελάτης Azure DevOps: Team Explorer). Πέρα από τον τυπικό επεξεργαστή και εντοπισμό σφαλμάτων που παρέχουν τα περισσότερα IDE, το Visual Studio περιλαμβάνει μεταγλωττιστές, εργαλεία συμπλήρωσης κώδικα, γραφικούς σχεδιαστές και πολλές άλλες δυνατότητες για να διευκολύνει τη διαδικασία ανάπτυξης λογισμικού. Η έκδοση που χρησιμοποιήθηκε για την υλοποίηση του προγραμματιστικού μέρους της εργασίας είναι η Community edition 2019. Η συγκεκριμένη έκδοση κυκλοφόρησε πρώτη φορά το Νοέμβριο του 2014, ως δωρεάν έκδοση, με παρόμοια λειτουργικότητα με το Visual Studio Professional [21].

3.2 .NET Framework

3.2.1 Περιγραφή

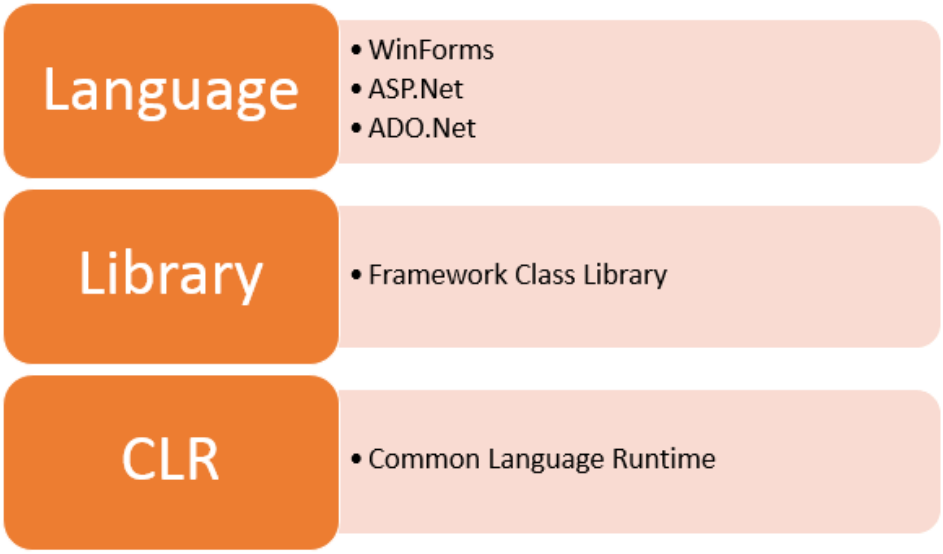
Το .Net Framework είναι μια πλατφόρμα ανάπτυξης λογισμικού που αναπτύχθηκε από τη Microsoft για τη δημιουργία και εκτέλεση εφαρμογών Windows. Το πλαίσιο .Net αποτελείται από εργαλεία για προγραμματιστές, γλώσσες προγραμματισμού και βιβλιοθήκες για τη δημιουργία desktop και web εφαρμογών καθώς και για την ανάπτυξη web services. Χρησιμοποιείται επίσης για τη δημιουργία ιστοσελίδων, υπηρεσιών διαδικτύου και παιχνιδιών. Το .Net Framework προοριζόταν να

δημιουργήσει εφαρμογές, οι οποίες θα εκτελούνται στην πλατφόρμα των Windows. Η πρώτη έκδοση του πλαισίου .Net κυκλοφόρησε το έτος 2002. Η έκδοση ονομάστηκε .Net framework 1.0. Το πλαίσιο Microsoft .Net έχει προχωρήσει πολύ από τότε και η τρέχουσα έκδοση είναι .Net Framework 4.7.2. Το πλαίσιο υποστηρίζει επίσης διάφορες γλώσσες προγραμματισμού όπως η Visual Basic και η C#. Έτσι, οι προγραμματιστές μπορούν να διαλέξουν και να επιλέξουν τη γλώσσα για την ανάπτυξη της απαιτούμενης εφαρμογής [22].

3.2.2 Αρχιτεκτονική

Η αρχιτεκτονική του .Net Framework είναι ένα μοντέλο προγραμματισμού για την πλατφόρμα .Net που παρέχει ένα περιβάλλον εκτέλεσης ενσωματώνοντας διάφορες γλώσσες προγραμματισμού για ανάπτυξη διαφόρων εφαρμογών. Αποτελείται από βιβλιοθήκες κλάσεων και επαναχρησιμοποιήσιμα στοιχεία.

Η βασική αρχιτεκτονική του .Net Framework είναι όπως φαίνεται παρακάτω.



Εικόνα 3.1 Βασική αρχιτεκτονική του.Net Framework

Η αρχιτεκτονική του .Net framework βασίζεται στα ακόλουθα βασικά στοιχεία:

1. Κοινή γλώσσα εκτέλεσης (Common Language Infrastructure)

Το "Common Language Infrastructure" ή CLI είναι μια πλατφόρμα στην αρχιτεκτονική .Net στην οποία εκτελούνται τα προγράμματα. Το CLI έχει τα ακόλουθα βασικά χαρακτηριστικά:

- Χειρισμός εξαίρεσης

Εξαιρέσεις είναι σφάλματα που εμφανίζονται κατά την εκτέλεση της εφαρμογής. Εάν μια εφαρμογή προσπαθήσει να ανοίξει ένα αρχείο στον τοπικό υπολογιστή, αλλά το αρχείο δεν υπάρχει ή εάν η εφαρμογή προσπαθήσει να πάρει κάποιες εγγραφές από μια βάση δεδομένων, αλλά η σύνδεση με τη βάση δεδομένων δεν είναι έγκυρη.

- Συλλογή απορριμμάτων

Η συλλογή σκουπιδιών είναι η διαδικασία απομάκρυνσης ανεπιθύμητων πόρων όταν δεν απαιτούνται πλέον. Ένας πόρος αρχείου που δεν απαιτείται πλέον. Εάν η εφαρμογή έχει ολοκληρώσει όλες τις λειτουργίες σε ένα αρχείο, τότε η διεργασία που εκτελείται για τον χειρισμό του αρχείου ενδέχεται να μην απαιτείται πλέον.Εάν η εφαρμογή έχει ολοκληρώσει όλες τις λειτουργίες σε μια βάση δεδομένων, τότε η σύνδεση της βάσης δεδομένων μπορεί να μην απαιτείται πλέον.

- Εργασία με διάφορες γλώσσες προγραμματισμού

Ένας προγραμματιστής μπορεί να αναπτύξει μια εφαρμογή σε μια ποικιλία γλωσσών προγραμματισμού .Net. Το πρώτο επίπεδο είναι η ίδια η γλώσσα προγραμματισμού, τα πιο κοινά είναι τα VB.Net και C#. Υπάρχει ένας μεταγλωττιστής που θα είναι ξεχωριστός για κάθε γλώσσα προγραμματισμού. Έτσι, κάτω από τη γλώσσα VB.Net, θα υπάρχει ένας ξεχωριστός μεταγλωττιστής VB.Net. Ομοίως, για το C#, θα έχετε έναν άλλο μεταγλωττιστή. Και το τελευταίο επίπεδο στο .Net είναι το Common Language Interpreter - που θα χρησιμοποιηθεί για την εκτέλεση ενός .net προγράμματος που έχει αναπτυχθεί σε οποιαδήποτε γλώσσα προγραμματισμού. Έτσι, ο επόμενος μεταγλωττιστής θα στείλει το πρόγραμμα στο επίπεδο CLI για να τρέξει την εφαρμογή .Net.

2. Βιβλιοθήκη Κλάσεων (Class Library)

Το .NET Framework περιλαμβάνει ένα σύνολο συγκεκριμένων βιβλιοθηκών κλάσεων. Μια βιβλιοθήκη κλάσης είναι μια συλλογή μεθόδων και λειτουργιών που μπορούν να χρησιμοποιηθούν για ένα σκοπό. Για παράδειγμα, υπάρχει μια βιβλιοθήκη κλάσης με μεθόδους για τον χειρισμό όλων των λειτουργιών σε επίπεδο αρχείου. Υπάρχει λοιπόν μια μέθοδος που μπορεί να χρησιμοποιηθεί για την ανάγνωση του κειμένου από ένα αρχείο. Ομοίως, υπάρχει μια μέθοδος για να γράψουμε κείμενο σε ένα αρχείο.

3. Γλώσσες

Οι τύποι εφαρμογών που μπορούν να ενσωματωθούν στο.Net Framework ταξινομούνται ευρέως στις ακόλουθες κατηγορίες.

- WinForms

Χρησιμοποιείται για την ανάπτυξη εφαρμογών που βασίζονται σε φόρμες, οι οποίες θα εκτελούνται σε μηχάνημα τελικού χρήστη (client-based application).

- ASP.Net

Χρησιμοποιείται για την ανάπτυξη εφαρμογών που βασίζονται στον ιστό, οι οποίες έχουν σχεδιαστεί για να εκτελούνται σε οποιοδήποτε πρόγραμμα περιήγησης, όπως Internet Explorer, Chrome ή Firefox. Η εφαρμογή Ιστού θα υποβληθεί σε επεξεργασία σε διακομιστή, στον οποίο θα πρέπει να έχουν εγκατασταθεί Υπηρεσίες Πληροφοριών Διαδικτύου (Internet Information Services) ένα στοιχείο της Microsoft που χρησιμοποιείται για την εκτέλεση μιας εφαρμογής Asp.Net. Το αποτέλεσμα της εκτέλεσης αποστέλλεται στη συνέχεια στους υπολογιστές -πελάτες και η έξοδος εμφανίζεται στο πρόγραμμα περιήγησης.

- ADO.Net

Αυτή η τεχνολογία χρησιμοποιείται για την ανάπτυξη εφαρμογών για αλληλεπίδραση με βάσεις δεδομένων, όπως Oracle ή Microsoft SQL Server.

3.2.3 Χαρακτηριστικά

1) Διαλειτουργικότητα

Το πλαίσιο .Net παρέχει μεγάλη υποστήριξη με προηγούμενες εκδόσεις. Αν είχαμε μια εφαρμογή βασισμένη σε παλαιότερη έκδοση του πλαισίου .Net, π.χ. 2.0. προσπαθήσουμε να εκτελέσουμε την ίδια εφαρμογή σε ένα μηχάνημα που είχε την υψηλότερη έκδοση του πλαισίου .Net, π.χ 3.5 η εφαρμογή θα εξακολουθούσε να λειτουργεί. Αυτό συμβαίνει επειδή με κάθε έκδοση, η Microsoft διασφαλίζει ότι οι παλαιότερες εκδόσεις είναι διαλειτουργικές με την τελευταία έκδοση.

2) Φορητότητα

Οι εφαρμογές που βασίζονται στο πλαίσιο .Net μπορούν να λειτουργήσουν σε οποιαδήποτε πλατφόρμα Windows. Και τώρα τον τελευταίο καιρό, η Microsoft σκέφτεται επίσης να κάνει τα προϊόντα της Microsoft να λειτουργούν σε άλλες πλατφόρμες, όπως το iOS και το Linux.

3) Ασφάλεια

Το .NET Framework διαθέτει έναν καλό μηχανισμό ασφαλείας. Ο ενσωματωμένος μηχανισμός ασφαλείας βοηθά τόσο στην επικύρωση όσο και στην επαλήθευση εφαρμογών. Κάθε εφαρμογή μπορεί να καθορίσει ρητά τον μηχανισμό ασφαλείας τους.

Κάθε μηχανισμός ασφαλείας χρησιμοποιείται για να παραχωρήσει στον χρήστη πρόσβαση στον κώδικα ή στο τρέχον πρόγραμμα.

4) Διαχείριση μνήμης

Το CLR κάνει όλη τη δουλειά όσον αφορά τη διαχείριση μνήμης. Το.Net Framework έχει όλη τη δυνατότητα να δει αυτούς τους πόρους, οι οποίοι δεν χρησιμοποιούνται από ένα τρέχον πρόγραμμα. Στη συνέχεια θα αποδεσμεύσει τους πόρους αυτούς αναλόγως. Αυτό γίνεται μέσω ενός προγράμματος που ονομάζεται "Garbage Collector", το οποίο εκτελείται ως μέρος του .Net Framework. Ο συλλέκτης απορριμμάτων λειτουργεί σε τακτά χρονικά διαστήματα και συνεχίζει να ελέγχει ποιοι πόροι του συστήματος δεν χρησιμοποιούνται και τους απελευθερώνει ανάλογα.

5) Απλοποιημένη ανάπτυξη

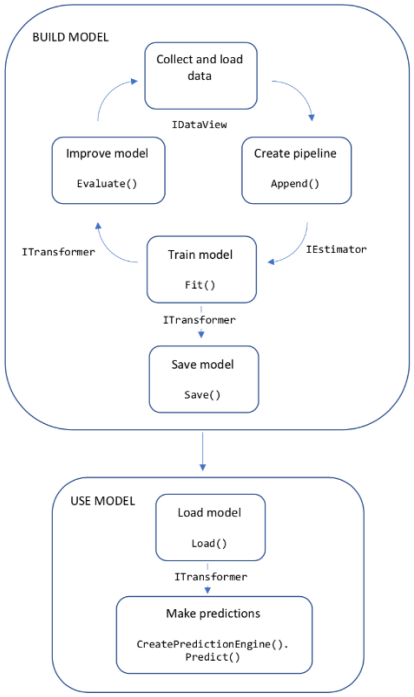
Το πλαίσιο .Net διαθέτει επίσης εργαλεία, τα οποία μπορούν να χρησιμοποιηθούν για τη δημιουργία πακέτων εφαρμογών που έχουν δημιουργηθεί στο.Net Framework. Αυτά τα πακέτα μπορούν στη συνέχεια να διανεμηθούν σε μηχανές -πελάτες. Στη συνέχεια, τα πακέτα εγκαθιστούν αυτόματα την εφαρμογή.

3.3 ML.NET

Το ML.NET δίνει τη δυνατότητα να προσθέσουμε μηχανική εκμάθηση σε εφαρμογές .NET. Με αυτήν τη δυνατότητα, μπορούμε να κάνουμε αυτόματες προβλέψεις χρησιμοποιώντας τα δεδομένα που είναι διαθέσιμα στην εφαρμογή μας. Οι εφαρμογές μηχανικής μάθησης χρησιμοποιούν μοτίβα στα δεδομένα για να κάνουν προβλέψεις και δεν προγραμματίζονται ρητά (αυστηρά). Κεντρικό σημείο στο ML.NET είναι ένα μοντέλο μηχανικής μάθησης. Το μοντέλο καθορίζει τα βήματα που απαιτούνται για τη μετατροπή των δεδομένων εισόδου σε πρόβλεψη. Με το ML.NET, μπορούμε να εκπαιδεύσουμε ένα προσαρμοσμένο μοντέλο καθορίζοντας έναν αλγόριθμο ή μπορούμε να εισαγάγετε προ-εκπαιδευμένα μοντέλα από τη πλατφόρμα TensorFlow και το πρότυπο ONNX. Μόλις δημιουργήσουμε ένα μοντέλο, μπορούμε να το προσθέσουμε στην εφαρμογή μας για να κάνουμε τις προβλέψεις. Το ML.NET εκτελείται σε λειτουργικά συστήματα Linux και macOS χρησιμοποιώντας το Framework .NET Core ή σε λειτουργικό σύστημα Windows χρησιμοποιώντας το .NET Framework. Έκδοση των 64 bit υποστηρίζεται σε όλες τις πλατφόρμες. Η έκδοση των 32 bit υποστηρίζεται στα Windows, εκτός από τις λειτουργίες που σχετίζονται με τη πλατφόρμα TensorFlow, το framework LightGBM και το πρότυπο ONNX [23].

Το ML.NET είναι η λύση που ανέπτυξε η Microsoft για το πρόβλημα της υποστήριξης των προγραμματιστών με ένα πλαίσιο μηχανικής μάθησης για συγγραφή, δοκιμή και ανάπτυξη ενός τρόπου κωδικοποίησης και αυτοματοποίησης της ροής εργασίας που απαιτείται για την παραγωγή ενός μοντέλου μηχανικής μάθησης. Η υλοποίηση του έγινε με στόχο την παροχή ενός εργαλείου που είναι εύκολο στη χρήση, επεκτάσιμο σε μεγάλα σύνολα δεδομένων, παρέχοντας ταυτόχρονα καλή απόδοση και σε θέση να ενοποιήσει σε έναν ενιαίο API για μετασχηματισμούς δεδομένων , υπερσύγχρονα μοντέλα μηχανικής εκμάθησης. Στην τρέχουσα εφαρμογή του, Το ML.NET περιλαμβάνει 2773K γραμμές κώδικα C# και περίπου 74K γραμμές κώδικα C ++, ο δεύτερος χρησιμοποιείται κυρίως για υψηλή απόδοση λειτουργίες γραμμικής άλγεβρας που χρησιμοποιούν οδηγίες SIMD (Single Instruction, Multiple Data Units). Το ML.NET υποστηρίζει περισσότερα από 40 μοντέλα μηχανικής εκμάθησης [24].

Το παρακάτω διάγραμμα αντιπροσωπεύει τη δομή του κώδικα εφαρμογής, καθώς και την επαναληπτική διαδικασία ανάπτυξης μοντέλου:



Εικόνα 3.2 Ροή εργασιών κώδικα στο ML.NET

3.3.1 Ροή εργασιών κώδικα

- Συλλέγουμε και φορτώνουμε δεδομένα για την εκπαίδευση του μοντέλου σε ένα αντικείμενο IDataView

- Καθορίζουμε έναν αγωγό λειτουργιών για εξαγωγή χαρακτηριστικών και εφαρμογή αλγορίθμου μηχανικής μάθησης
- Εκπαιδεύστε το μοντέλο καλώντας τη συνάρτηση Fit () στον αγωγό
- Αξιολογούμε το μοντέλο και επαναλαμβάνουμε για να το βελτίωσουμε
- Αποθηκεύουμε το μοντέλο σε δυαδική μορφή, για να το χρησιμοποιήσουμε σε μια εφαρμογή
- Φορτώνουμε ξανά το μοντέλο σε ένα αντικείμενο τύπου ITransformer
- Πραγματοποιήστε προβλέψεις καλώντας τη μέθοδο CreatePredictionEngine.Predict ()

3.3.2 Μοντέλο μηχανικής μάθησης στο ML.NET

- Γενικά

Το πιο βασικό μοντέλο είναι η δισδιάστατη γραμμική παλινδρόμηση, όπου η συνεχής ποσότητα είναι ανάλογη με την άλλη, όπως στο παραπάνω παράδειγμα της τιμής κατοικίας. Το μοντέλο είναι απλά: Price = b + Size*w. Οι παράμετροι b και w υπολογίζονται τοποθετώντας μια γραμμή σε ένα σύνολο ζευγών (size, price). Τα δεδομένα που χρησιμοποιούνται για την εύρεση των παραμέτρων του μοντέλου ονομάζονται δεδομένα εκπαίδευσης (training data). Οι εισόδους ενός μοντέλου μηχανικής μάθησης ονομάζονται χαρακτηριστικά (features). Σε αυτό το παράδειγμα το Size είναι το μόνο χαρακτηριστικό. Οι τιμές που παρέχονται από άμεση παρατήρηση (ground-truth values) που χρησιμοποιούνται για την εκπαίδευση ενός μοντέλου μηχανικής μάθησης ονομάζονται ετικέτες (labels). Στο συγκεκριμένο παράδειγμα οι τιμές Price στα δεδομένα εκπαίδευσης είναι οι ετικέτες.

- Προετοιμασία δεδομένων

Στις περισσότερες περιπτώσεις, τα δεδομένα που έχουμε στη διάθεσή μας δεν είναι κατάλληλα για απευθείας χρήση στην εκπαίδευση ενός μοντέλου μηχανικής μάθησης. Τα μη επεξεργασμένα δεδομένα πρέπει να ετοιμαστούν ή να υποστούν προεπεξεργασία, προκειμένου να χρησιμοποιηθούν για την εύρεση των παραμέτρων του μοντέλου μας. Τα δεδομένα σας μπορεί να χρειαστεί να μετατραπούν από τιμές συμβολοσειράς σε αριθμητική αναπαράσταση. Ενδέχεται να έχουμε περιττές πληροφορίες στα δεδομένα εισαγωγής μας. Ίσως χρειαστεί να μειώσουμε ή να επεκτείνουμε τις διαστάσεις των δεδομένων εισαγωγής. Τα δεδομένα σας μπορεί να χρειαστεί να ομαλοποιηθούν ή να κλιμακωθούν.

- Αξιολόγηση μοντέλου

Μόλις εκπαιδεύσουμε το μοντέλο μας μπορούμε να μάθουμε πόσο καλά θα κάνει μελλοντικές προβλέψεις αξιολογώντας το μοντέλο μας σε σχέση με ορισμένα νέα δεδομένα δοκιμών. Κάθε τύπος εργασίας μηχανικής μάθησης έχει μετρήσεις που χρησιμοποιούνται για την αξιολόγηση της ακρίβειας του μοντέλου έναντι του συνόλου δεδομένων δοκιμής. Οι μετρήσεις αξιολόγησης μάς λένε ότι το σφάλμα είναι χαμηλό και ότι η συσχέτιση μεταξύ της προβλεπόμενης εξόδου και της εξόδου της δοκιμής είναι υψηλή. Σε πραγματικά παραδείγματα, απαιτείται περισσότερος ρυθμίσεις και βελτιώσεις για την επίτευξη καλών μετρήσεων μοντέλου.

3.3.3 Αρχιτεκτονική ML.NET

- Γενικά

Μια εφαρμογή ML.NET ξεκινά με ένα αντικείμενο MLContext. Αυτό το μοναδικό αντικείμενο περιέχει καταλόγους (catalogs). Ένας κατάλογος είναι ένα εργοστάσιο (factory) για φόρτωση και αποθήκευση δεδομένων, μετασχηματισμούς, εκπαιδευτές και μοντέλα λειτουργικών στοιχείων. Κάθε αντικείμενο καταλόγου έχει μεθόδους για τη δημιουργία των διαφορετικών τύπων στοιχείων.

- Δημιουργία του αγωγού (pipeline)

Μέσα σε κάθε κατάλογο υπάρχει ένα σύνολο επεκτάσιμων μεθόδων. Οι μέθοδοι αυτοί χρησιμοποιούνται για τη δημιουργία ενός αγωγού εκπαίδευσης. Σε αυτό το σημείο, τα αντικείμενα δημιουργούνται μόνο. Δεν πραγματοποιείται καμία εκτέλεση.

- Εκπαίδευση του μοντέλου

Μόλις δημιουργηθούν τα αντικείμενα του αγωγού, τα δεδομένα μπορούν να χρησιμοποιηθούν για την εκπαίδευση του μοντέλου. Η μέθοδος Fit () χρησιμοποιεί τα δεδομένα εκπαίδευσης εισόδου για να εκτιμήσει τις παραμέτρους του μοντέλου. Αυτό είναι γνωστό ως εκπαίδευση του μοντέλου. Μετά την κλήση Fit (), οι τιμές των παραμέτρων είναι γνωστές. Τα περισσότερα μοντέλα θα έχουν πολλές παραμέτρους. Το αντικείμενο του μοντέλου που προκύπτει υλοποιεί τη διεπαφή ITransformer. Δηλαδή, το μοντέλο μετατρέπει δεδομένα εισόδου σε προβλέψεις.

- Χρήση του μοντέλου

Μπορούμε να μετατρέψουμε τα δεδομένα εισαγωγής σε προβλέψεις μαζικά ή μία εισαγωγή κάθε φορά. Η μέθοδος CreatePredictionEngine () παίρνει μια κλάση εισόδου και μια κλάση εξόδου. Τα ονόματα πεδίων ή / και τα χαρακτηριστικά κώδικα καθορίζουν τα ονόματα των στηλών δεδομένων που χρησιμοποιήθηκαν κατά τη διάρκεια της εκπαίδευσης του μοντέλου και της πρόβλεψης.

- Μοντέλα δεδομένων και σχήμα

Στον πυρήνα ενός αγωγού μηχανικής εκμάθησης στο ML.NET βρίσκονται τα αντικείμενα DataView. Κάθε μετασχηματισμός στον αγωγό έχει ένα σχήμα εισόδου (ονόματα δεδομένων, τύπους και μεγέθη που ο μετασχηματισμός αναμένει να δει στην είσοδό του) και ένα σχήμα εξόδου (ονόματα δεδομένων, τύποι και μεγέθη που παράγει ο μετασχηματισμός μετά τον μετασχηματισμό). Εάν το σχήμα εξόδου από έναν μετασχηματισμό στον αγωγό δεν ταιριάζει με το σχήμα εισόδου του επόμενου μετασχηματισμού, το ML.NET παράγει μια εξαίρεση. Ένα αντικείμενο προβολής δεδομένων (DataView) έχει στήλες και σειρές. Κάθε στήλη έχει όνομα και τύπο και μήκος. Όλοι οι αλγόριθμοι στο ML.NET αναζητούν μια στήλη εισόδου που είναι ένα διάνυσμα. Από προεπιλογή, αυτή η διανυσματική στήλη ονομάζεται Features. Όλοι οι αλγόριθμοι δημιουργούν επίσης νέες στήλες αφού εκτελέσουν μια πρόβλεψη. Τα σταθερά ονόματα αυτών των νέων στηλών εξαρτώνται από τον τύπο του αλγορίθμου μηχανικής μάθησης. Μια σημαντική ιδιότητα των αντικειμένων DataView είναι ότι αξιολογούνται με αργό ρυθμό. Οι προβολές δεδομένων φορτώνονται και λειτουργούν μόνο κατά τη διάρκεια της εκπαίδευσης και της αξιολόγησης του μοντέλου και της πρόβλεψης δεδομένων.

- Ανάπτυξη μοντέλου

Σε πραγματικές εφαρμογές, ο κωδικός εκπαίδευσης και αξιολόγησης μοντέλου θα είναι ξεχωριστός από την πρόβλεψή μας. Στην πραγματικότητα, αυτές οι δύο δραστηριότητες εκτελούνται συχνά από ξεχωριστές ομάδες. Η ομάδα ανάπτυξης μοντέλων μπορεί να αποθηκεύσει το μοντέλο για χρήση στην εφαρμογή πρόβλεψης.

3.3.4 Εργασίες στο ML.Net

- Γενικά

Μια εργασία μηχανικής μάθησης είναι ο τύπος πρόβλεψης ή συμπερασμάτων που γίνεται, με βάση το πρόβλημα ή την ερώτηση που υποβάλλεται, και τα διαθέσιμα

δεδομένα. Για παράδειγμα, η εργασία ταξινόμησης αναθέτει δεδομένα σε κατηγορίες και η ομαδοποίηση εργασιών ομαδοποιεί δεδομένα ανάλογα με την ομοιότητα. Οι εργασίες μηχανικής εκμάθησης βασίζονται σε μοτίβα στα δεδομένα αντί να προγραμματίζονται ρητά. Υπάρχουν διάφορες εργασίες μηχανικής εκμάθησης από τις οποίες μπορούμε να επιλέξουμε στο ML.NET και ορισμένες συνήθεις περιπτώσεις χρήσης. Μόλις αποφασίσουμε ποια εργασία είναι κατάλληλη για το σενάριό μας, πρέπει να επιλέξουμε τον καλύτερο αλγόριθμο για να εκπαιδεύσουμε το μοντέλο μας [26].

- Binary classification

Μια εποπτευόμενη εργασία μηχανικής μάθησης που χρησιμοποιείται για να προβλέψει σε ποια από τις δύο κλάσεις (κατηγορίες) ανήκει ένα παράδειγμα δεδομένων. Η εισαγωγή ενός αλγορίθμου ταξινόμησης είναι ένα σύνολο παραδειγμάτων με ετικέτα, όπου κάθε ετικέτα είναι ένας ακέραιος αριθμός είτε 0 είτε 1. Η έξοδος ενός δυαδικού αλγορίθμου ταξινόμησης είναι ένας ταξινομητής, τον οποίο μπορούμε να χρησιμοποιήσουμε για να προβλέψουμε την κλάση των νέων δεδομένων χωρίς ετικέτα. Για καλύτερα αποτελέσματα με δυαδική ταξινόμηση, τα δεδομένα εκπαίδευσης πρέπει να είναι ισορροπημένα (δηλαδή ίσοι αριθμοί θετικών και αρνητικών δεδομένων προπόνησης). Οι τιμές που λείπουν πρέπει να διαχειρίζονται πριν από την εκπαίδευση. Τα δεδομένα εισόδου της στήλης της ετικέτας πρέπει να είναι τύπου Boolean. Τα δεδομένα εισόδου στηλών που αφορούν χαρακτηριστικά (features) πρέπει να είναι διάνυσμα σταθερού μεγέθους τύπου Single. Αυτός ο τύπος της εργασίας μπορεί να χρησιμοποιηθεί σε σενάρια όπως κατανόηση του αισθήματος των σχολίων στο Twitter ως "θετικό" ή "αρνητικό", διάγνωση εάν ένας ασθενής έχει κάποια ασθένεια ή όχι, λήψη απόφασης για επισήμανση ενός μηνύματος ηλεκτρονικού ταχυδρομείου ως "ανεπιθύμητου" ή όχι και για τον προσδιορισμό εάν μια φωτογραφία περιέχει ένα συγκεκριμένο αντικείμενο ή όχι, όπως ένα σκύλο ή ένα φρούτο [26].

- Multiclass classification

Μια εποπτευόμενη εργασία μηχανικής μάθησης που χρησιμοποιείται για την πρόβλεψη της κλάσης (κατηγορία) ενός στιγμιότυπου δεδομένων. Η είσοδος ενός αλγορίθμου ταξινόμησης είναι ένα σύνολο παραδειγμάτων με επισήμανση. Κάθε ετικέτα ξεκινά κανονικά ως κείμενο. Στη συνέχεια εκτελείται μέσω της μεθόδου TermTransform, η οποία το μετατρέπει σε τύπο κλειδιού (αριθμητικό). Η έξοδος ενός αλγορίθμου ταξινόμησης είναι ένας ταξινομητής, τον οποίο μπορούμε να χρησιμοποιήσουμε για να προβλέψουμε την κλάση των νέων στιγμιότυπων χωρίς ετικέτα. Τα δεδομένα εισόδου της στήλης της ετικέτας πρέπει να είναι τύπου κλειδιού. Η στήλη χαρακτηριστικών πρέπει να είναι διάνυσμα σταθερού μεγέθους τύπου Single. Αυτός ο τύπος της εργασίας μπορεί να χρησιμοποιηθεί σε σενάρια όπως ο προσδιορισμός της φυλής ενός σκύλου ως "Σιβηρίας Husky", "Golden Retriever", "Poodle" κ.λπ. , κατανόηση των κριτικών

ταινιών ως "θετικά", "ουδέτερα" ή "αρνητικά", και κατηγοριοποίηση κριτικών ξενοδοχείων σε "τοποθεσία", "τιμή", "καθαριότητα" κ.λπ [26].

- Regression

Μια εποπτευόμενη εργασία μηχανικής εκμάθησης που χρησιμοποιείται για την πρόβλεψη της τιμής της ετικέτας από ένα σύνολο σχετικών χαρακτηριστικών. Η ετικέτα μπορεί να έχει οποιαδήποτε πραγματική τιμή και δεν είναι από ένα πεπερασμένο σύνολο τιμών όπως στις εργασίες ταξινόμησης. Οι αλγόριθμοι παλινδρόμησης μοντελοποιούν την εξάρτηση της ετικέτας από τα σχετιζόμενα χαρακτηριστικά της για να καθορίσουν πώς θα αλλάξει η ετικέτα καθώς οι τιμές των χαρακτηριστικών ποικίλλουν. Η εισαγωγή ενός αλγορίθμου παλινδρόμησης είναι ένα σύνολο στιγμιότυπων με ετικέτες γνωστών τιμών. Η έξοδος ενός αλγορίθμου παλινδρόμησης είναι μια συνάρτηση, την οποία μπορούμε να χρησιμοποιήσουμε για να προβλέψουμε την τιμή της ετικέτας για οποιοδήποτε είσοδο ενός νέου συνόλου χαρακτηριστικών. Τα δεδομένα εισόδου της στήλης της ετικέτας πρέπει να είναι ενιαία. Αυτός ο τύπος της εργασίας μπορεί να χρησιμοποιηθεί σε σενάρια όπως πρόβλεψη τιμών κατοικιών με βάση τα χαρακτηριστικά του σπιτιού, όπως τον αριθμό των υπνοδωματίων, την τοποθεσία ή το μέγεθος. Πρόβλεψη μελλοντικών τιμών μετοχών βάσει ιστορικών δεδομένων και τρεχουσών τάσεων της αγοράς. Πρόβλεψη πωλήσεων ενός προϊόντος βάσει διαφημιστικών προϋπολογισμών [26].

- Clustering

Μια μη εποπτευόμενη εργασία μηχανικής μάθησης που χρησιμοποιείται για την ομαδοποίηση στιγμιότυπων δεδομένων σε ομάδες που περιέχουν παρόμοια χαρακτηριστικά. Η ομαδοποίηση μπορεί επίσης να χρησιμοποιηθεί για τον εντοπισμό σχέσεων σε ένα σύνολο δεδομένων που ενδεχομένως να μην προέρχονται λογικά από την απλή παρατήρηση. Οι είσοδοι και οι έξοδοι ενός αλγορίθμου ομαδοποίησης εξαρτώνται από την επιλεγμένη μεθοδολογία. Μπορούμε να ακολουθήσουμε μια προσέγγιση διανομής, κεντροειδούς, συνδεσιμότητας ή με βάση την πυκνότητα. Το ML.NET υποστηρίζει επί του παρόντος μια κεντροειδή προσέγγιση με χρήση συμπλέγματος K-Means. Τα χαρακτηριστικά εισόδου πρέπει να είναι τύπου Single. Αυτός ο τύπος της εργασίας μπορεί να χρησιμοποιηθεί σε σενάρια όπως κατανόηση των επισκεπτών ενός ξενοδοχείου με βάση τις συνήθειες και τα χαρακτηριστικά των επιλογών του ξενοδοχείου. Προσδιορισμός πελατών και δημογραφικών στοιχείων για τη δημιουργία στοχευμένων διαφημιστικών καμπανιών. Κατηγοριοποίηση αποθέματος με βάση μετρικές κατασκευής [26].

3.4 Αλγόριθμοι στο ML.Net

3.4.1 Γενικά

Για κάθε εργασία ML.NET, υπάρχουν πολλοί αλγόριθμοι μηχανικής μάθησης και η επιλογή εξαρτάται από το προς επίλυση πρόβλημα, τα χαρακτηριστικά των δεδομένων και τους υπολογιστικούς και αποθηκευτικούς πόρους που έχουμε στη διάθεση μας. Είναι σημαντικό να σημειωθεί ότι η εκπαίδευση ενός μοντέλου μηχανικής μάθησης είναι μια επαναληπτική διαδικασία και συνήθως χρειάζεται να δοκιμάσουμε πολλούς αλγόριθμους για να βρούμε αυτόν που λειτουργεί καλύτερα. Οι αλγόριθμοι επενεργούν σε χαρακτηριστικά (features). Τα χαρακτηριστικά αυτά είναι αριθμητικές τιμές που υπολογίζονται από τα δεδομένα εισόδου και αποτελούν βέλτιστες εισροές για αλγόριθμους μηχανικής μάθησης. Πρώτα όμως πρέπει να μετατρέψουμε τα ακατέργαστα δεδομένα προς εισαγωγή σε χαρακτηριστικά χρησιμοποιώντας έναν ή περισσότερους μετασχηματισμούς δεδομένων. Για παράδειγμα, τα δεδομένα κειμένου μετατρέπονται σε ένα σύνολο μετρήσεων λέξεων και μετρήσεων συνδυασμού λέξεων. Μόλις εξαχθούν τα χαρακτηριστικά από έναν τύπο ακατέργαστων δεδομένων χρησιμοποιώντας μετασχηματισμούς δεδομένων, τα δεδομένα αναφέρονται ως “χαρακτηρισμένα” (featurized). Ένας αλγόριθμος είναι τα μαθηματικά που εκτελούνται για την παραγωγή ενός μοντέλου. Διαφορετικοί αλγόριθμοι παράγουν μοντέλα με διαφορετικά χαρακτηριστικά. Με το ML.NET, ο ίδιος αλγόριθμος μπορεί να εφαρμοστεί σε διαφορετικές εργασίες. Για παράδειγμα, ο Stochastic Dual Coordinate Ascent μπορεί να χρησιμοποιηθεί για Binary Classification, Multiclass Classification και Regression. Η διαφορά είναι στο πώς ερμηνεύεται η έξοδος του αλγορίθμου για να ταιριάζει με την εργασία. Για κάθε συνδυασμό αλγορίθμου / εργασίας, το ML.NET παρέχει ένα δομικό στοιχείο (component) που εκτελεί τον αλγόριθμο εκπαίδευσης και κάνει την ερμηνεία. Αυτά τα δομικά στοιχεία ονομάζονται εκπαιδευτές [27].

3.4.2 Γραμμικοί αλγόριθμοι (Linear)

Οι γραμμικοί αλγόριθμοι παράγουν ένα μοντέλο που υπολογίζει τις βαθμολογίες από έναν γραμμικό συνδυασμό των δεδομένων εισαγωγής και ενός συνόλου βαρών. Τα βάρη είναι παράμετροι του μοντέλου που υπολογίζονται κατά τη διάρκεια της εκπαίδευσης. Οι γραμμικοί αλγόριθμοι λειτουργούν καλά για χαρακτηριστικά που διαχωρίζονται γραμμικά. Πριν από την προπόνηση με έναν γραμμικό αλγόριθμο, τα χαρακτηριστικά πρέπει να ομαλοποιηθούν. Αυτό αποτρέπει ένα χαρακτηριστικό από το να έχει μεγαλύτερη επιρροή στο αποτέλεσμα από άλλα. Σε γενικές γραμμές, οι γραμμικοί αλγόριθμοι είναι επεκτάσιμοι, γρήγοροι, οικονομικοί στην εκπαίδευση και στη πρόβλεψη. Κλιμακώνονται ανάλογα με τον αριθμό των χαρακτηριστικών και με το μέγεθος του συνόλου δεδομένων εκπαίδευσης. Οι γραμμικοί αλγόριθμοι κάνουν πολλαπλά περάσματα πάνω στα δεδομένα εκπαίδευσης. Εάν το σύνολο δεδομένων γεμίζει τη μνήμη, προσθέτοντας ένα σημείο ελέγχου κρυφής μνήμης στον αγωγό

ML.NET πριν προστεθεί ο εκπαιδευτής, θα επιτρέψει την εκπαίδευση να εκτελεστεί πιο γρήγορα [27].

Αλγόριθμος	Ιδιότητες	Εκπαιδευτές
Averaged perceptron	Ιδανικός για ταξινόμηση κειμένου	AveragedPerceptronTrainer
Stochastic dual coordinated ascent	Δεν απαιτείται ρύθμιση για καλή προεπιλεγμένη απόδοση	SdcaLogisticRegressionBinaryTrainer SdcaNonCalibratedBinaryTrainer SdcaMaximumEntropyMulticlassTrainer SdcaNonCalibratedMulticlassTrainer SdcaRegressionTrainer
L-BFGS	Χρησιμοποιήστε το όταν ο αριθμός των χαρακτηριστικών είναι μεγάλος. Παράγει στατιστικά εκπαίδευσης λογιστικής παλινδρόμησης, αλλά δεν κλιμακώνεται όπως ο AveragedPerceptronTrainer	LbfgsLogisticRegressionBinaryTrainer LbfgsMaximumEntropyMulticlassTrainer LbfgsPoissonRegressionTrainer
Symbolic stochastic gradient descent	Γρηγορότερος και πιο ακριβής γραμμικός εκπαιδευτής δυαδικής ταξινόμησης. Κλιμακώνεται ομαλά ανάλογα με τον αριθμό των επεξεργαστών	SymbolicSgdLogisticRegressionBinaryTrainer

Πίνακας 3.1 Γραμμικοί αλγόριθμοι

3.4.3 Αλγόριθμοι δέντρων αποφάσεων (Decision trees)

Οι αλγόριθμοι δέντρων αποφάσεων δημιουργούν ένα μοντέλο που περιέχει μια σειρά αποφάσεων και ουσιαστικά ένα διάγραμμα ροής μέσω των τιμών των δεδομένων. Τα χαρακτηριστικά δεν χρειάζεται να διαχωρίζονται γραμμικά για να χρησιμοποιήσουν αυτόν τον τύπο αλγορίθμου και δεν χρειάζεται να ομαλοποιηθούν, επειδή οι μεμονωμένες τιμές στο διάνυσμα των χαρακτηριστικών χρησιμοποιούνται ανεξάρτητα στη διαδικασία λήψης αποφάσεων. Οι αλγόριθμοι δέντρων αποφάσεων είναι γενικά πολύ ακριβείς. Οι αλγόριθμοι δέντρων αποφάσεων καταναλώνουν περισσότερους πόρους και δεν κλιμακώνονται όπως και οι γραμμικοί. Αποδίδουν καλά σε σύνολα δεδομένων που μπορούν να χωρέσουν στη μνήμη. Τα δέντρα ενισχυμένης απόφασης (Boosted decision trees) είναι ένα σύνολο μικρών δέντρων όπου κάθε δέντρο βαθμολογεί τα δεδομένα εισόδου και περνά τη βαθμολογία στο επόμενο δέντρο για να παράγει μια καλύτερη βαθμολογία, και ούτω καθεξής, όπου κάθε δέντρο στο σύνολο βελτιώνεται από το προηγούμενο [27].

Αλγόριθμος	Ιδιότητες	Εκπαιδευτές
Light gradient boosted machine	Γρηγορότερος και ακριβέστερος από τους εκπαιδευτές δυαδικής ταξινόμησης δέντρων. Υψηλές δυνατότητες βελτίωσης	LightGbmBinaryTrainer LightGbmMulticlassTrainer LightGbmRegressionTrainer LightGbmRankingTrainer
Fast tree	Χρησιμοποιείται για δεδομένα εικόνας. Ανθεκτικό σε μη ισορροπημένα δεδομένα. Υψηλές δυνατότητες βελτίωσης	FastTreeBinaryTrainer FastTreeRegressionTrainer FastTreeTweedieTrainer FastTreeRankingTrainer
Fast forest	Λειτουργεί καλά με δεδομένα που έχουν πολύ "θόρυβο"	FastForestBinaryTrainer FastForestRegressionTrainer
Generalized additive model (GAM)	Ιδανικό για προβλήματα που αποδίδουν καλά με αλγόριθμους δέντρων, αλλά όπου η επεξηγηματικότητα είναι προτεραιότητα	GamBinaryTrainer GamRegressionTrainer

Πίνακας 3.2 Αλγόριθμοι δέντρων αποφάσεων

3.4.4 Αλγόριθμοι παραγοντοποίησης πινάκων (Matrix factorization)

Η οικογένεια μηχανών παραγοντοποίησης είναι μια ισχυρή ομάδα μοντέλων για εποπτευόμενα μαθησιακά προβλήματα. Παρουσιάστηκε για πρώτη φορά στο έγγραφο Steffen Rendle's Factorization Machines το 2010. Αργότερα, μία από τις γενικευμένες εκδόσεις της, η μηχανή παραγοντοποίησης με γνώμονα το πεδίο (field-aware factorization machine), έγινε μια σημαντική μονάδα πρόβλεψης σε πρόσφατα συστήματα προτάσεων και διαγωνισμούς πρόβλεψης αναλογίας κλικ προς αριθμό εμφανίσεων όπως είναι οι λύσεις που κέρδισαν στο Steffen Rendle's KDD-Cup του 2012 (Track 1 and Track 2), και Criteo's, Avazu's και Outbrain για τις προκλήσεις πρόβλεψης κλικ στο Kaggle. Οι μηχανισμοί παραγοντοποίησης είναι ιδιαίτερα ισχυροί όταν οι συνδέσεις χαρακτηριστικών συσχετίζονται εξαιρετικά με το σήμα που θέλουμε να προβλέψουμε. Ένα παράδειγμα ζευγών χαρακτηριστικών που μπορούν να σχηματίσουν σημαντικές συνδέσεις είναι το User ID και το Music ID στις προτάσεις μουσικής. Όταν ένα σύνολο δεδομένων αποτελείται μόνο από πυκνά αριθμητικά

χαρακτηριστικά, δεν συνιστάται η χρήση μηχανής παραγοντοποίησης ή θα πρέπει να εκτελεστούν υπό προϋποθέσεις [27].

3.4.5 Μετά-αλγόριθμοι (Meta algorithms)

Αλγόριθμος	Ιδιότητες	Εκπαιδευτές
One versus all	Αυτός ο ταξινομητής πολλαπλών κλάσεων εκπαιδεύει έναν δυαδικό ταξινομητή για κάθε κατηγορία, ο οποίος διακρίνει αυτήν την κλάση από όλες τις άλλες κλάσεις. Περιορίζεται σε κλίμακα από τον αριθμό των κλάσεων που πρέπει να κατηγοριοποιηθούν	OneVersusAllTrainer<BinaryClassificationTrainer>
Pairwise coupling	Αυτός ο ταξινομητής πολλαπλών κλάσεων εκπαιδεύει έναν δυαδικό αλγόριθμο ταξινόμησης σε κάθε ζεύγος κλάσεων. Περιορίζεται σε κλίμακα από τον αριθμό των κλάσεων, καθώς πρέπει να εκπαιδευτεί κάθε συνδυασμός δύο κλάσεων	PairwiseCouplingTrainer<BinaryClassificationTrainer>

Πίνακας 3.3 Μετά-αλγόριθμοι

ΚΕΦΑΛΑΙΟ 4

4.1 Επιλογή τεχνικής και αλγορίθμου

Σε συνέχεια των προαναφερθέντων, ο αλγόριθμός που επιλέχθηκε για την υλοποίηση του project είναι ο FastTree, η λειτουργία του οποίου περιγράφεται στο κεφάλαιο 2.9 . Αποτελεί μια αποτελεσματική εφαρμογή του αλγορίθμου MART ο οποίος χρησιμοποιεί τη τεχνική Gradient Boosting όπως αυτά περιγράφονται στα κεφάλαια 2.8 και 2.7 αντίστοιχα.

Η τεχνική Gradient Boosting μπορεί να χρησιμοποιηθεί στον τομέα της εκμάθησης κατάταξης (Learning to rank). Οι μηχανές αναζήτησης Yahoo και Yandex χρησιμοποιούν παραλλαγές της τεχνικής αυτής στις μηχανικής μάθησης μηχανές κατάταξης που χρησιμοποιούν. Ανάμεσα στα πλεονεκτήματα της τεχνικής αυτής είναι η υψηλή προγνωστική της ακρίβεια, η μεγάλη ευελιξία της όσον αφορά τη βελτιστοποίηση της σε διαφορετικά εύρη απωλειών και η μη ανάγκη για προ-επεξεργασία των δεδομένων [28].

Για τη προσομοίωση ερωτημάτων μέσω της παραγωγής ζευγών τυχαίων αριθμών χρησιμοποιήθηκε η κλάση Random του .NET Framework η οποία αντιπροσωπεύει μια γεννήτρια ψευδοτυχαίων αριθμών, η οποία είναι ένας αλγόριθμος που παράγει μια ακολουθία αριθμών που πληρούν ορισμένες στατιστικές απαιτήσεις. Οι ψευδοτυχαίοι αριθμοί επιλέγονται με ίση πιθανότητα από ένα πεπερασμένο σύνολο αριθμών. Οι επιλεγμένοι αριθμοί δεν είναι εντελώς τυχαίοι επειδή χρησιμοποιείται ένας μαθηματικός αλγόριθμος για την επιλογή τους, αλλά είναι αρκετά τυχαίοι για πρακτικούς σκοπούς. Η τρέχουσα υλοποίηση της κλάσης Random βασίζεται σε μια τροποποιημένη έκδοση του αλγορίθμου γεννήτριας αφαιρετικών τυχαίων αριθμών του Donald E. Knuth [29]. Για τον υπολογισμό της διαφοράς μεταξύ διαστημάτων, μέγιστη-ελάχιστη τιμή των ερωτημάτων και μέγιστη-ελάχιστη τιμή των δεδομένων των κόμβων χρησιμοποιήθηκαν οι τεχνικές Length distance και Overlap [30] που παραθέτονται στη παρακάτω εικόνα.

Lengths distance:

$$D_L(p,q) := 1 - \frac{\min\{||e_p - s_p||, ||e_q - s_q||\}}{\max\{||e_p - s_p||, ||e_q - s_q||\}}$$

Overlap:

$$D_O(p,q) := 1 - \frac{||p \cap q||}{\min\{||e_p - s_p||, ||e_q - s_q||\}}$$

with the interval

$$p \cap q = \begin{cases} (\max\{s_p, s_q\}, \min\{e_p, e_q\}) & \text{for } \max\{s_p, s_q\} < \min\{e_p, e_q\} \\ 0 & \text{else} \end{cases}$$

Εικόνα 4.1 Τεχνικές Length distance και Overlap [30]

4.2 Περιγραφή αλγορίθμου

Αρχικά κατά την εκτέλεση του αλγορίθμου αρχικοποιούμε το περιβάλλον ML.NET. Έπειτα εκπαιδεύουμε το μοντέλο μας με βάση ένα σύνολο δεδομένων που έχουμε επιλέξει. Η διαδικασία της εκπαίδευσης του μοντέλου μας προϋποθέτει τη κατάλληλη παραμετροποίηση των δεδομένων, την έξοδο που περιμένουμε, τα πεδία που θα χρησιμοποιήσουμε καθώς και τη μέθοδο εκπαίδευσης. Όπως αναφέρθηκε και περιγράφηκε παραπάνω η μέθοδος εκπαίδευσης στη συγκεκριμένη υλοποίηση είναι η FastTree. Η παραπάνω διαδικασία έχει ως αποτέλεσμα την επιστροφή του εκπαιδευμένου πλέον μοντέλου μας. Για την εκπαίδευση του μοντέλου μας δημιουργήθηκε ένα dataset με 6 στήλες. Οι 5 πρώτες αφορούν τιμές δεδομένων από το 0 έως το 1 με ένα δεκαδικό ψηφίο και η 6η αντιπροσωπεύει την ομοιότητα τους μέσω

της σχέσης 1 – μ.ο. των τιμών. Έχουν χρησιμοποιηθεί όλοι δυνατοί συνδυασμοί τιμών στο προαναφερθέν εύρος δημιουργώντας ένα training dataset περίπου 6000 εγγραφών. Στη παρακάτω εικόνα φαίνεται ένα δείγμα του dataset χρησιμοποιήθηκε για την εκπαίδευση του μοντέλου.

Πεδίο1	Πεδίο2	Πεδίο3	Πεδίο4	Πεδίο5	Ομοιότητα
0.1,	0.1,	0.1,	0.1,	0.1,	0.9
0.1,	0.1,	0.1,	0.1,	0.2,	0.88
0.1,	0.1,	0.1,	0.1,	0.3,	0.86
0.1,	0.1,	0.1,	0.1,	0.4,	0.84
0.1,	0.1,	0.1,	0.1,	0.5,	0.82
0.1,	0.1,	0.1,	0.1,	0.6,	0.8
0.1,	0.1,	0.1,	0.1,	0.7,	0.78
0.1,	0.1,	0.1,	0.1,	0.8,	0.76
0.1,	0.1,	0.1,	0.1,	0.9,	0.74
0.1,	0.1,	0.1,	0.2,	0.1,	0.88
0.1,	0.1,	0.1,	0.2,	0.2,	0.86
0.1,	0.1,	0.1,	0.2,	0.3,	0.84
0.1,	0.1,	0.1,	0.2,	0.4,	0.82
0.1,	0.1,	0.1,	0.2,	0.5,	0.8
0.1,	0.1,	0.1,	0.2,	0.6,	0.78
0.1,	0.1,	0.1,	0.2,	0.7,	0.76
0.1,	0.1,	0.1,	0.2,	0.8,	0.74
0.1,	0.1,	0.1,	0.2,	0.9,	0.72
0.1,	0.1,	0.1,	0.3,	0.1,	0.86
0.1,	0.1,	0.1,	0.3,	0.2,	0.84

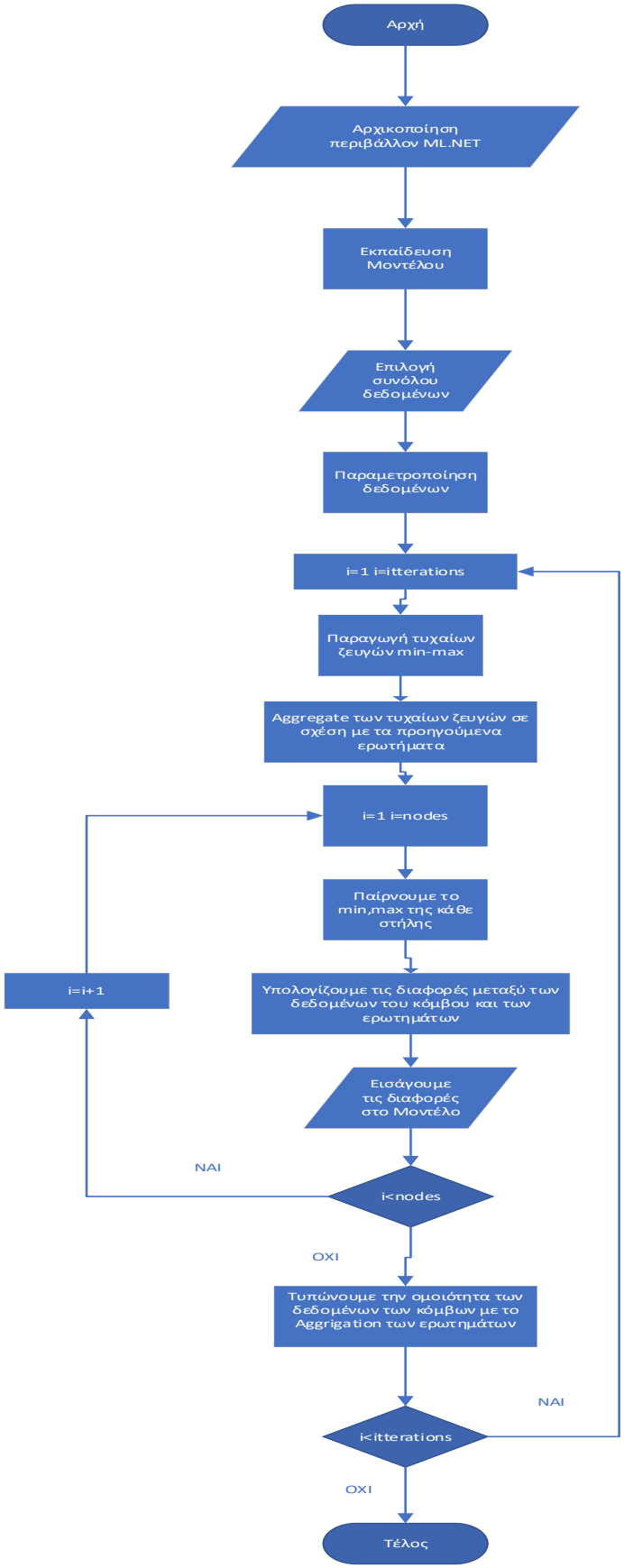
Εικόνα 4.2 Σύνολο δεδομένων για εκπαίδευση

Στη συνέχεια επιλέγουμε ένα σύνολο δεδομένων το οποίο το οποίο τμηματοποιείτε κατάλληλα σε όσα τμήματα επιθυμούμε και τα οποία στη προσομοίωση μας θα αναπαριστούν τους κατανεμημένους κόμβους. Στη περίπτωση μας χρησιμοποιήθηκε το γνωστό σύνολο δεδομένων iris [31] που είναι ίσως η πιο γνωστή βάση δεδομένων που μπορεί να βρεθεί στη βιβλιογραφία για την αναγνώριση προτύπων, τροποποιημένο με τέτοιο τρόπο ώστε και η τελευταία του στήλη να περιέχει αριθμητική τιμή και όχι χαρακτήρες.

Έπειτα για κάθε επαναληπτική διαδικασία παράγουμε τυχαία ζεύγη αριθμών τα οποία και προσομοιώνουν τα ερωτήματα μας προς τους κόμβους. Τα παραγόμενα ζεύγη συγκρίνονται με προηγούμενα ερωτήματα και γίνεται μία συσσωμάτωση αυξάνοντας όπου χρειάζεται το διάστημα αυτών. Υπολογίζουμε την απόσταση των δεδομένων της στήλης του κάθε κόμβου (μέγιστο - ελάχιστο) με αυτά των ερωτημάτων. Οι τιμές αυτές

τροφοδοτούν το μοντέλο μας το οποίο μας επιστρέφει μία πρόβλεψη ως προς την ομοιότητα των ερωτημάτων και των δεδομένων των κόμβων.

Η ροή του αλγορίθμου αποτυπώνεται στο παρακάτω διάγραμμα ροής.

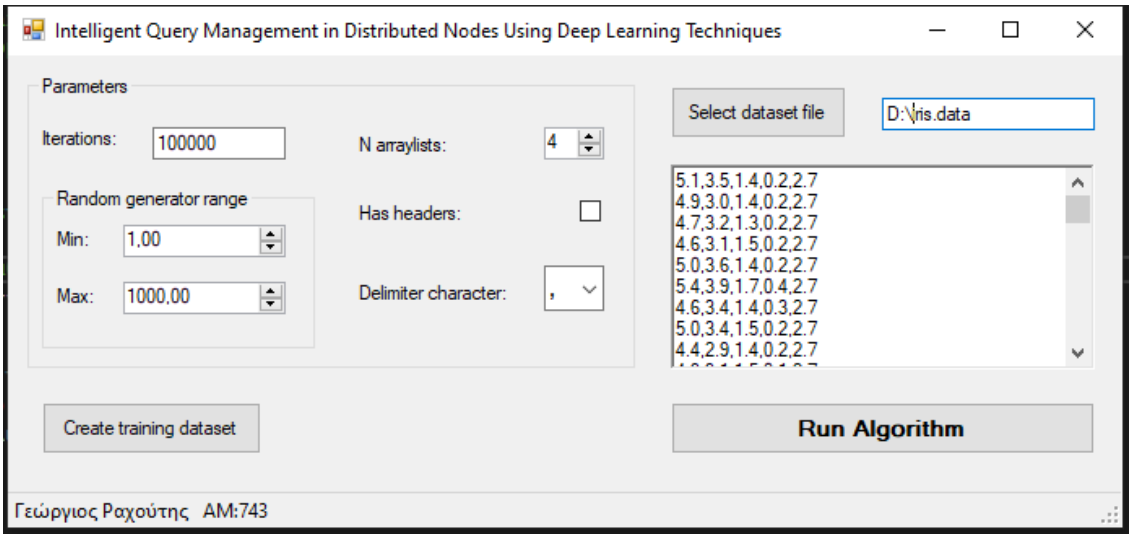


Εικόνα 4.3 Διάγραμμα ροής αλγορίθμου

4.3 Περιγραφή περιβάλλοντος

Η χρήση του λογισμικού που αναπτύχθηκε γίνεται μέσα από γραφικό περιβάλλον, δίνοντας πέρα από ευκολία στη χρήση, μεγάλες δυνατότητες παραμετροποίησης κάθε φορά που εκτελούμε τον αλγόριθμο που περιγράφηκε. Η παραμετροποίηση έγκειται στην επιλογή των επαναλήψεων που θα τρέξει ο αλγόριθμος (iterations), την επιλογή των αριθμών των λιστών που θα παίζουν το ρόλο των κατανεμημένων κόμβων (arraylists), την επιλογή να διαγράψουμε τις κεφαλίδες από το επιλεγμένο dataset που θα τρέξει ο αλγόριθμος (Has headers) καθώς και την επιλογή του χαρακτήρα που διαχωρίζει τις τιμές (delimiter character). Υπάρχει επίσης η δυνατότητα να επιλέξουμε το εύρος των τυχαίων αριθμών που θα παράξει ο αλγόριθμος υπό το ρόλο ερωτημάτων στους κατανεμημένους κόμβους, ώστε να συμβαδίζει με το εύρος των τιμών του dataset που έχουμε επιλέξει για πιο ρεαλιστική προσέγγιση.

Επίσης δίνεται η δυνατότητα να αναζητήσουμε και να επιλέξουμε το αρχείο δεδομένων της επιλογής μας (select dataset file) και για το οποίο υπάρχει προεπισκόπηση. Για την εκπαίδευση του μοντέλου μας έχει δημιουργηθεί ένα αρχείο όπως περιγράφεται στη παράγραφο 4.2 αλλά δύναται να δημιουργηθεί και να χρησιμοποιηθεί διαφορετικό (Create training dataset).



Εικόνα 4.4 Περιβάλλον λογισμικού

4.4 Περιγραφή βασικών κλάσεων

Form1()

- Θέτουμε delimited χαρακτήρα το σύμβολο «;»

- Αρχικοποιούμε το περιβάλλον ML.NET
- Καλούμε τη μέθοδο train για να εκπαιδεύσουμε το μοντέλο μας με ορίσματα Train(mlContext, _trainDataPath);
- Το MlContext είναι το default και το trainDataPath είναι η διαδρομή του dataset που έχουμε δημιουργήσει για να εκπαιδεύσουμε το μοντέλο μας (TrainingDataset.csv)
- Καλούμε τη μέθοδο Train()

Train (mlContext, _trainDataPath)

- Παραμετροποιούμε το dataset που χρησιμοποιούμε για την εκπαίδευση του μοντέλου μας
- Δηλώνουμε ποια θα είναι η έξοδος που περιμένουμε, ορίζουμε τα πεδία που θα χρησιμοποιήσουμε, και τη μέθοδο εκπαίδευσης η οποία είναι η FastTree
- Επιστρέφουμε το εκπαιδευμένο μοντέλο από το dataset που του δώσαμε

openFileBtn_Click()

- Επιλέγουμε κατάλληλο αρχείο με δεδομένα. Η επιλογή dataset ξεκινάει από τη διαδρομή c:\ και ψάχνει κατάλληλα αρχεία .data.csv, .kaggle
- Κάνει preview το αρχείο που επιλέξαμε
- Καλούμε την loadData()

loadData()

- Διαβάζει το επιλεγμένο dataset σε γραμμές από string
- Ελέγχουμε αν έχει επικεφαλίδες το αρχείο και τις αφαιρούμε
- Μετράμε πόσα πεδία έχει η κάθε γραμμή για να τα μεταφράσουμε σε στήλες
- Δημιουργούμε dynamic list of dynamic objects από τη κάθε γραμμή
- Ορίζουμε σε πόσα arraylists έχουμε χωρίσει το dataset
- Υπολογίζουμε τον αριθμό των εγγραφών του κάθε arraylist

- Κατανέμουμε τις ανάλογες εγγραφές σε κάθε arraylist (Κόμβο)

runFileData()

- Καλεί την algorithm()

algorithm()

- Εκτελούμε τυχαία ερωτήματα στους κόμβους
- Εκτελούμε τόσα ερωτήματα όσα iterations έχουμε επιλέξει στην εφαρμογή
- Για κάθε iteration
    - Παράγουμε ζεύγη τυχαίων αριθμών min - max για κάθε στήλη (3 στήλες – 3 ζεύγη τυχαίων αριθμών) χρησιμοποιούμε τη κλάση **Random()** ως γεννήτρια τυχαίων αριθμών
    - Κρατάμε το aggregation των τυχαίων τιμών και το οποίο είναι αυτό που θα συγκρίνουμε με τα min-max των στηλών
    - Τα συγκρίνουμε με αυτά που έχουμε ήδη (τα aggregation) για να κρατήσουμε το interval οπότε τα αυξάνουμε αν χρειαστεί
    - Για κάθε arraylist (Κόμβο)
        - Αρχικοποιούμε τη λίστα που κρατάμε τις διαφορές
        - Σε αυτή τη λίστα θα αποθηκεύουμε το Min , Max του κόμβου για κάθε στήλη
        - Τη στήλη κάθε κόμβου τη μετατρέπουμε σε λίστα
        - Παίρνουμε το Min, Max της κάθε στήλης του επιλεγμένου κόμβου
        - Υπολογίζουμε τις διαφορές μεταξύ των δεδομένων του κόμβου (Min,Max) και του interval των ερωτημάτων με την **calculateIntervalDistance()**
        - Δημιουργούμε ένα αντικείμενο από τις παραπάνω τιμές (Διαφορά1, Διαφορά2, Διαφορά3, ομοιότητα)

- Εισάγουμε στο μοντέλο το αντικείμενο που δημιουργήσαμε με την **Predict**(mlContext, model, dataSample)
- Το μοντέλο επιστρέφει έναν αριθμό που αντικατοπτρίζει την ομοιότητα
- Τυπώνουμε την ομοιότητα του ερωτήματος σε σχέση με τον επιλεγμένο κόμβο

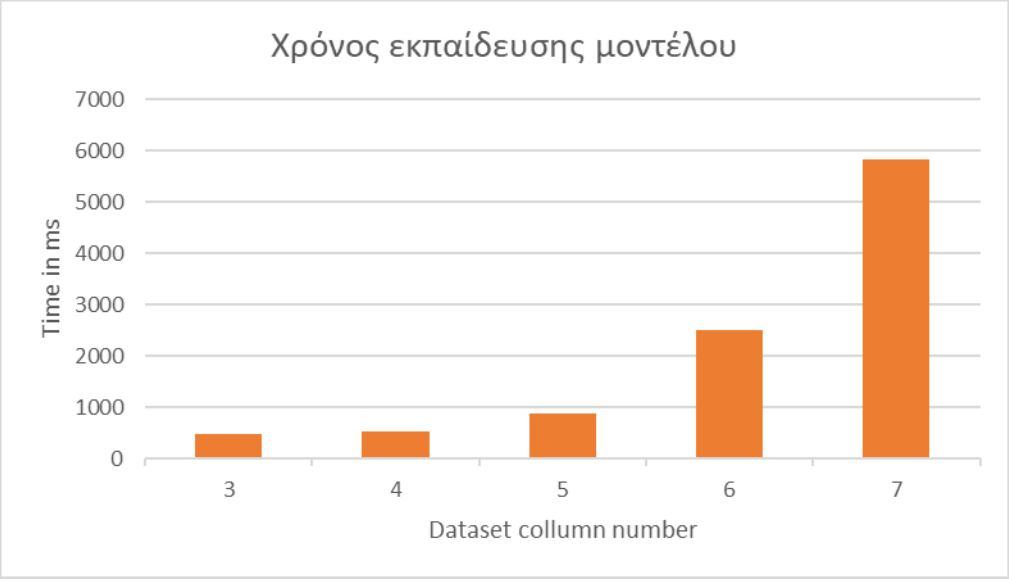
calculateIntervalDistance()

- Υπολογίζουμε τη διαφορά δύο διαστημάτων με τη μέθοδο Overlap όπως φαίνεται στην εικόνα 3.3

4.5 Πειραματική αποτίμηση

4.5.1 Εκπαίδευση μοντέλου

Για την εκπαίδευση του μοντέλου όπως περιγράφηκε και στη παράγραφο 4.2 χρησιμοποιήθηκε ένα dataset 5 στηλών. Παρόλο αυτά δοκιμάστηκαν διάφορα dataset με διαφορετικό πλήθος στηλών τα οποία είχαν επίπτωση όσον αφορά το χρόνο που χρειαζόταν το καθένα ώστε να εκπαιδευτεί το μοντέλο και το οποίο απεικονίζονται στην εικόνα 4.5.



Εικόνα 4.5 Χρόνος εκπαίδευσης μοντέλου

4.5.2 Παραγωγή τυχαίων αριθμών

Για κάθε επαναληπτική διαδικασία παράγονται τυχαία ζεύγη αριθμών τα οποία και προσομοιώνουν τα ερωτήματα μας προς τους κόμβους. Για κάθε σετ πειραμάτων το εύρος τυχαίων αριθμών κυμαίνεται από 1 έως 100 για το $1^{ο}$ σετ και από 1 έως 1000 για τι $2^{ο}$ σετ. Στο $3^{ο}$ και $4^{ο}$ σετ χρησιμοποιούνται και τα δύο εύρη αριθμών. Όπως περιγράφεται και στη παράγραφο 4.1 για τη παραγωγή τυχαίων αριθμών χρησιμοποιήθηκε η κλάση Random του .NET Framework η οποία αντιπροσωπεύει μια γεννήτρια ψευδοτυχαίων αριθμών, η οποία είναι ένας αλγόριθμος που παράγει μια ακολουθία αριθμών που πληρούν ορισμένες στατιστικές απαιτήσεις. Οι ψευδοτυχαίοι αριθμοί επιλέγονται με ίση πιθανότητα από ένα πεπερασμένο σύνολο αριθμών. Στο πρώτο σετ πειραμάτων τα ερωτήματα που φτάνουν στους κόμβους ελέγχονται ανεξάρτητα το ένα από το άλλο. Στο δεύτερο σετ πειραμάτων τα παραγόμενα ζεύγη συγκρίνονται με προηγούμενα ερωτήματα και γίνεται μία συσσωμάτωση των τιμών των ερωτημάτων αυξάνοντας όπου χρειάζεται το διάστημα αυτών. Στο τρίτο σετ πειραμάτων πέρα από τη συσσωμάτωση που περιγράφηκε παραπάνω οι τιμές των ερωτημάτων προστίθενται στα δεδομένα των κόμβων προσομοιώνοντας προς αποθήκευση δεδομένα στους κόμβους. Στο τέταρτο σετ πειραμάτων τα ερωτήματα ελέγχονται ανεξάρτητα το ένα από το άλλο όπως στο πρώτο σετ αλλά οι τιμές των ερωτημάτων προστίθενται στα δεδομένα των κόμβων προσομοιώνοντας προς αποθήκευση δεδομένα στους κόμβους όπως στο τρίτο σετ.

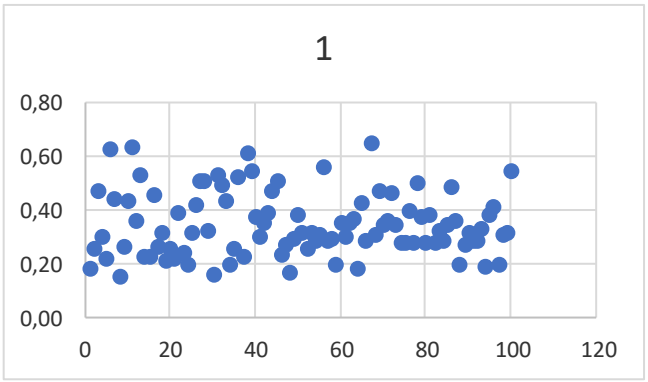
4.5.3 Υπολογισμός μετρικών

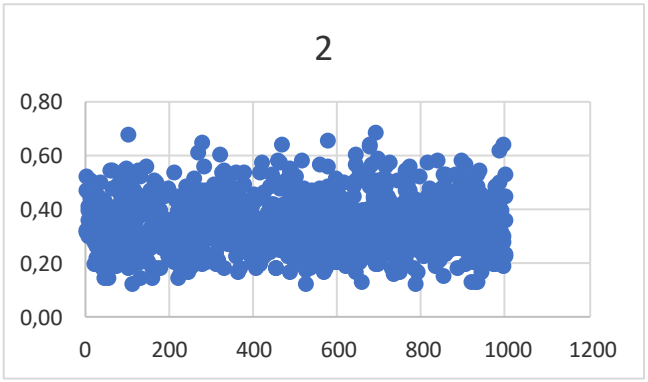
Μετά τη παραγωγή των τυχαίων αριθμών υπολογίζετε η απόσταση των δεδομένων της στήλης του κάθε κόμβου (μέγιστο - ελάχιστο) με αυτά των ερωτημάτων. Οι τιμές αυτές τροφοδοτούν το εκπαιδευμένο μοντέλο μας το οποίο μας επιστρέφει μία πρόβλεψη ως προς την ομοιότητα των ερωτημάτων και των δεδομένων των κόμβων. Σε κάθε επαναληπτική διαδικασία η επιλογή του εκάστοτε κόμβου γίνεται τυχαία. Στο δεύτερο και τρίτο σετ πειραμάτων λόγω της συσσωμάτωσης των τιμών των ερωτημάτων και την αύξηση του προς σύγκριση εύρους τιμών, μειώνεται το ελάχιστο και αυξάνεται το μέγιστο, παρατηρείται ότι μετά από κάποιες επαναλήψεις η ομοιότητα αυξάνεται και κάποια στιγμή σταθεροποιείται σε μία τιμή σχετικά υψηλή επειδή τα δεδομένα του κόμβου (ελάχιστο-μέγιστο) τείνουν να προσεγγίσουν αυτά των ερωτημάτων. Το πλήθος των επαναλήψεων/ερωτημάτων που επιλέχθηκαν να γίνουν σε κάθε σετ πειραμάτων είναι εκατό και χίλια αντίστοιχα. Οι παράμετροι των πειραμάτων φαίνονται στους παρακάτω πίνακες.

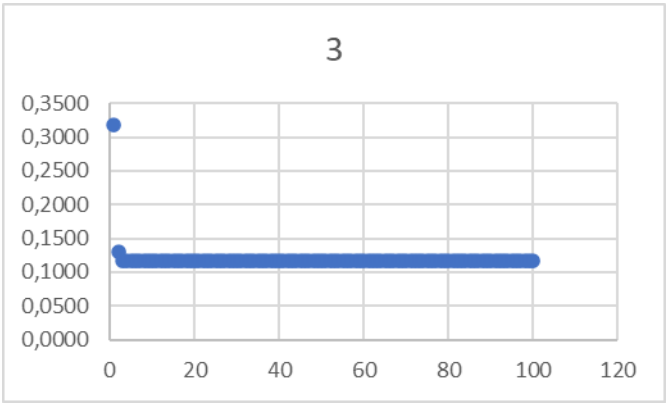
Πίνακας 4.1 1ο Σετ πειραμάτων

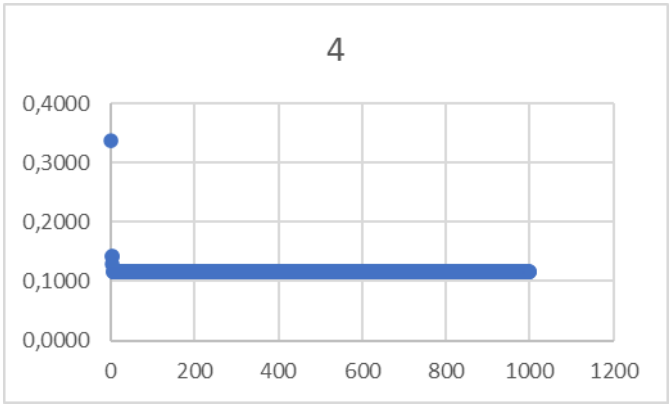
A/A	Πλήθος Dataset	Επαναλήψεις	Εύρος ερωτημάτων	Aggregation	Εισαγωγή ερωτημάτων στους κόμβους	Κόμβοι	Στήλες
1	10000	100	1-100	ΟΧΙ	ΟΧΙ	4	5
2	10000	1000	1-100	ΟΧΙ	ΟΧΙ	4	5
3	100	100	1-100	ΟΧΙ	ΟΧΙ	4	5
4	100	1000	1-100	ΟΧΙ	ΟΧΙ	4	5

Πίνακας 4.2 Αποτίμηση 1ου Σετ πειραμάτων





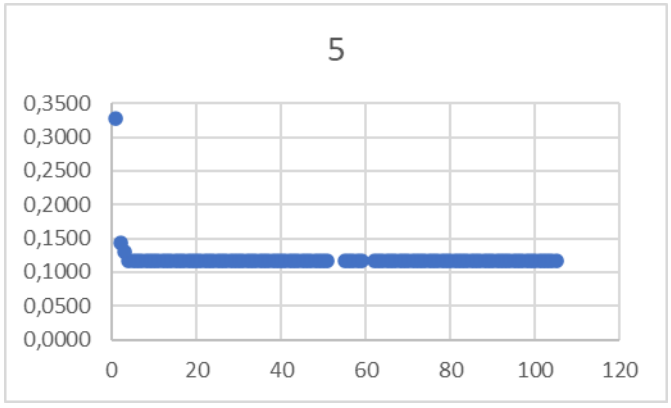


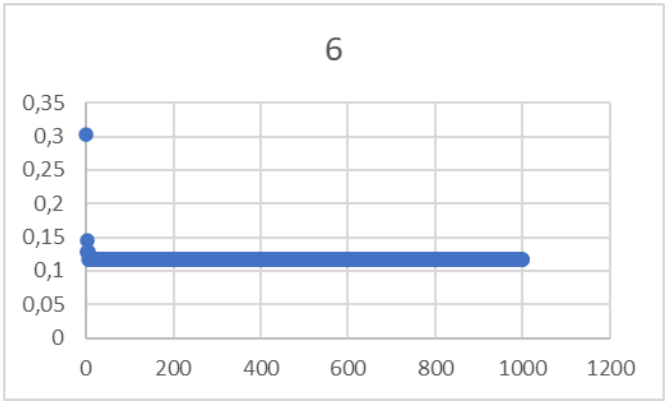


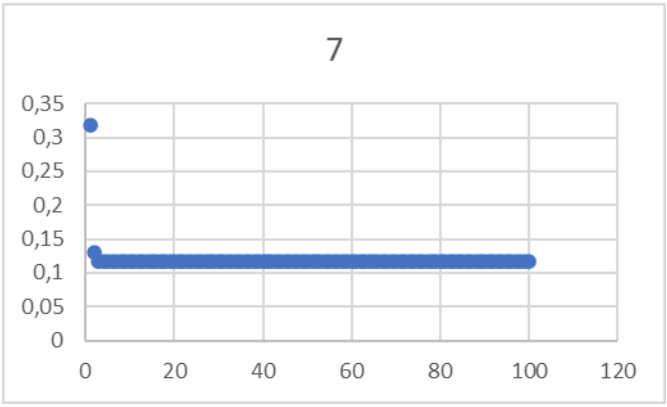
Πίνακας 4.3 2ο Σετ πειραμάτων

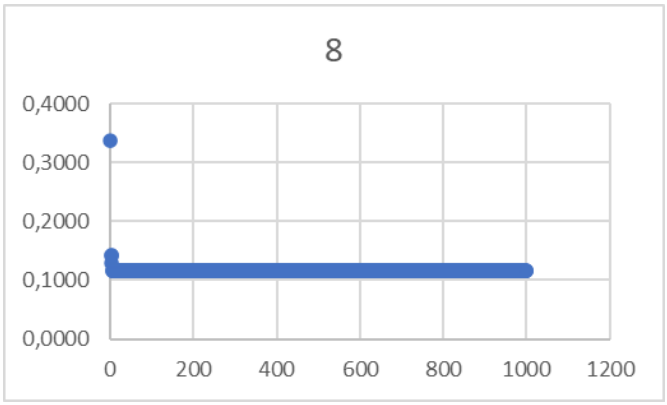
A/A	Πλήθος Dataset	Επαναλήψεις	Εύρος ερωτημάτων	Aggregation	Εισαγωγή ερωτημάτων στους κόμβους	Κόμβοι	Στήλες
5	10000	100	1-1000	ΝΑΙ	ΟΧΙ	4	5
6	10000	1000	1-1000	ΝΑΙ	ΟΧΙ	4	5
7	100	100	1-1000	ΝΑΙ	ΟΧΙ	4	5
8	100	1000	1-1000	ΝΑΙ	ΟΧΙ	4	5

Πίνακας 4.4 Αποτίμηση 2ου Σετ πειραμάτων





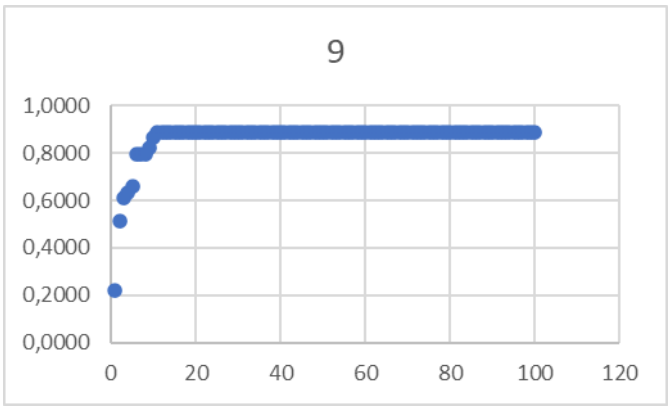


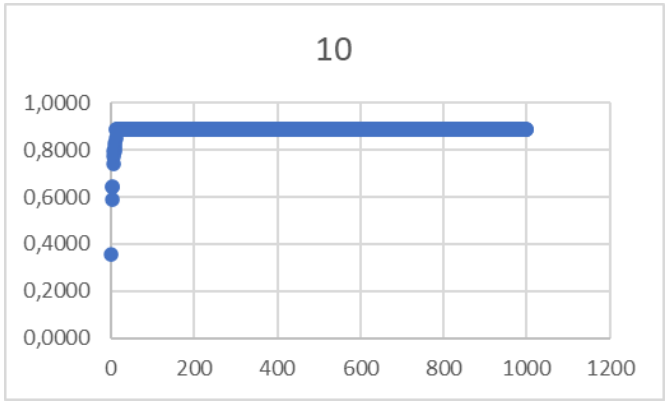


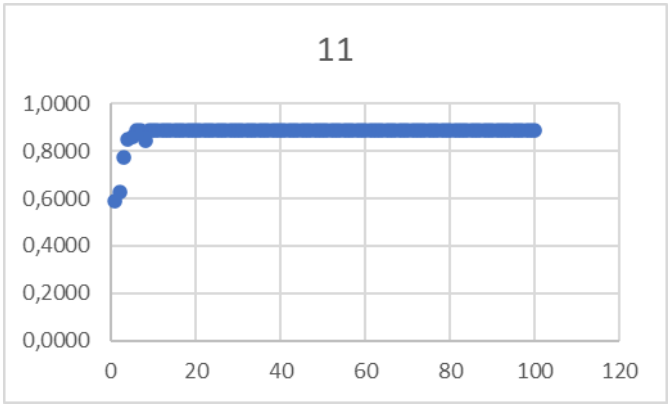
Πίνακας 4.5 3ο Σετ πειραμάτων

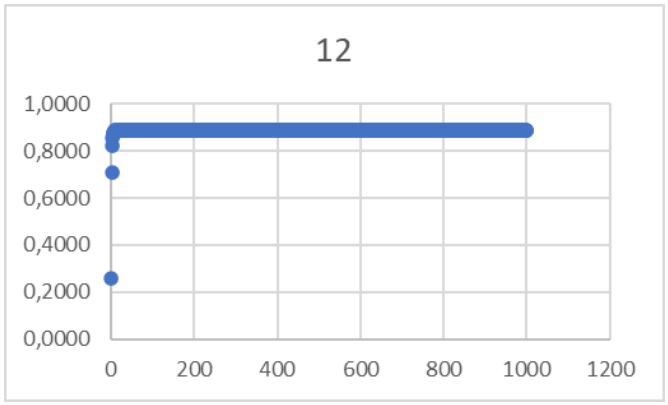
A/A	Πλήθος Dataset	Επαναλήψεις	Εύρος ερωτημάτων	Agregation	Εισαγωγή ερωτημάτων στους κόμβους	Κόμβοι	Στήλες
9	10000	100	1-100	NAI	NAI	4	5
10	10000	1000	1-100	NAI	NAI	4	5
11	100	100	1-100	NAI	NAI	4	5
12	100	1000	1-100	NAI	NAI	4	5
13	10000	100	1-1000	NAI	NAI	4	5
14	10000	1000	1-1000	NAI	NAI	4	5
15	100	100	1-1000	NAI	NAI	4	5
16	100	1000	1-1000	NAI	NAI	4	5

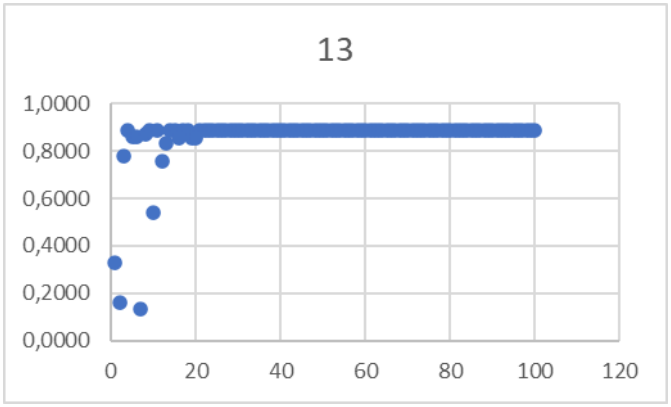
Πίνακας 4.6 Αποτίμηση 3ου Σετ πειραμάτων

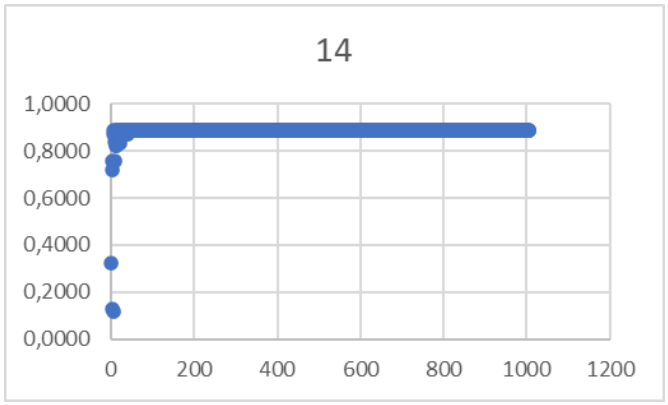


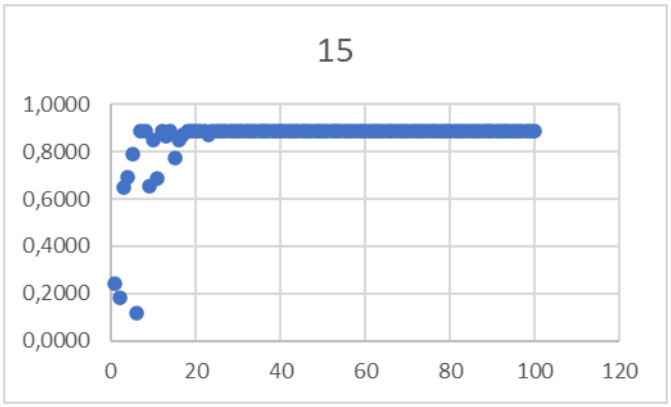


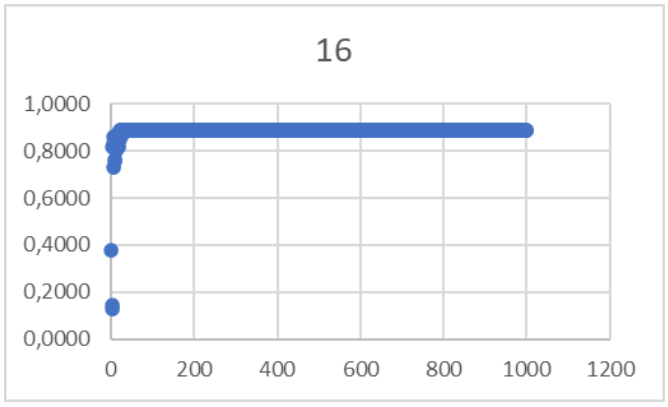





13
1,0000
0,8000
0,6000
0,4000
0,2000
0,0000
0
20
40
60
80
100
120


14
1,0000
0,8000
0,6000
0,4000
0,2000
0,0000
0
200
400
600
800
1000
1200

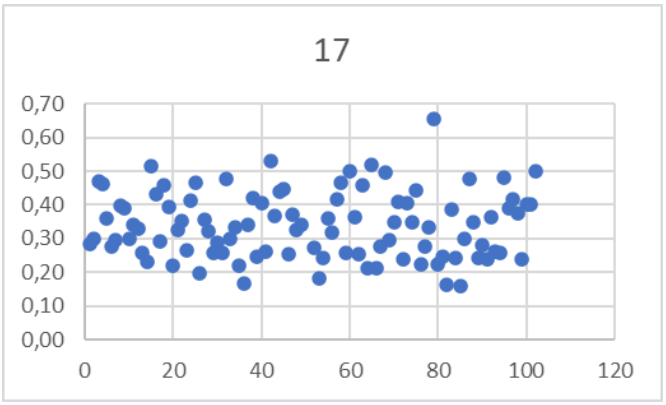

15
1,0000
0,8000
0,6000
0,4000
0,2000
0,0000
0
20
40
60
80
100
120

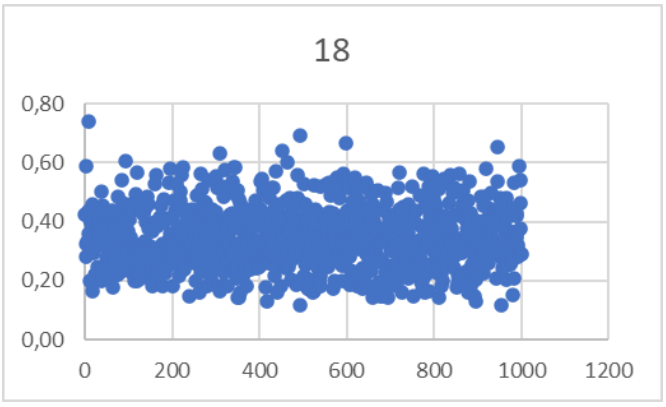

16
1,0000
0,8000
0,6000
0,4000
0,2000
0,0000
0
200
400
600
800
1000
1200

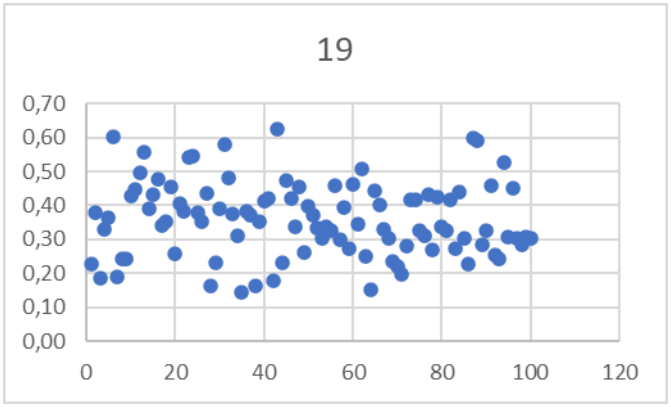
Πίνακας 4.7 4ο Σετ πειραμάτων

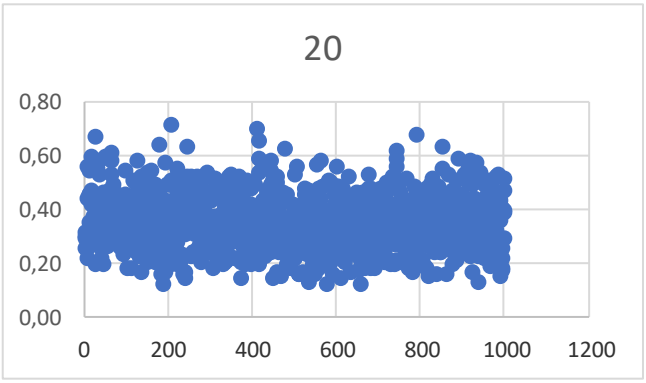
A/A	Πλήθος Dataset	Επαναλήψεις	Εύρος ερωτημάτων	Aggregation	Εισαγωγή ερωτημάτων στους κόμβους	Κόμβοι	Στήλες
17	10000	100	1-100	OXI	NAI	4	5
18	10000	1000	1-100	OXI	NAI	4	5
19	100	100	1-100	OXI	NAI	4	5
20	100	1000	1-100	OXI	NAI	4	5
21	10000	100	1-1000	OXI	NAI	4	5
22	10000	1000	1-1000	OXI	NAI	4	5
23	100	100	1-1000	OXI	NAI	4	5
24	100	1000	1-1000	OXI	NAI	4	5

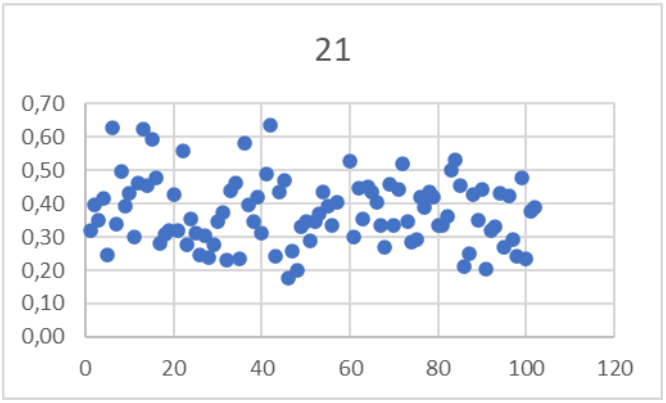
Πίνακας 4.8 Αποτίμηση 4ου Σετ πειραμάτων

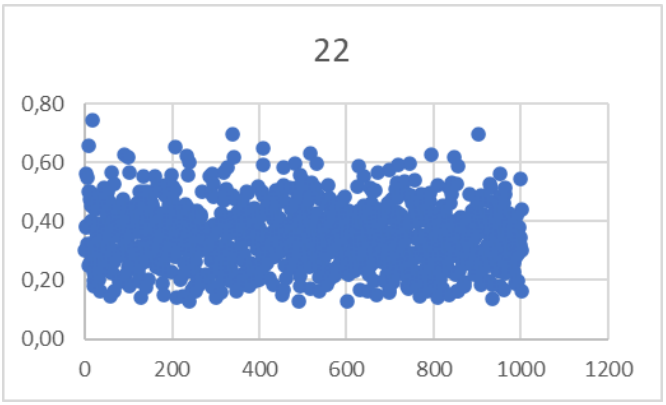


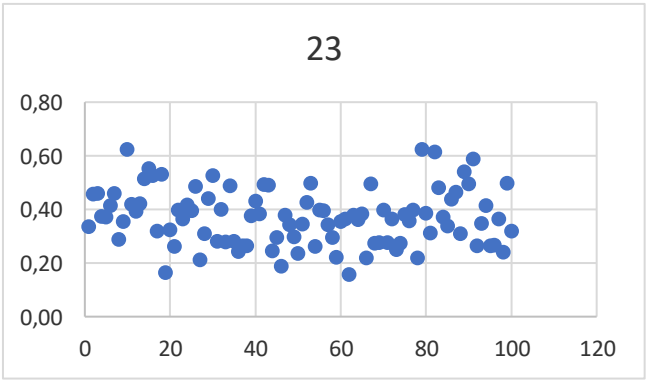


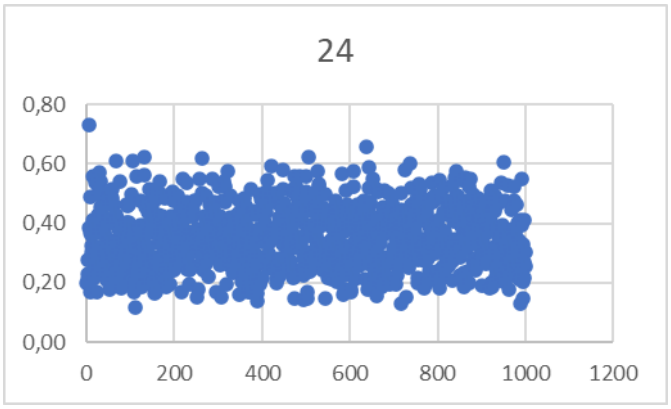

19
0,70
0,60
0,50
0,40
0,30
0,20
0,10
0,00
0
20
40
60
80
100
120


20
0,80
0,60
0,40
0,20
0,00
0
200
400
600
800
1000
1200


21
0,70
0,60
0,50
0,40
0,30
0,20
0,10
0,00
0
20
40
60
80
100
120


22
0,80
0,60
0,40
0,20
0,00
0
200
400
600
800
1000
1200





4.5.4 Συμπεράσματα

Όσον αφορά το εργαλείο ανάπτυξης και το framework που χρησιμοποιηθήκε, μπορούμε να συμπεράνουμε ότι είναι σχετικά εύκολο να δημιουργήσουμε εφαρμογές μηχανικής εκμάθησης χρησιμοποιώντας το ML.NET χρησιμοποιώντας τη γλώσσα προγραμματισμού C#. Με αυτό το πλαίσιο, η Microsoft έχει κάνει αρκετά εύκολο για τους προγραμματιστές .NET που διστάζουν να μάθουν python, το να δημιουργήσουν εφαρμογές μηχανικής εκμάθησης , χρησιμοποιόντας τις υπάρχουσες δεξιότητες και τις βιβλιοθήκες που έχουν μάθει ως προγραμματιστές .NET.

Επιπρόσθετα της αρχικής επιλογής του σωστού αλγορίθμου, η βελτίωση της απόδοσης του μοντέλου απαιτεί πρόσθετη γνώση και πειραματισμό για την επίτευξη βέλτιστης απόδοσης. Μερικές από τις τεχνικές που μπορούννα χρησιμοποιήθούν για να βελτιώσουμε το μοντέλο μας είναι η κανονικοποίηση (Regularization) η επάυξηση των δεδομένων (Data Augmentation). και η Μηχανική Χαρακτηριστικών (Feature Engineering). Η μεν πρώτη χρησιμοποιείται για την αποφυγή της υπερπροσαρμογής, όπου το μοντέλο είναι πολύ περίπλοκο και ταιριάζει πολύ κοντά στα δεδομένα εκπαίδευσης, οδηγώντας τελικά σε κακή απόδοση γενίκευσης σε νέα δεδομένα. Έτσι, η τακτοποίηση περιλαμβάνει την προσθήκη ενός όρου ποινής στη συνάρτηση απώλειας που ενθαρρύνει το μοντέλο να έχει απλούστερα βάρη ή συντελεστές. Η δεύτερη πρακτική περιλαμβάνει τη δημιουργία πρόσθετων δεδομένων εκπαίδευσης με την

εφαρμογή μετασχηματισμών στα υπάρχοντα δεδομένα. Η επαύξηση μπορεί να βοηθήσει στην αύξηση του μεγέθους του σετ εκπαίδευσης και, επομένως, στη βελτίωση της ικανότητας του μοντέλου να γενικεύεται σε νέα δεδομένα. Τέλος η τρίτη πρακτική πραγματεύεται την επιλογή ή τη δημιουργία νέων χαρακτηριστικών από τα πρωτογενή δεδομένα που είναι πιο ενημερωτικά ή σχετικά με την εργασία πρόβλεψης. Αυτό μπορεί να περιλαμβάνει την κλιμάκωση ή τον μετασχηματισμό υπαρχόντων χαρακτηριστικών, τη δημιουργία νέων χαρακτηριστικών από τα υπάρχοντα ή την επιλογή ενός υποσυνόλου από τα πιο σημαντικά χαρακτηριστικά.

Παρατηρώντας τις γραφικές παραστάσεις σε συνάρτηση πάντα με τα δεδομένα του κάθε σετ πειραμάτων μπορούν να εξαχθούν κάποια γενικά συμπεράσματα. Όσο μεγαλύτερο είναι το πλήθος των δεδομένων του dataset σε συνδυασμό με το μεγαλύτερο πλήθος των ερωτημάτων παρατηρείται μία αυξητική τάση στην ομοιότητα που παράγει το μοντέλο μεταξύ των τιμών των δύο αυτών ομάδων δεδομένων. Η συσσωμάτωση (aggregation) των τιμών των ερωτημάτων με προηγούμενα ερωτήματα αυξάνοντας όπου χρειάζεται το διάστημα αυτών φαίνονται ότι δε προσφέρει θετικά στην αποτίμηση καθώς είναι πολύ πιθανόν τα τυχαία ερωτήματα που παράγονται στα αρχικά στάδια να τείνουν σε μέγιστες η ελάχιστες τιμές και πολύ γρήγορα στην επαναλήπτικη διαδικασία να παρατηρείται ενα μέγιστο – ελάχιστο αυτών το οποίο διατηρείται μέχρι το τέλος της διαδικασίας με αποτέλεσμα η τιμή της ομοιότητας που παράγει το μοντέλο να παραμένει σταθερά υψηλή. Μία διαφορετική προσέγγιση της συσσωμάτωσης των τιμών αυτών θέτοντας όρια στην αποδεκτή απόκλιση ίσως πρόσφερε θετικά στη πειραματική αποτίμηση.

ΠΑΡΑΡΤΗΜΑ

Κώδικας λογισμικού

```
using FileHelpers;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Dynamic;

using System.Globalization;

using System.IO;

using System.Linq;
```

```
using System.Windows.Forms;

using Microsoft.ML;



namespace PtixiakiApp

{

   public  partial  class Form1 : Form

   {

      //Σε πόσα ArrayLists χωρίζουμε το dataset

      int arrayListNo;

      //Πόσες στήλες έχει το dataset

      int collumnNo;

      //Λίστα που έχει μέσα της τα ArrayLists

      List<List<dynamic>> myArrayLists = new List<List<dynamic>>();

      //Λιστα που κρατάει τα aggregation

      List<Minmax> aggregation = new List<Minmax>();

      //Λίστα των M διαφορών μεταξύ aggregation κομβων και aggregated των ερωτημάτων

      List<decimal> nodeQueriesDiff = new List<decimal>();

      //Λίστα με τα δεδομένα για το training dataset

      List<List<Decimal>> trainingDataset = new List<List<decimal>>();



      //Η διαδρομή για το training dataset

      //static readonly string _trainDataPath = Path.Combine(Environment.CurrentDirectory, "Data", "taxi-fare-train.csv");

      static readonly string _trainDataPath = Path.Combine(Environment.CurrentDirectory, "Data", "TrainingDataset.csv");

      //static readonly string _testDataPath = Path.Combine(Environment.CurrentDirectory, "Data", "taxi-fare-test.csv");

      //Η διαδρομή όπου θα σωθεί το μοντέλο που θα δημιουργήσουμε
```

```
    static readonly string _modelPath = Path.Combine(Environment.CurrentDirectory, "Data",
"Model.zip");


    MLContext mlContext;
    ITransformer model;


    int myNode;



    public class Minmax
    {
        public decimal min { get; set; }
        public decimal max { get; set; }
    }


    public Form1()
    {
        InitializeComponent();
        //Επιλεγμένος χαρακτήρας delimiter to ","
        delimitCharCombo.SelectedIndex = 0;
        //Ενεργοποιούνται τα tooltips
        setTooltips();


        //Αρχικοποιούμε το περιβάλλον ML.NET
        mlContext = new MLContext(seed: 0);
        //Καλούμε τη μέθοδο train για να εκπαιδεύσουμε το μοντέλο μας
        model = Train(mlContext, _trainDataPath);
        //Evaluate(mlContext, model);
    }
```

```
//Επιλέγουμε το αρχείο με το dataset που θέλουμε
private void openFileBtn_Click(object sender, EventArgs e)
{
   //η επιλογή dataset ξεκινάει από το c:\ και ψαχνει κατάλληλα αρχεία .data.csv .kaggle
   OpenFileDialog openFileDialog1 = new OpenFileDialog
   {
      InitialDirectory = @"C:\",
      Title = "Browse Text Files",
      CheckFileExists = true,
      CheckPathExists = true,
      Filter = "txt files (*.txt)|*.txt|UCI files (*.data)|*.data|kaggle files (*.csv)|*.csv",
      FilterIndex = 2,
      RestoreDirectory = true,
   };

   if (openFileDialog1.ShowDialog() == DialogResult.OK)
   {
      filePathTxt.Text = openFileDialog1.FileName;
   }

   previewFileTxt.Text = File.ReadAllText(filePathTxt.Text);

   loadData();
}

//Εκτελείτε ο αλγόριθμος
private void runFileData_Click(object sender, EventArgs e)
```

```
{
    //Εκτελούμε τον αλγόριθμό μας
    algorithm();
}

//Φορτώνουμε-χωρίζουμε τις εγγραφές από το επιλεγμένο dataset σε όσα arraylist έχουμε επιλέξει τα οποία και θα είναι οι κόμβοι μας
private void loadData()
{
    myArrayLists.Clear();

    if (filePathTxt.Text == string.Empty)
    {
        MessageBox.Show("Παρακαλώ επιλέξτε αρχείο", "Προειδοποίηση", MessageBoxButtons.OK, MessageBoxIcon.Warning);
    }
    else
    {
        //Διαβαζει το επιλεγμένο dataset σε γραμμές από string
        char delimiter = delimitCharCombo.SelectedItem.ToString()[0];
        string[] lines = File.ReadAllLines(filePathTxt.Text);
        var dynamicList = new List<dynamic>();

        //Ελέγχουμε αν έχει επικεφαλίδες το αρχείο και τις αφαιρούμε
        if (hasHeadersCheck.Checked == true)
        {
            lines = lines.Skip(1).ToArray();
        }
```

```
//Μετράμε πόσα πεδία έχει η κάθε γραμμή για να τα μεταφράσουμε σε στήλες
string[] myRow = lines[0].Split(delimiter);
//arrayCounter = myRow.Length;
collumnNo = myRow.Length;

//Επιλέγει ο χρήστης σε ποιο column θα κάνουμε ερωτήματα
//selectedColumn.Minimum = 1;
//selectedColumn.Maximum = arrayCounter;

//Load dynamic list of dynamic objects
foreach (string line in lines)
{
   if (line != string.Empty)
   {
      string[] col = line.Split(delimiter);
      IDictionary<string, object> obj = new ExpandoObject();

      for (int i = 0; i < col.Length; i++)
      {
         var propName = "Column" + (i + 1).ToString();
         obj[propName] = col[i];
      }

      dynamicList.Add(obj);
   }
}

//Σε πόσα arraylists έχουμε χωρίσει το dataset
```

```
        arrayListNo = Convert.ToInt32(arraylistCounter.Value);
        //ο αριθμός των εγγραφών του κάθε arraylist
        int recordsNo = dynamicList.Count() / arrayListNo;
        //Το επιλεγμένο arraylist
        //int selected = Convert.ToInt32(selectedArray.Value);

        //
        for(int i = 1; i <= arrayListNo; i++)
        {
           //Η τελευταία εγγραφή του επιλεγμένου arraylis σε σχέση με τις συνολικές
           int endpoint = i * recordsNo;
           //Η πρώτη εγγραφή του επιλεγμένου arraylis σε σχέση με τις συνολικές
           int startpoint = endpoint - recordsNo + 1;
           //Οι εγγραφές του επιλεγμένου arraylist
           List<dynamic> selectedRecords = dynamicList.GetRange(startpoint - 1, recordsNo);
           //Προσθέτουμε το κάθε arraylist στο συνολικό myArrayLists
           myArrayLists.Add(selectedRecords);
        }
     }
  }

  //Εκτελούμε τυχαία ερωτήματα σε τυχαίους κόμβους
  private void algorithm()
  {
     int iterations = Convert.ToInt32(iterationsTxt.Text);
     decimal minRange = minRangeCntrl.Value;
     decimal maxRange = maxRangeCntrl.Value;
     Random rng = new Random();
```

```
//Εκτελούμε τόσα ερωτήματα όσα iterations εχουμε επιλέξει
for (int iteration = 1; iteration <= iterations; iteration++)
{
   //Αρχικοποιούμε τη λίστα που κρατάμε τις διαφορές
   nodeQueriesDiff.Clear();

   //Επιλέγουμε έναν τυχαίο κόμβο   ---------- 1 ----------
   int randomNode = rng.Next(1, arrayListNo);
   myNode = randomNode;
   //Φορτώνουμε τα δεδομένα του τυχαίου κόμβου
   List<dynamic> selectedNode = myArrayLists[randomNode - 1];

   //Μετατρέπουμε τα δεδομένα του κόμβου σε List<DataRow>
   DataTable myDatatable = ToDataTable(selectedNode);
   List<DataRow> dataRowList = myDatatable.AsEnumerable().ToList();

   //Σε αυτή τη λίστα θα αποθηκεύουμε το Min , Max του κόμβου για κάθε στήλη
   List<Minmax> nodeMinMax = new List<Minmax>();

   //Παίρνουμε το Min , Max της κάθε στήλης του επιλεγμένου κόμβου   ---------- 2 ----------
   for (int column = 0; column < collumnNo; column++)
   {
      List<decimal> myColumnData = new List<decimal>();

      foreach (var dataRow in dataRowList)
      {
         decimal myNum;
```

```
                if (!decimal.TryParse(dataRow.ItemArray[column].ToString(), out myNum))
                {
                    break;
                }
                else
                {
                    myNum = Convert.ToDecimal(dataRow.ItemArray[column].ToString(), new
CultureInfo("en-US"));
                    myColumnData.Add(myNum);
                }
            }

            if(myColumnData.Count > 0)
            {
                //columnMinMax.Add(new Tuple<decimal, decimal>(myColumnData.Min(),
myColumnData.Max()));
                Minmax colMinMax = new Minmax();
                colMinMax.min = myColumnData.Min();
                colMinMax.max = myColumnData.Max();
                nodeMinMax.Add(colMinMax);
            }
        }

        //Θα παράξουμε ζευγάρια τυχαίων αριθμών Min , Max που θα προσομειώνουν τα ερωτήματα
---------- 3,4 ----------
        for (int i = 0; i < nodeMinMax.Count(); i++)
        {
            decimal minValue = NextDecimal(rng, minRange, maxRange);
            decimal maxValue = NextDecimal(rng, minRange, maxRange);
```

```
            while (minValue > maxValue)
            {
                maxValue = NextDecimal(rng, minRange, maxRange);
            }

            //Τα συγκρίνουμε με αυτά που έχουμε ήδη για να κρατήσουμε το interval οπότε τα αυξάνουμε αν χρειαστει
            if(aggregation.Count() > nodeMinMax.Count())
            {
                if (minValue < aggregation[i].min)
                {
                    aggregation[i].min = Math.Round(minValue, 2);
                }
                if (maxValue > aggregation[i].max)
                {
                    aggregation[i].max = Math.Round(maxValue, 2);
                }
            }
             else
             {
                Minmax firstMinMax = new Minmax{ min = minValue , max = maxValue};
                aggregation.Add(firstMinMax);
             }
        }

        //Υπολογίζουμε τις διαφορές μεταξύ των δεδομένων του κόμβου (Min,Max) και του interval των ερωτημάτων  ---------- 5 ----------
        for (int i = 0; i < nodeMinMax.Count(); i++)
        {
```

```
            decimal diff = calculateIntervalDistance(nodeMinMax[i], aggregation[i]);


            //Κάνουμε normalize τις διαφορές ώστε να είναι στο διάστημα (0,1) ------ 6 --------- διαιρούμε τη κάθε τιμή με το max της αντίστοιχης στήλης
            diff = diff / nodeMinMax[i].max;


            nodeQueriesDiff.Add(diff);
        }


        //Δημιουργούμε ένα αντικείμενο από τις παραπάνω τιμές
        var dataSample = new DatasetClass()
        {
            First = (float)nodeQueriesDiff[0],
            Second = (float)nodeQueriesDiff[1],
            Third = (float)nodeQueriesDiff[2],
            Fourth = (float)nodeQueriesDiff[3],
            Fifth = (float)nodeQueriesDiff[4],
            Fitting = 0 // To predict. Actual/Observed = 15.5
        };


        //Είσάγουμε στο μοντέλο το αντικείμενο που δημιουργήσαμε
        Predict(mlContext, model, dataSample, myNode);
    }


    MessageBox.Show("Πέρας διαδικασίας", "Ειδοποίηση", MessageBoxButtons.OK);
}


#region Machine learning methods
```

```
    //Η μέθοδος όπου θα εκπαιδεύσουμε το μοντέλο μας

    public static ITransformer Train(MLContext mlContext, string dataPath)

    {

        //Παραμετροποιούμε το dataset που χρησιμοποιούμε για την εκπαίδευση του μοντέλου μας

        IDataView dataView = mlContext.Data.LoadFromTextFile<DatasetClass>(dataPath, hasHeader:
false, separatorChar: ',');

        //Δηλώνουμε ποιά θα είναι η έξοδος που περιμένουμε, οριζουμε τα πεδία που θα
χρησιμοποιήσουμε, και τη μέθοδο εκπαίδευσης η οποία είναι η FastTree

        var pipeline = mlContext.Transforms.CopyColumns(outputColumnName: "Label",
inputColumnName: "Fitting")

        .Append(mlContext.Transforms.Concatenate("Features", "First", "Second", "Third", "Fourth",
"Fifth"))

        .Append(mlContext.Regression.Trainers.FastTree());



        //Επιστρέφουμε το εκπαιδευμένο μοντέλο από το dataset που του δώσαμε

        var model = pipeline.Fit(dataView);

        return model;

    }



    //Η μέθοδος που θα κάνει τη πρόβλεψη

    private static void Predict(MLContext mlContext, ITransformer model, DatasetClass dataSample, int
node)

    {



        //Ορίζουμε την είσοδο και την έξοδο στο μοντέλο μας

        var predictionFunction = mlContext.Model.CreatePredictionEngine<DatasetClass,
DatasetPrediction>(model);

        //Το μοντέλο κάνει μία πρόβλεψη

        var prediction = predictionFunction.Predict(dataSample);

        Console.WriteLine($"Predicted Similaritly for node {node.ToString()} : {prediction.Fitting}");

    }
```

```
#endregion

#region Helper Methods

//Μετατρέπουμε μία δυναμική λίστα σε datatable
public static DataTable ToDataTable(IEnumerable<dynamic> items)
{
   var data = items.ToArray();
   if (data.Count() == 0) return null;

   var dt = new DataTable();
   foreach (var key in ((IDictionary<string, object>)data[0]).Keys)
   {
      dt.Columns.Add(key);
   }
   foreach (var d in data)
   {
      dt.Rows.Add(((IDictionary<string, object>)d).Values.ToArray());
   }
   return dt;
}

//Μετατρέπουμε τον τυχαίο ακέραιο σε decimal μέσα στο επιλεγμένο διάστημα
public decimal NextDecimal(Random rnd, decimal min, decimal max)
{
   return Convert.ToDecimal(rnd.NextDouble()) * (max - min) + min;
}
```

```
//Ενεργοποιούμε tooltips με πληροφορίες για τα πεδία
private void setTooltips()
{
   ToolTip toolTip1 = new ToolTip();
   toolTip1.SetToolTip(this.iterationsTxt, "The number of iterations");
   toolTip1.SetToolTip(this.arraylistCounter, "The number of arraylists to seperate the records");
   toolTip1.SetToolTip(this.hasHeadersCheck, "To remove the first row that contains headers");
   toolTip1.SetToolTip(this.filePathTxt, ".data and .csv are supported");
}

//Υπολογίζουμε τη διαφορά δύο διαστημάτων με τη μέθοδο Overlap (DOI: 10.18154/RWTH-2017-09590)
public decimal calculateIntervalDistance(Minmax node, Minmax aggregate)
{
   decimal distance;

   Minmax intersect = new Minmax();
   intersect.min = Math.Max(node.min, aggregate.min);
   intersect.max = Math.Min(node.max, aggregate.max);

   Decimal intersectAbs = Math.Abs(intersect.min - intersect.max);
   Decimal dividerAbs = Math.Min((Math.Abs(node.max - node.min)), (Math.Abs(aggregate.max - aggregate.min)));

   if(dividerAbs != 0)
   {
      distance = Math.Abs(1 - (intersectAbs / dividerAbs));
   }
```

```
    else
    {
        distance = 0;
    }

    distance = Math.Round(distance, 2);

    return distance;
}

//Δημιουργούμε το Training dataset
private void createTrainDataset_Click(object sender, EventArgs e)
{
    //Αν δεν υπάρχει φάκελος όπου θα δημιουργηθεί το αρχείο τον δημιουργεί
    if (!Directory.Exists(@"C:\Temp"))
        Directory.CreateDirectory(@"C:\Temp");

    //Το όνομα και η διαδρομή του αρχείου που θα δημιουργηθεί
    string fileName = @"C:\Temp\TrainingDataset.txt";

    //ελεγχει αν υπάρχει το αρχείο και αν ναι το διαγραφει
    if (File.Exists(fileName))
    {
        File.Delete(fileName);
    }
```

```
//Οι τιμές που πρόκειται να χρησιμοποιηθούν για το Training dataset

IEnumerable<Double> m_oEnum = new List<Double>() { 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 };


//Καλούμε τη μέθοδο που θα δημιουργήσει το συνδεασμό δεδομένων για το Training dataset

IEnumerable<IEnumerable<Double>> result =
TrainingDataset.GetPermutationsWithRept(m_oEnum, collumnNo);


//For every row we calculate the similarity field (1-average) and create a string for every row

//Για κάθε εγγραφή υπολογίζουμε και ακόμα ένα παδίο που είναι η ομοιότητα (1 - μ.ο. των
πεδίων)

foreach (var row in result.ToList())

{

   string trainRow = "";

   Double sum = 0;

   Double similarity;

   List<string> lineToAppend = new List<string>();


   //Προσπελαύνουμε τα στοιχεία της κάθε εγγραφής

   for (int i = 0; i < row.Count(); i++)

   {

      sum = sum + row.ToList()[i];

      string value = row.ToList()[i].ToString();

      value = value.Replace(",", ".");

      trainRow = trainRow + "," + value;

   }


   similarity = 1 - (sum / collumnNo);

   trainRow = trainRow + "," + similarity.ToString().Replace(",", ".");

   trainRow = trainRow.Remove(0, 1);
```

```
            lineToAppend.Add(trainRow);

            File.AppendAllLines(fileName, lineToAppend);

        }

    }


    #endregion


  }

}
```

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] B. U. Libraries, "Selecting a data repository," – *Boston University Data Services – Data Services at Boston University*. [Online]. Available: https://www.bu.edu/data/share/selecting-a-data-repository/. [Accessed: 09-Feb-2023]

[2] "Home | re3data.org," Re3data.org, 2019. https://www.re3data.org/

[3] S. Sakr, A. Liu, D. M. Batista, and M. Alomari, "A survey of large scale data management approaches in Cloud Environments," *IEEE Communications Surveys & Tutorials*, vol. 13, no. 3, pp. 311–336, 2011.

[4] P. Valduriez, "Principles of distributed data management in 2020?," *Lecture Notes in Computer Science*, pp. 1–11, 2011.

[5] S. Langella, S. Hastings, S. Oster, T. Kurc, U. Catalyurek, and J. Saltz, "A distributed data management middleware for data-driven application systems," *2004 IEEE International Conference on Cluster Computing (IEEE Cat. No.04EX935)*.

[6] T. P. Raptis and A. Passarella, "A distributed data management scheme for Industrial Iot Environments," *2017 IEEE 13th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, 2017.

[7] A. Chugh, V. K. Sharma, and C. Jain, "Big Data and query optimization techniques," *Advances in Computing and Intelligent Systems*, pp. 337–345, 2020.

[8] "Artificial Intelligence," *Encyclopædia Britannica*, 08-Feb-2023. [Online]. Available: https://www.britannica.com/technology/artificial-intelligence. [Accessed: 09-Feb-2023]

[9] J. Murphree, "Machine learning anomaly detection in large systems," *2016 IEEE AUTOTESTCON*, 2016.

[10] G. Shobha and S. Rangaswamy, "Machine learning," *Handbook of Statistics*, pp. 197–228, 2018.

[11] A. Molodoria, "5 essential machine learning algorithms for Business Applications," *MobiDev*, 06-Sep-2022. [Online]. Available: https://mobidev.biz/blog/5-essential-machine-learning-techniques. [Accessed: 09-Feb-2023]

[12] A. Anand, "Top 6 machine learning techniques," *Analytics Steps*. [Online]. Available: https://www.analyticssteps.com/blogs/top-6-machine-learning-techniques. [Accessed: 09-Feb-2023]

[13] L. Rokach and O. Maimon, "Decision trees," *Data Mining and Knowledge Discovery Handbook*, pp. 165–192.

[14] ProjectPro, "Common machine learning algorithms for beginners," *ProjectPro*, 05-Jul-2022. [Online]. Available: https://www.projectpro.io/article/common-machine-learning-algorithms-for-beginners/202#toc-13. [Accessed: 09-Feb-2023]

[15] *Regression trees*. [Online]. Available: https://help.pyramidanalytics.com/Content/Root/MainClient/apps/Model/Model%20Pro/Data%20Flow/ML/RegressionTrees.htm. [Accessed: 09-Feb-2023]

[16] A. Natekin and A. Knoll, "Gradient Boosting Machines, a tutorial," *Frontiers in Neurorobotics*, vol. 7, 2013.

[17] J. Brownlee, "A gentle introduction to the gradient boosting algorithm for machine learning," *MachineLearningMastery.com*, 14-Aug-2020. [Online]. Available: https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/. [Accessed: 09-Feb-2023]

[18] Vinayak, R. K., & Gilad-Bachrach, R. (2015, February). Dart: Dropouts meet multiple additive regression trees. In *Artificial Intelligence and Statistics* (pp. 489-497). PMLR.

[19] Natke, "FastTreeRegressionTrainer class (Microsoft.ml.trainers.FastTree)," *(Microsoft.ML.Trainers.FastTree) | Microsoft Learn*. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/api/microsoft.ml.trainers.fasttree.fasttreeregressiontrainer?view=ml-dotnet. [Accessed: 09-Feb-2023]

[20] "Microsoft Visual studio," *Wikipedia*, 13-Sep-2022. [Online]. Available: https://en.wikipedia.org/wiki/Microsoft_Visual_Studio. [Accessed: 09-Feb-2023]

[21] TerryGLee, "Overview of visual studio," *Overview of Visual Studio | Microsoft Learn*. [Online]. Available: https://docs.microsoft.com/el-gr/visualstudio/get-started/visual-studio-ide?view=vs-2019. [Accessed: 09-Feb-2023]

[22] "What is .NET framework? A software development framework.," *Microsoft*. [Online]. Available: https://dotnet.microsoft.com/learn/dotnet/what-is-dotnet-framework. [Accessed: 09-Feb-2023]

[23] "What is ML.NET and how does it work? - ML.NET," *ML.NET | Microsoft Learn*. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/machine-learning/how-does-mldotnet-work. [Accessed: 09-Feb-2023]

[24] Z. Ahmed, S. Amizadeh, M. Bilenko, R. Carr, W.-S. Chin, Y. Dekel, X. Dupre, V. Eksarevskiy, S. Filipi, T. Finley, A. Goswami, M. Hoover, S. Inglis, M. Interlandi, N. Kazmi, G. Krivosheev, P. Luferenko, I. Matantsev, S. Matusevych, S. Moradi, G. Nazirov, J. Ormont, G. Oshri, A. Pagnoni, J. Parmar, P. Roy, M. Z. Siddiqui, M. Weimer, S. Zahirazami, and Y. Zhu, "Machine learning at Microsoft with ML.NET," *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019.

[25] "Machine learning tasks - ML.NET," *Machine learning tasks - ML.NET | Microsoft Learn*. [Online]. Available: https://learn.microsoft.com/en-us/dotnet/machine-learning/resources/tasks. [Accessed: 09-Feb-2023]

[26] "Machine learning tasks - ML.NET," *Machine learning tasks - ML.NET | Microsoft Learn*. [Online]. Available: https://learn.microsoft.com/en-us/dotnet/machine-learning/resources/tasks. [Accessed: 09-Feb-2023]

[27] "How to choose an ML.NET algorithm - ML.NET," *ML.NET | Microsoft Learn*. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/machine-learning/how-to-choose-an-ml-net-algorithm. [Accessed: 09-Feb-2023]

[28] "Gradient boosting," *Wikipedia*, 29-Jan-2023. [Online]. Available: https://en.wikipedia.org/wiki/Gradient_boosting#Gradient_tree_boosting. [Accessed: 09-Feb-2023]

[29[D. E. Knuth, *The Art of Computer Programming Volume 2: Seminumerical Algorithms*. Boston: Addison-Wesley, 1998.

[30] Hafiler, M., Jeschke, S., & Meisen, T. (2017). Similarity analysis of time interval data sets regarding time shifts and rescaling.

[31] "UCI Machine Learning Repository: Iris data set." [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Iris/. [Accessed: 12-Feb-2023]