ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΑΝΑΓΩΡΙΣΗ ΑΝΩΜΑΛΩΝ ΠΑΝΩ ΣΕ ΡΟΕΣ ΔΕΔΟΜΕΝΩΝ

ΛΑΡΕΝΤΖΑΚΗΣ ΓΕΩΡΓΙΟΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ

Κωνσταντίνος Κολομβάτσος
Επίκουρος Καθηγητής

Λαμία Νοέμβριος 2023

ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΑΝΑΓΩΡΙΣΗ ΑΝΩΜΑΛΩΝ ΠΑΝΩ ΣΕ ΡΟΕΣ ΔΕΔΟΜΕΝΩΝ

ΛΑΡΕΝΤΖΑΚΗΣ ΓΕΩΡΓΙΟΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ

Κωνσταντίνος Κολομβάτσος
Επίκουρος Καθηγητής

Λαμία Νοέμβριος 2023

«Με ατομική μου ευθύνη και γνωρίζοντας τις κυρώσεις [(1)], που προβλέπονται από της διατάξεις της παρ. 6 του άρθρου 22 του Ν. 1599/1986, δηλώνω ότι:

*1. Δεν παραθέτω κομμάτια βιβλίων ή άρθρων ή εργασιών άλλων αυτολεξεί **χωρίς να τα περικλείω σε εισαγωγικά** και χωρίς να αναφέρω το συγγραφέα, τη χρονολογία, τη σελίδα. Η αυτολεξεί παράθεση χωρίς εισαγωγικά χωρίς αναφορά στην πηγή, είναι λογοκλοπή. Πέραν της αυτολεξεί παράθεσης, λογοκλοπή θεωρείται και η παράφραση εδαφίων από έργα άλλων, συμπεριλαμβανομένων και έργων συμφοιτητών μου, καθώς και η παράθεση στοιχείων που άλλοι συνέλεξαν ή επεξεργάσθηκαν, χωρίς αναφορά στην πηγή. Αναφέρω πάντοτε με πληρότητα την πηγή κάτω από τον πίνακα ή σχέδιο, όπως στα παραθέματα.*

*2. Δέχομαι ότι η αυτολεξεί **παράθεση χωρίς εισαγωγικά**, ακόμα κι αν συνοδεύεται από αναφορά στην πηγή σε κάποιο άλλο σημείο του κειμένου ή στο τέλος του, είναι αντιγραφή. Η αναφορά στην πηγή στο τέλος π.χ. μιας παραγράφου ή μιας σελίδας, δεν δικαιολογεί συρραφή εδαφίων έργου άλλου συγγραφέα, έστω και παραφρασμένων, και παρουσίασή τους ως δική μου εργασία.*

3. Δέχομαι ότι υπάρχει επίσης περιορισμός στο μέγεθος και στη συχνότητα των παραθεμάτων που μπορώ να εντάξω στην εργασία μου εντός εισαγωγικών. Κάθε μεγάλο παράθεμα (π.χ. σε πίνακα ή πλαίσιο, κλπ), προϋποθέτει ειδικές ρυθμίσεις, και όταν δημοσιεύεται προϋποθέτει την άδεια του συγγραφέα ή του εκδότη. Το ίδιο και οι πίνακες και τα σχέδια

4. Δέχομαι όλες τις συνέπειες σε περίπτωση λογοκλοπής ή αντιγραφής.

Ημερομηνία: 02/11/2023

Ο – Η Δηλ.
Λαρεντζάκης Γεώργιος

(1) «Όποιος εν γνώσει του δηλώνει ψευδή γεγονότα ή αρνείται ή αποκρύπτει τα αληθινά με έγγραφη υπεύθυνη δήλωση του άρθρου 8 παρ. 4 Ν. 1599/1986 τιμωρείται με φυλάκιση τουλάχιστον τριών μηνών. Εάν ο υπαίτιος αυτών των πράξεων σκόπευε να προσπορίσει στον εαυτόν του ή σε άλλον περιουσιακό όφελος βλάπτοντας τρίτον ή σκόπευε να βλάψει άλλον, τιμωρείται με κάθειρξη μέχρι 10 ετών.»

ΠΕΡΙΛΗΨΗ

Η συγκεκριμένη εργασία έχει ως σκοπό την μελέτη στη ανίχνευση ακραίων τιμών σε σύνολα δεδομένων. Κατά κύριο λόγο θα δοθεί βαρύτητα στην ανίχνευση αυτών των τιμών πάνω σε στατικά δεδομένα αλλά θα προταθούν και κάποιες ενδεικτικές λύσεις στα χρονικά μεταβαλλόμενα δεδομένα (χρονοσειρές). Η προσέγγιση θα γίνει με αλγορίθμους μη επιβλεπόμενης μάθησης (unsupervised learning) καθώς είναι πιο ρεαλιστική σε προβλήματα τέτοιας φύσης. Θα αναλύσουμε τις ακραίες τιμές, τον ορισμό και τα είδη των μεθόδων εντοπισμού τους. Επίσης θα μιλήσουμε για την εύρεση κατάλληλων αλγορίθμων που βασίζονται στις συγκεκριμένες μεθόδους αλλά και τα χαρακτηριστικά των δεδομένων, των ακραίων τιμών και κατά πόσο όλα αυτά συνδέονται μεταξύ τους. Με βάση τα παραπάνω, θα δημιουργήσουμε μία συνάρτηση ανάλυσης δεδομένων και απόφασης βέλτιστου μοντέλου ανάλογα με τα χαρακτηριστικά των δεδομένων αλλά και των ακραίων τιμών. Το τελικό πρόγραμμα θα κάνει όλα, για όσα θα μιλήσουμε παρακάτω, βασισμένο σε μια συνάρτηση που θα αποφασίζει ποιοι αλγόριθμοι θα χρησιμοποιηθούν ανάλογα με κάποια χαρακτηριστικά των δεδομένων (dataset) και τα αποτελέσματα που προκύπτουν. Το πρόγραμμα θα μπορούσε να χαρακτηριστεί σαν ένα χαμηλού επιπέδου έμπειρο σύστημα (expert system). Οι συναρτήσεις που θα χρησιμοποιήσουμε για το διαχωρισμό των συνόλων δεδομένων σε διάφορες υποομάδες ανάλογα με τα χαρακτηριστικά τους. Την επιλογή των κατάλληλων μοντέλων/αλγορίθμων μηχανικής μάθησης (Machine learning) αλλά και βαθιάς μάθησης (Deep Learning) ανάλογα με την υποομάδα. Τη σημασία και τους λόγους για τους οποίους γίνεται αυτός ο διαχωρισμός θα αναλυθούν στις παρακάτω ενότητες.

Table of Contents

ΚΕΦΑΛΑΙΟ 1 Εισαγωγή

Εδώ και πολλά χρόνια η ανίχνευση ακραίων τιμών ή ανωμαλιών είναι ένα [1]πρόβλημα που απασχολεί τους επιστήμονες των υπολογιστών και της στατιστικής. Με την πάροδο του χρόνου, την εξέλιξη της τεχνολογίας, την αύξησης των δεδομένων αλλά και τους σημαντικότητας στην ανάλυση τους, το πρόβλημα αυτό άρχισε να αποκτά μεγαλύτερη βαρύτητα. Άρχισαν να προκύπτουν όλο και νέες τεχνικές και μέθοδοι στην ανίχνευση ανωμαλιών σε μεγάλα δεδομένα. Το πως προκύπτουν αυτές αλλά και το πως χειρίζονται μετά τον εντοπισμό τους, εξαρτάται ανάλογα με τον τομέα και το είδος της ακραίας τιμής. Το τι υποδηλώνει η κάθε μια από αυτές έχει ως αποτέλεσμα διαφορετική πληροφορία και αντιμετώπιση. Είτε θεωρούνται ως λάθος υπολογισμός ή απλά θόρυβος. Από την ανάλυση των συγκεκριμένων τιμών μπορεί να οδηγηθούμε σε πληροφορίες μεγάλης σημασίας, κάνοντας σημαντικό τον εντοπισμό πριν από την ανάλυση και επιλογή μοντέλου εκμάθησης.

Οι τιμές αυτές μπορούν να προέρχονται είτε από ανθρώπινο παράγοντα (λάθος, απάτη), είτε από σφάλμα αισθητήρων-μηχανών ή λόγω φυσικών αποκλίσεων στους [2] πληθυσμούς δεδομένων. Οπότε η εύρεση τους μπορεί να αποβεί πολύ σημαντική για την αποφυγή σφαλμάτων σε συστήματα, απάτες, ασφάλεια σε περιβάλλοντα καθώς μία ακραία τιμή μπορεί να υποδηλώνει βλάβη σε μία μηχανή, κακόβουλο εισβολέα σε ένα σύστημα. Αρκετά σημαντική είναι η αναγνώρισή τους επίσης στην ασφάλεια για πιστωτικές κάρτες όπου μία ακραία αλλαγή στο μοτίβο των αγορών μπορεί να υποδεικνύει μια κλεμμένη κάρτα.

ΕΦΑΡΜΟΓΕΣ ΤΗΣ ΕΥΡΕΣΗΣ ΑΝΩΜΑΛΙΩΝ ΣΕ ΠΡΟΒΛΗΜΑΤΑ ΤΗΣ ΚΑΘΗΜΕΡΙΝΟΤΗΤΑΣ

1. “ Απάτη σε πιστωτικές κάρτες (απόσπασμα από [1] σελίδες 3-4)
2. Ιατρικές χρήσεις π.χ. παρακολούθηση για τυχόν ακραίες τιμές σε καρδιογράφημα ή φαρμακευτικές με σκοπό την εύρεση νέων δομών μορίων.
3. Εσφαλμένη ομαδοποίηση δεδομένων που μπορεί να επιδράσει αρνητικά στην επίδοση του μοντέλου εκμάθησης.
4. Εντοπισμός απρόβλεπτων καταγραφών σε βάσεις δεδομένων (απάτη, σφάλματα).
5. Ανίχνευση σε δορυφορικές εικόνες για ταξινόμηση και εύρεση νέων χαρακτηριστικών των εικόνων.
6. Έξυπνες πόλεις, μέσω αισθητήρων οποιαδήποτε ακραία τιμή σε ρύπανση αέρα, θερμοκρασία, ηχορύπανση, κυκλοφοριακή συμφόρηση μπορεί να δώσει χρήσιμες πληροφορίες και να αποτρέψει ανεπιθύμητες καταστάσεις.
7. Εύρεση εισβολής σε υπολογιστικό σύστημα χωρίς άδεια.
8. Βλάβη σε μηχανικά συστήματα και κινητήρες.

9. Ανίχνευση σε παραγωγικές γραμμές για ελαττωματικά προϊόντα.

10. Ανίχνευση τάσης σε χρονοσειρές π.χ. χρηματιστήριο.

11. Έλεγχο αιτήσεων σε δάνεια για εντοπισμό παραπλάνησης ή πελατών που θα παρουσιάσουν προβλήματα στην εξόφληση τους μελλοντικά"

ΔΙΑΦΟΡΑ ΑΚΡΑΙΑΣ ΤΙΜΗΣ (OUTLIER) ΜΕ ΘΟΡΥΒΟ (NOISE)

Έχοντας αναφέρει τους λόγους που προκύπτει αυτό πρόβλημα, την σημασία επίλυσης του σε καθημερινά ζητήματα και την αξία της πληροφορία που μπορεί να προκύψει από αυτά, καλό θα ήταν να δοθεί ένας ορισμός για το τι εκφράζεται ως ανωμαλία στα δεδομένα (outlier) αλλά και να διευκρινίσουμε την διαφορά ανωμαλιών και θορύβου (noise vs outlier). Αν θα μπορούσαμε να το διατυπώσουμε [3]όσο πιο απλά γίνεται μία ***ανωμαλία είναι ένα εξαιρετικά απομακρυσμένο σημείο δεδομένων σε σχέση με το πιο κοντινό σημείο δεδομένων και τις υπόλοιπες γειτονικές τιμές σε ένα γράφημα ή ένα σύνολο στοιχείων το οποίο επεξεργαζόμαστε***. Ένα σημείο ή ένα σύνολο σημείων αρκετά χαμηλό ή υψηλό συγκριτικά με τα υπόλοιπα γειτονικά δεδομένα που ξεχωρίζει από αυτά. Σαν θόρυβο από την άλλη ***ορίζουμε τα λανθασμένα στοιχεία ή τις λάθος τιμές χαρακτηριστικών***. Η ακραία τιμή έχει μεγαλύτερο εύρος σαν έννοια εκτός από τα [4]σφάλματα στα δεδομένα και περιέχει ασυμβίβαστα δεδομένα που προκύπτουν μέσω της διακύμανσης του συνόλου, τα οποία όμως μπορεί να περιέχουν χρήσιμη πληροφορία σχετικά με το δείγμα**.**

Στην παρακάτω εικόνα παράδειγμα ακραίας τιμής στην φύση.

(παράδειγμα ανωμαλίας στην φύση, εικόνα από [3])

ΚΕΦΑΛΑΙΟ 2 Βιβλιογραφική Επισκόπηση

Η ανίχνευση ανωμαλιών που λόγω της τεχνολογικής ανάπτυξης, την αύξηση των δεδομένων αλλά και της συλλογής τους ώστε να αντληθούν χρήσιμα συμπεράσματα και πληροφορίες, έχει αποτελέσει ένα σημαντικό πεδίο της επιστημονικής έρευνας βρίσκοντας εφαρμογές σε πολλούς τομείς. Υπάρχουν πολλές έρευνες και επιστημονικά άρθρα σχετικά με διαφόρους τομείς των ακραίων τιμών και διαφορετικές προσεγγίσεις. Με την αύξηση του όγκου των δεδομένων και την πάροδο του χρόνου άρχισαν να προβάλουν νέες προκλήσεις αλλά και νέες προσεγγίσεις ώστε να αντιμετωπιστεί το συγκεκριμένο πρόβλημα.

Η πρώτη επιστήμη που ασχολήθηκε με αυτό το αντικείμενο ήταν η στατιστική. Χρησιμοποιήθηκαν στατιστικά μοντέλα που ανέλυαν τα στατιστικά χαρακτηριστικά των δεδομένων και των κατανομών ώστε να προβούν σε συμπεράσματα. Έπειτα μέσω της μηχανικής μάθησης χρησιμοποιώντας αλγορίθμους για την εκπαίδευση των μοντέλων ανίχνευσης ακραίων τιμών και τέλος βάση της μεθόδου βαθιάς μάθησης όπου νευρωνικά δίκτυα εκπαιδεύονταν στην ανίχνευση ανωμαλιών.

Στην παρούσα εργασία θα χρησιμοποιηθούν και οι τρεις προσεγγίσεις που αναφέρθηκαν στην προηγούμενη παράγραφο. Θα προσπαθήσουμε όμως να δώσουμε βαρύτητα ως προς την ανάλυση των συνόλων δεδομένων, των μεθόδων ανίχνευσης ανωμαλιών και των αλγορίθμων με αποτέλεσμα την επιλογή του βέλτιστου μοντέλου ανίχνευσης ανάλογα με τα χαρακτηριστικά της συγκεκριμένης ομάδας συνόλων δεδομένων. Έπειτα να θα αξιολογούμε τα μοντέλα ανίχνευσης ως προς την επίδοση τους στα δεδομένα.

Η μελέτη και η υλοποίηση του προγράμματος έγινε κατά κύριο λόγο βάση επιστημονικών άρθρων των τελευταίων ετών. Παρατηρήθηκε ότι ενώ υπήρχαν πολλές έρευνες πάνω σε εξελιγμένα μοντέλα αλγορίθμων και συνδυασμού αλγορίθμων. Δεν υπήρχαν αρκετές έρευνες που να δίνουν βαρύτητα στην ανάλυση των χαρακτηριστικών των δεδομένων με σκοπό την επιλογή της βέλτιστης μεθόδου και αλγορίθμου βάση αυτών. Η υλοποίηση του προγράμματος γίνεται μέσω της γλώσσας python, των αλγορίθμων (εκτός ενός βαθιάς μάθησης) αλλά και των συνόλων δεδομένων μέσω της βιβλιοθήκης PYOD. Δεύτερη παρατήρηση είναι δύσκολο να βρεθούν ‘labeled’ δεδομένα για ανίχνευση ανωμαλιών ώστε να γίνει αξιολόγηση της επίδοσης των μοντέλων. Τα σύνολα που θα χρησιμοποιήσουμε είναι από πραγματικά δεδομένα αλλά με προσθήκη τεχνητών ανωμαλιών.

Όλα όσα αναφέρθηκαν παραπάνω αντιπροσωπεύουν μόνο ένα μικρό μέρος των ερευνών που έχουν γίνει πάνω στο αντικείμενο. Η συνεχής ανάπτυξη μεθόδων, τεχνικών και τεχνολογιών σε αυτόν τον τομέα δείχνει την σημαντικότητα της έρευνας αλλά και την ανάγκη για την επίλυση αντίστοιχων προβλημάτων .

ΚΕΦΑΛΑΙΟ 3 ΕΙΔΗ ΑΝΩΜΑΛΙΩΝ ΚΑΙ ΜΕΘΟΔΟΙ ΕΝΤΟΠΙΣΜΟΥ ΣΕ ΣΤΑΤΙΚΑ ΔΕΔΟΜΕΝΑ

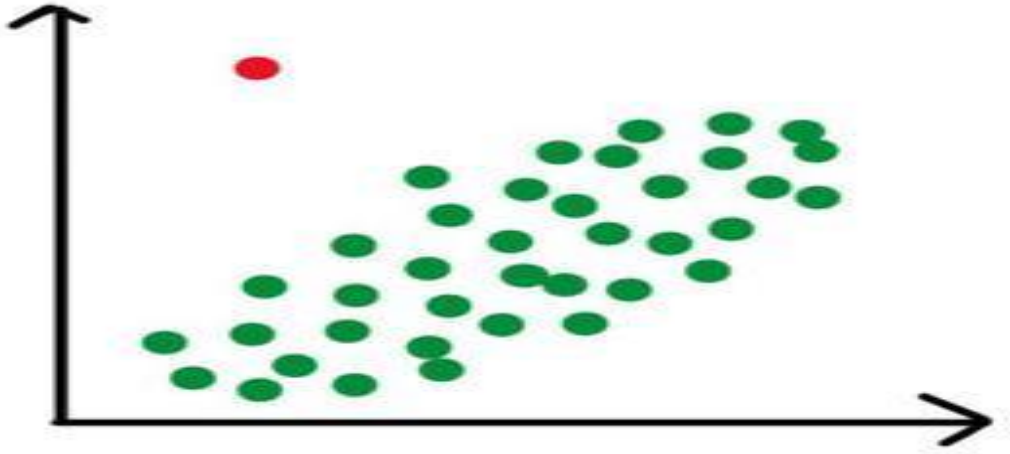
(Υποκεφάλαιο 3.1 ΕΙΔΗ ΑΝΩΜΑΛΙΩΝ ΣΕ ΣΤΑΤΙΚΑ ΔΕΔΟΜΕΝΑ)

Θέλοντας να αναλύσουμε μία ανώμαλη τιμή θα μπορούσαμε να την κατηγοριοποιήσουμε σε τέσσερις διαφορετικούς τύπους.

Καθολική ανωμαλία (global outlier)

Είναι σημεία που διαφέρουν αρκετά σε σχέση με το υπόλοιπο σύνολο των [5] δεδομένων. Οι λόγοι που προκύπτουν αφορούν σε λάθη σε μετρήσεις στην συγκέντρωση δεδομένων ή ασυνήθιστες τιμές στοιχείων σε σχέση με το σύνολο. Αυτό μπορεί να οδηγήσει σε εσφαλμένη ομαδοποίηση δεδομένων που μπορεί να [6] επιδράσει αρνητικά στην επίδοση του μοντέλου εκμάθησης. Για την αντιμετώπιση τους δηλαδή τον εντοπισμό και την επεξεργασία τους μπορούν να χρησιμοποιηθούν μοντέλα μηχανικής μάθησης, στατιστικά μοντέλα ή απλά με την εύρεση μέσω οπτικοποίησης των δεδομένων.

Καθολική ανωμαλία (global outlier) σε στατικά δεδομένα.

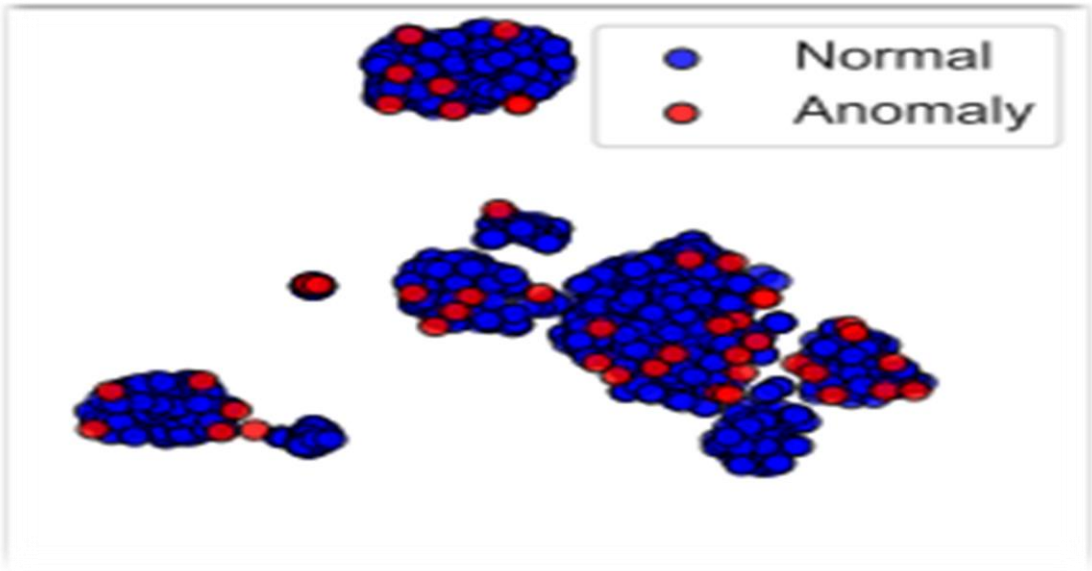


(παράδειγμα καθολικής ανωμαλίας στα στατικά δεδομένα, εικόνα από [5])

Τοπική ανωμαλία (local anomaly)

Είναι σημεία που δεν διαφέρουν αρκετά σε σχέση με το υπόλοιπο σύνολο των δεδομένων αλλά αν παρατηρηθούν με τα γειτονικά σημεία τότε μπορούν να θεωρηθούν ως ακραίες τιμές. Για την αντιμετώπιση τους δηλαδή τον εντοπισμό και την επεξεργασία τους μπορούν να χρησιμοποιηθούν μοντέλα μηχανικής μάθησης (που βασίζονται στον χωρικό εντοπισμό, δηλαδή μέσω της απόστασης distance-based, clustering (συσταδοποίηση), density-based (πυκνότητας).

Τοπική ανωμαλία (local anomaly) σε στατικά δεδομένα.

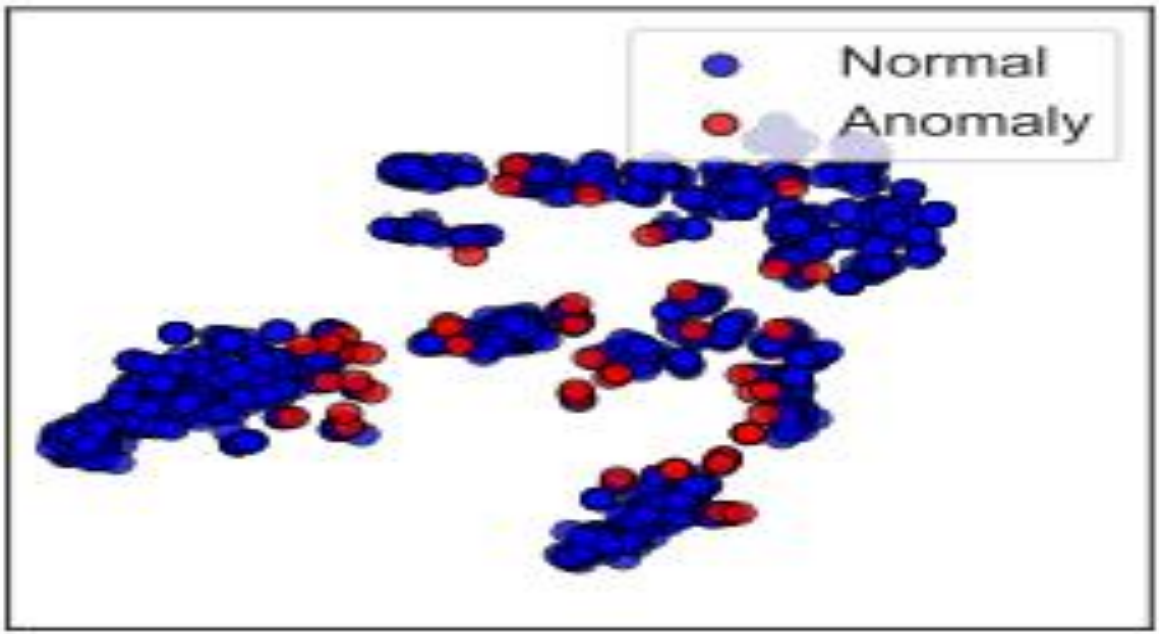


(παράδειγμα τοπική ανωμαλίας στα στατικά δεδομένα, εικόνα από [9])

Contextual-conditional outliers

Είναι σημεία που διαφέρουν από την αναμενόμενη συμπεριφορά σε μία υποομάδα, δηλαδή τα σημεία αυτά μπορεί να μην θεωρηθούν ακραία αν παρατηρηθούν σε διαφορετικά πλαίσια. Εάν εστιάσουμε την συζήτηση μας μόνο σε χρονοσειρές τότε το πλαίσιο αυτό μπορεί να θεωρηθεί ο χρόνος και οι contextual-conditional outliers βρίσκονται συχνά σε αυτές. Για τον εντοπισμό τους χρησιμοποιούνται ειδικά μοντέλα μηχανικής μάθησης για τις συγκεκριμένες τιμές, clustering (συσταδοποίηση) για, επίσης η γνώση πάνω στο αντικείμενο του συνόλου δεδομένων είναι αναγκαία για την εύρεση αναγνώριση τέτοιου είδους ακραίων τιμών.

(Contextual-conditional outliers) σε στατικά δεδομένα.

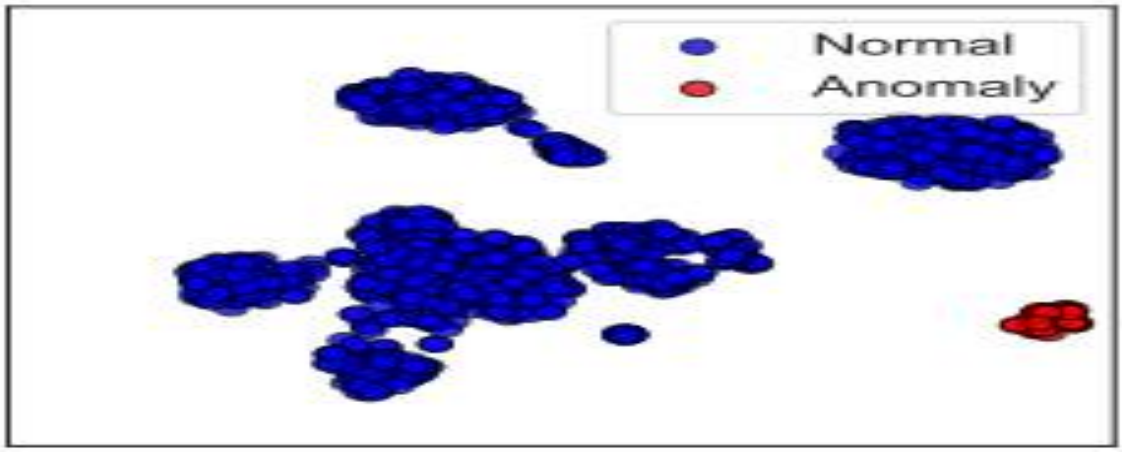


(παράδειγμα contextual outlier σε στατικα δεδομένα, εικόνα από [9])

Ανωμαλίες συστάδων (Collective Outliers)

Ομάδες δεδομένων που συλλογικά διαφέρουν από το υπόλοιπο σύνολο αλλά ατομικά μπορεί να μην θεωρούνταν ανωμαλίες. Για την εύρεση τέτοιων ανωμαλιών μπορούν να χρησιμοποιηθούν τεχνικές clustering (συσταδοποίηση), τεχνικές sub-spaced και density based. Ο εντοπισμός τους είναι πιο δύσκολο από αυτόν των μεμονωμένων ακραίων τιμών και η γνώση πάνω στο αντικείμενο του συνόλου δεδομένων (dataset) που εξετάζεται είναι πολύ σημαντική για τον αποτελεσματικό χειρισμό τους.

Ανωμαλίες συστάδων (Collective Outliers) σε στατικά δεδομένα.



(παράδειγμα ανωμαλίας συστάδων σε στατικά δεδομένα, εικόνα από [9])

Ας δώσουμε ένα παράδειγμα της καθημερινότητας για κάθε ένα τύπο ανωμαλίας ώστε να υπάρχει καλύτερη κατανόηση επί του θέματος. Ένας άνθρωπος που συνήθως κάνει αγορές διαδικτυακά με την κάρτα του από 100 με 200 ευρώ, μια μέρα κάνει αγορές 10000 ευρώ, επειδή αυτό δεν έχει ξανασυμβεί στο ιστορικό των συναλλαγών του μπορεί να θεωρηθεί σαν καθολική ανωμαλία (global outlier). Έστω ότι χιονίζει στην Ελλάδα, αυτό από μόνο του δεν θα μπορούσε να θεωρηθεί σαν ανωμαλία από την στιγμή που χιονίζει κάθε χρόνο στην Ελλάδα, αν όμως ο μήνας που χιόνιζε ήταν ο Αύγουστος τότε αυτό θα μπορούσε να θεωρηθεί από τα συμφραζόμενα σαν ακραία τιμή (Contextual-conditional outlier). Αν ένας φοιτητής λείπει από το μάθημα της σχολής μια μέρα δεν θα μπορούσαμε να το θεωρήσουμε ανωμαλία μιας και αυτό δεν είναι ένα εξαιρετικά ασυνήθιστο φαινόμενο, αν όμως οι μισοί φοιτητές λείπουν από το μάθημα την ίδια μέρα αυτό θα ήταν μία ανωμαλία συστάδων (Collective Outliers).

(Υποκεφάλαιο 3.2 ΠΡΟΣΕΓΓΙΣΕΙΣ ΜΕΘΟΔΩΝ ΑΝΙΧΝΕΥΣΗΣ ΑΝΩΜΑΛΙΩΝ)

Για την εύρεση ανωμαλιών υπάρχουν τρεις προσεγγίσεις όσον αφορά την μηχανική μάθηση (Machine learning) αλλά και την βαθιά μάθηση (Deep Learning). [1]

Προσέγγιση με επίβλεψη (supervised learning): Ο εντοπισμός των ακραίων τιμών[γίνεται με προηγούμενη γνώση από τα δεδομένα. Το μοντέλο έχει στο σύνολο εκπαίδευσης και κανονικά δεδομένα αλλά και ασυνεπή δεδομένα γνωρίζοντας σε ποια ομάδα ανήκει το κάθε δεδομένο και στα καινούρια δεδομένα που θα επεξεργαστεί θα κάνει προβλέψεις βάση των πληροφοριών από το προηγούμενο σύνολο.

Προσέγγιση με μη επίβλεψη (unsupervised learning): Ο εντοπισμός των ακραίων τιμών γίνεται χωρίς προηγούμενη γνώση από τα δεδομένα, εφαρμόζονται αλγόριθμοι σε δεδομένα χωρίς "ετικέτα" (non labelled). Το μοντέλο δεν έχει πληροφορία από το σύνολο εκπαίδευσης ώστε να διαχωρίσει ποια δεδομένα είναι ανωμαλίες και ποια όχι και ο εντοπισμός επιτυγχάνεται μέσω αλγορίθμων που βρίσκουν μοτίβα μέσα στα δεδομένα ώστε να κάνουν προβλέψεις. Στο τελικό πρόγραμμα θα χρησιμοποιηθεί κυρίως αυτή η προσέγγιση με τέτοιου είδους μοντέλα καθώς είναι πιο κοντά στα ρεαλιστικά πλαίσια, δηλαδή στα περισσότερα δεδομένα δεν υπάρχει γνώση εξ αρχής (labelled δεδομένα) έτσι για να τα αναλύσουμε και να γίνει ο εντοπισμός των ακραίων τιμών δεν μπορούν να χρησιμοποιηθούν μοντέλα supervised ή semi-supervised.

Προσέγγιση με ήμι-επίβλεψη (semi-supervised learning): Σε αυτήν την περίπτωση το μοντέλο θα έχει στο σύνολο εκπαίδευσης μόνο κανονικά δεδομένα ή σε ελάχιστες

περιπτώσεις μόνο ασυνεπή δεδομένα. Το μοντέλο εκπαιδεύεται σε κανονικά δεδομένα και μέσω αυτών μαθαίνει να κάνει τον διαχωρισμό ώστε να ανιχνεύει τις ακραίες τιμές.

Από τις τρεις προσεγγίσεις που αναφέρθηκαν θα ασχοληθούμε στην παρούσα εργασία με την την πρώτη κυρίως, δηλαδή προσέγγιση με μη επίβλεψη (unsupervised learning). Αυτή με την σειρά της μπορεί να χωριστεί σε διάφορες υποομάδες ή μεθόδους ως προς την ανίχνευση ακραίων τιμών που θα χρησιμοποιηθούν από το τελικό πρόγραμμα. Σε κάθε μία υποομάδα μπορούν να κατηγοριοποιηθούν οι διάφοροι αλγόριθμοι που θα αναφερθούν παρακάτω στην εργασία. Η κάθε μέθοδος έχει διαφορετική προσέγγιση ως προς την ανίχνευση των ακραίων τιμών και η επιλογή της έχει να κάνει με τα χαρακτηριστικά των δεδομένων. Τα δεδομένα τα οποία θα αναλύσουμε είναι κατά κύριο λόγο στατικά παρόλα αυτά θα γίνουν και αναφορές σε δεδομένα μη στατικά, δηλαδή χρονικά μεταβαλλόμενα.

(Υποκεφάλαιο 3.3 ΜΕΘΟΔΟΙ ΕΥΡΕΣΗΣ ΑΝΩΜΑΛΙΩΝ)

Μέθοδος με βάση το βάθος (depth-based) : Η ανίχνευση για τις ανωμαλίες γίνεται[7] [8] στα όρια του χώρου δεδομένων, ξεκινάει από το "βαθύτερο" σημείο, σαν το κέντρο του χώρου δεδομένων που είναι τα περισσότερα κανονικά δεδομένα και αναζητάει ακραίες τιμές προς τα έξω στα όρια του χώρου. Δεν χρειάζεται να ξέρουμε σε ποια κατανομή ανήκει το σύνολο των δεδομένων και ούτε να χρησιμοποιήσουμε χωρικές αποστάσεις μεταξύ των δεδομένων για να βρούμε τις ακραίες τιμές.

Μέθοδος με βάση την απόσταση (distance-based): Η ανίχνευση για τις ανωμαλίες γίνεται βάση της απόστασης των γειτονικών σημείων σε σχέση με το σημείο που εξετάζουμε για ενδεχόμενη ακραία τιμή, πιο σωστά ενός αριθμού γειτονικών σημείων που συνήθως το αναφέρεται ως κ-γειτόνων. Είναι δύσκολο σε αυτήν τη μέθοδο να βρούμε τον βέλτιστο αριθμό "κ" ώστε να έχει μεγαλύτερη αποτελεσματικότητα ο distance-based αλγόριθμος που θα χρησιμοποιήσουμε. Αυτό συνήθως χρειάζεται αρκετές δοκιμές. Εάν το σύνολο δεδομένων έχει διαφορετικές πυκνότητες στον χώρο των δεδομένων τότε τέτοιου είδους αλγόριθμοι δυσκολεύονται να εντοπίσουν τις τοπικές ακραίες τιμές (local anomaly).

Μέθοδος με βάση την πυκνότητα (density-based approaches): Η ανίχνευση για τις ανωμαλίες γίνεται βάση της πυκνότητας σε κάποιες χωρικές περιοχές των δεδομένων, οι ακραίες τιμές βρίσκονται κυρίως στις περιοχές με μικρή πυκνότητα και εκεί είναι που γίνεται η αναζήτηση τέτοιων τιμών από τους διάφορους αλγορίθμους. Σε αντίθεση με distance-based τεχνικές μπορούν να εντοπίσουν τις ανωμαλίες που βρίσκονται στο χώρο δεδομένων διαφορετικών πυκνοτήτων, κατά κύριο λόγο τοπικές ακραίες τιμές (local anomaly). Υπάρχει δυσκολία στην εύρεση του βέλτιστου κατωφλιού για το πάνω και κάτω όριο ώστε να επιτύχουμε την μέγιστη απόδοση των αλγορίθμων που ακολουθούν την συγκεκριμένη προσέγγιση.

Μέθοδος με βάση την συσταδοποίηση (cluster-based Approaches): Η προσέγγιση αυτή ομαδοποιεί σύνολα δεδομένων που συσχετίζονται ή είναι όμοια με κάποιον τρόπο σε διάφορες συστάδες και όσες δεδομένα δεν ανήκουν σε κάποια συστάδα θεωρούνται ακραίες τιμές. Αν κάποιο δεδομένο βρίσκεται μακριά χωρικά από κάποια συστάδα τότε θεωρείται ακραία τιμή, αν μία συστάδα που έχει δημιουργηθεί έχει μικρό αριθμό δεδομένων ή η συστάδα των δεδομένων είναι διάσπαρτη ή αλλιώς "αραιή" χωρικά τότε η συγκεκριμένη ομάδα μπορεί να θεωρηθεί σαν σύμπλεγμα ακραίων τιμών. Η συγκεκριμένη προσέγγιση είναι ένα χαρακτηριστικό είδος προσέγγισης με μη επίβλεψη (unsupervised learning), δεν χρειάζονται δεδομένα με "ετικέτες" (labeled data) γνώση των δεδομένων στο σύνολο εκπαίδευσης για να μπορεί να λειτουργήσει ο αλγόριθμος και να δημιουργήσει τις συστάδες δηλαδή υποομάδες του συνόλου δεδομένων και να συγκρίνει τα υπόλοιπα δεδομένα με τις ομάδες ώστε να διαχωρίσει αν αυτά είναι ακραίες τιμές. Τέτοιο είδους μέθοδοι είναι κοστοβόρες υπολογιστικά και χρονικά και για να έχει μεγαλύτερη ακρίβεια ο αλγόριθμος πρέπει να επιλεχθεί η κατάλληλη μέθοδος συσταδοποίησης, για να γίνει αυτό πρέπει να αναλύσουμε τα δεδομένα επειδή δεν είναι εφικτό να γνωρίζουμε την κατάλληλη μέθοδο εξ αρχής χωρίς να έχουμε πληροφορίες από το σύνολο των δεδομένων που επεξεργαζόμαστε.

Στατιστικές μέθοδοι (statistical): Ο διαχωρισμός ανάμεσα σε ανωμαλία και κανονική τιμή στα δεδομένα επιτυγχάνεται βάση ενός στατιστικού μοντέλου. Εαν μια εγγραφή ακολουθεί το μοντέλο θεωρείται κανονική αλλιώς ακραία τιμή. Γίνεται χρήση κάποιων γνωστών στατιστικών μοντέλων ώστε να κατατάξουμε τα δεδομένα μας σε κάποια κατανομή και έπειτα να κάνουμε τον διαχωρισμό για το ποια ακολουθούν και ποια όχι την κατανομή. Η δυσκολία στην εφαρμογή της μεθόδου είναι ότι απευθύνεται σε σύνολα δεδομένων με ένα έως ελάχιστα χαρακτηριστικά. Όταν ένα σύνολο διαθέτει πολλά χαρακτηριστικά είναι εξαιρετικά δύσκολο να βρούμε τι κατανομή ακολουθεί και να μπορούμε να κάνουμε τον έλεγχο για τον διαχωρισμό. Όσο πιο πολλά χαρακτηριστικά υπάρχουν σε ένα σύνολο τόσο μεγαλύτερες πιθανότητες υπάρχουν κάποια χαρακτηριστικά να είναι συσχετισμένα (correlated) είτε θετικά δηλαδή όταν αυξάνεται η τιμή ενός χαρακτηριστικού αυξάνεται και αυτή του συσχετιζόμενου είτε αρνητικά δηλαδή όταν αυξάνεται ενός χαρακτηριστικού να μικραίνει η τιμή του άλλου συσχετιζόμενου και το αντίστροφο. Οποιοδήποτε και από τα δύο συμβαίνει καθιστά προβληματική την ανάλυση λόγο του φαινομένου της αλληλοσυσχέτισης (intercorrelation). Τέλος το να βρεθεί σύνολο δεδομένων που να ακολουθεί κανονική κατανομή είναι εξαιρετικά σπάνιο σε πραγματικά δεδομένα.

Μέθοδοι με βάση την γραφική αναπαράσταση (Graph-Based): Μέσω αυτής της [10]αναπαράστασης μετατρέποντας τα σημεία σε κόμβους του γράφου μπορεί να βρεθούν συσχετίσεις μεταξύ των δεδομένων και κάποια αναμενόμενα μοτίβα. Έπειτα θα μπορεί να γίνει ο διαχωρισμός ανάμεσα σε κανονικά σημεία ή μοτίβα μέσα στον γράφο και σε μη κανονικά ή αναμενόμενα. Υπάρχουν δύο γνωστές τεχνικές για ανίχνευση ανωμαλιών με γραφική αναπαράσταση. Η πρώτη είναι να γίνει η εύρεση των ανωμαλιών ψάχνοντας για συγκεκριμένα ασυνήθιστα υπογραφήματα στο γράφο. Η δεύτερη είναι η συλλογή διαφόρων υπογράφων και η σύγκριση μεταξύ τους για την εύρεση μη συνηθισμένων μοτίβων. Ένας χαρακτηριστικός αλγόριθμος με βάση την γραφική αναπαράσταση και αρκετά δημοφιλής είναι το (isolation forest) το οποίο θα χρησιμοποιηθεί στο πρόγραμμα μας και θα αναλυθεί παρακάτω στην εργασία.

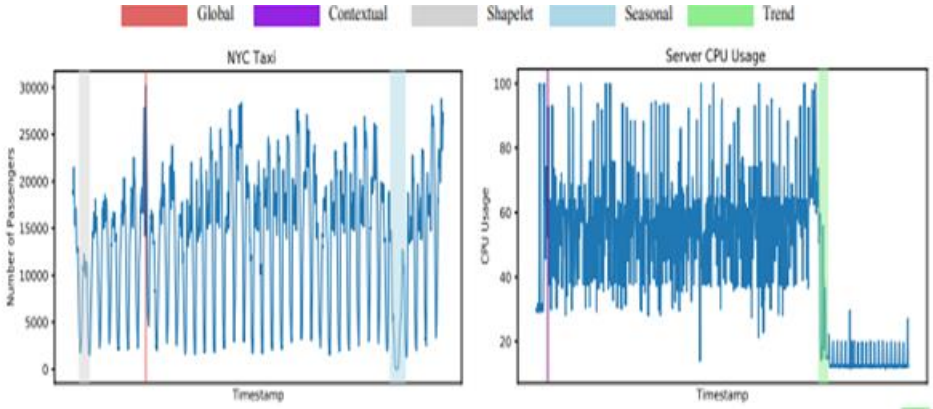
Μέθοδοι με βάση την μοντελοποίηση (Learning-based Models): Μαθαίνει τους [17] αλγόριθμους να ανιχνεύουν τις ακραίες τιμές από ένα σύνολο εκπαίδευσης όπου είναι γνωστές οι κανονικές τιμές και μέσω αυτής της πληροφορίας εκπαιδεύει τον αλγόριθμο να διαχωρίζει τις κανονικές από τις ακραίες τιμές και να τις ανιχνεύει. Τέτοιου είδους μέθοδοι είναι καλές σε μεγάλα δεδομένα αλλά λόγο των πολλών πράξεων και της βελτιστοποίησης των παραμέτρων τους για να είναι αποτελεσματικές μπορεί να αποβούν χρονοβόρες και κοστοβόρες υπολογιστικά.

ΚΕΦΑΛΑΙΟ 4 ΧΡΟΝΙΚΑ ΜΕΤΑΒΑΛΛΟΜΕΝΑ ΔΕΔΟΜΕΝΑ

Έχοντας μιλήσει για τις ακραίες τιμές και τις μεθόδους αντιμετώπισης τους στα στατικά δεδομένα θα παρουσιάσουμε αντίστοιχα στα χρονικά μεταβαλλόμενα. Εφόσον αντιμετωπίζουμε ξεχωριστά σύνολα είναι προφανές πως υπάρχουν διαφορές μεταξύ τους (π.χ. μπαίνει ο χρόνος σαν παράγοντας) οπότε ακόμα και αν αναφερόμαστε σε ίδιες ορολογίες ή ονομασίες η προσέγγιση και η αντιμετώπιση θα πρέπει να διαφέρει ώστε να έχουμε καλύτερη κατανόηση και αποτελεσματικότητα. Είναι σημαντικό να εξετάσουμε τους ορισμούς όχι σαν επέκταση από την προϋπάρχουσα γνώση των στατικών δεδομένων, δηλαδή διαχωρισμό μεταξύ ομοιότητας σημείων αλλά σαν την γενική δομή διαδοχικών δεδομένων στον χρόνο με αποτέλεσμα να μπαίνει και ο χρόνος σαν παράγοντας στην εξίσωση.

Ένα παράδειγμα για να κατανοήσουμε την σημασία όσον αναφέραμε στην [23] προηγούμενη παράγραφο φαίνεται στην παρακάτω εικόνα. Παρουσιάζει πως μια συλλογικά ακραία τιμή μπορεί να έχει εντελώς διαφορετική συμπεριφορά και να μην

προκύπτει επαρκής και ουσιαστική πληροφορία με τον προϋπάρχων ορισμό. Οπότε χρειάζεται να αναλυθεί και να κατηγοριοποιηθεί η συλλογικά ακραία τιμή ως προς την περιοδικότητα, την τάση και το σχήμα.

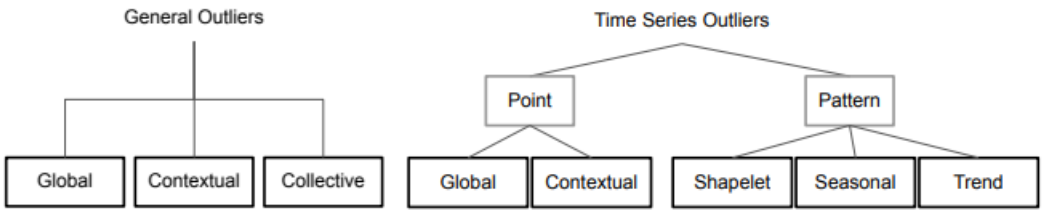


(είδη ανωμαλιών σε χρονοσειρές, εικόνα από [23])

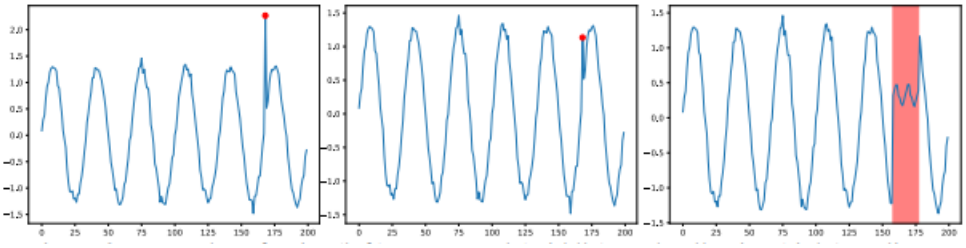
(ΥΠΟΚΕΦΑΛΑΙΟ 4.1 ΟΡΙΣΜΟΙ ΤΥΠΩΝ ΚΑΙ ΕΙΔΗ ΑΚΡΑΙΩΝ ΤΙΜΩΝ ΣΕ ΧΡΟΝΙΚΑ ΜΕΤΑΒΑΛΛΟΜΕΝΑ ΔΕΔΟΜΕΝΑ)

Μπορεί τα είδη των ακραίων τιμών να έχουν την ίδια ονομασία στα στατικά και χρονικά μεταβαλλόμενα δεδομένα όμως λόγω του χρόνου που είναι ένας παράγοντας που αλλάζει την σχέση των δεδομένων μεταξύ τους κάνοντας τα στοιχεία ακολουθιακά πρέπει να παρουσιάσουμε μια διαφορετική προσέγγιση στον ορισμό των συγκεκριμένων τιμών.

Αρχικά, θα μπορούσαμε να χωρίσουμε σε δύο αρχικές ομάδες ακραίων τιμών,[23] σε ακραίες τιμές ξεχωριστών σημείων "point " και σε ακραία μοτίβα τιμών "pattern ". Οι ακραίες τιμές ξεχωριστών σημείων χωρίζονται έπειτα σε "global outlier" και σε "contextual outlier". Η δεύτερη ομάδα των μοτίβων θα εξεταστεί ως προς το σχήμα, την εποχικότητα και την τάση. Με αυτόν τον τρόπο σε αντίθεση με την προϋπάρχουσα ορολογία καταφέρνουν να αναλύσουμε τις ανωμαλίες σε μια χρονοσειρά ως προς την συμπεριφορά των ακολουθιακών δεδομένων καταφέρνοντας να περιγράψουμε τις τιμές αυτές πιο ουσιαστικά αποσπώντας περισσότερη πληροφορία.

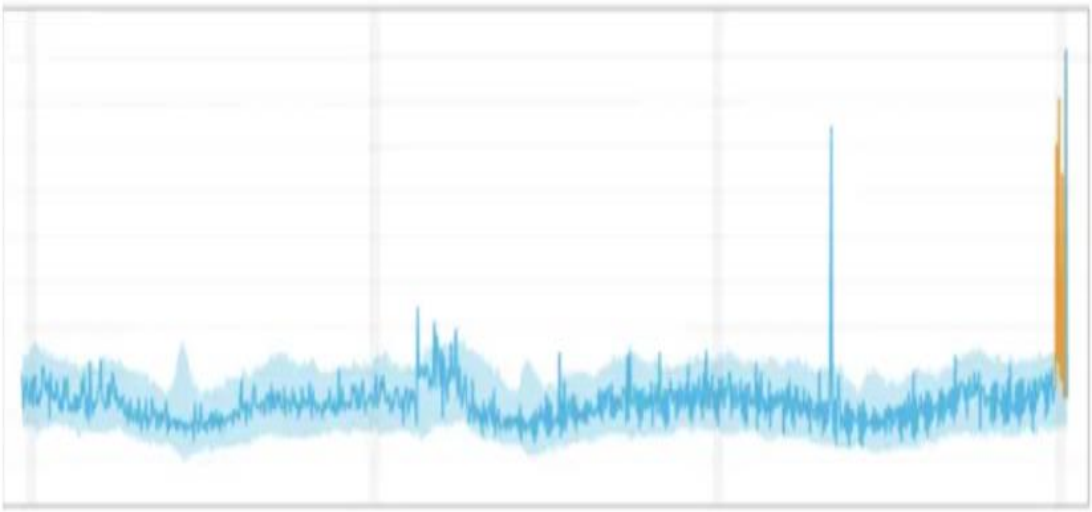


(αριστερά διαχωρισμός ειδών ακραίας τιμής βάση υπάρχουσας μεθοδολογίας , δεξιά διαχωρισμός ειδών ακραίας τιμής βάση της συμπεριφοράς, εικόνα από [23]))



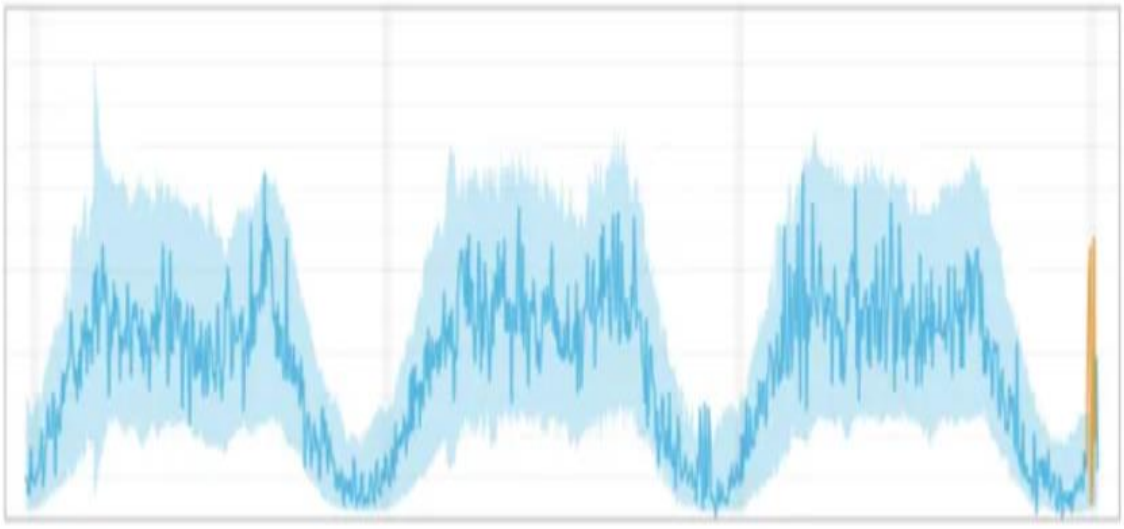
(από αριστερά προς τα δεξιά παράδειγμα καθολικής ακραίας τιμής ,contextual και ανωμαλίας συστάδων, εικόνα από [23])

Global (καθολική ανωμαλία): Είναι το μόνο είδος ακραίας τιμής που θα μπορούσε να έχει ακριβώς τον ίδιο ορισμό και στα χρονικά μεταβαλλόμενα και στα στατικά δεδομένα γιατί δεν επηρεάζεται η σημασία του από το χρόνο. Όπως και στα στατικά είναι ένα στοιχείο που διαφέρει αρκετά σε σχέση με τα υπόλοιπα σημεία στο σύνολο, οπότε σε οποιοδήποτε σημείο και αν βρισκόταν χρονικά θα το θεωρούσαμε ακραία τιμή. Η ανίχνευση τέτοιους είδους τιμών είναι πολύ σημαντική στα συγκεκριμένα δεδομένα διότι μπορούν να επηρεάσουν πολύ αρνητικά την απόδοση του μοντέλου και να προβούμε σε εσφαλμένα συμπεράσματα κάνοντας λάθος προβλέψεις από λάθος πληροφορίες.



(Καθολική ανωμαλία (global outlier) σε χρονικά μεταβαλλόμενα δεδομένα(time series, εικόνα από [6])

Contextual : Οι συγκεκριμένες ακραίες τιμές είναι δυσκολότερο να ανιχνευτούν συγκριτικά με τις global καθώς μπορεί να μην φαίνονται εντελώς διαφορετικές σχετικά με το υπόλοιπο σύνολο αλλά σε συγκεκριμένα χρονικά πεδία να διαφέρουν.



((Contextual anomaly) σε χρονικά μεταβαλλόμενα δεδομένα (timeseries) , εικόνα από [6])

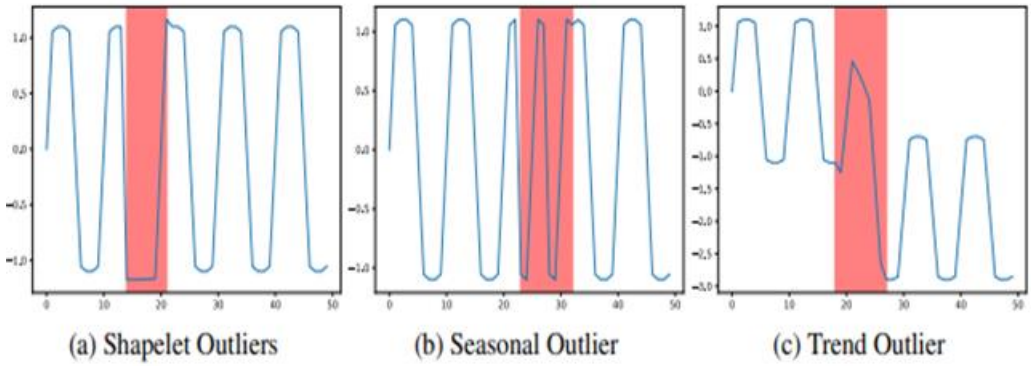
pattern outliers (collective outliers) : Είναι μία ορολογία που υπήρχε και στα στατικά δεδομένα (**collective outliers**) αλλά διαφέρει η σημασία της στις χρονοσειρές. Έχει να κάνει με μια συλλογή δεδομένων που ξεχωριστά το κάθε σημείο μπορεί να μην ήταν ακραία τιμή αλλά συλλογικά να διαφέρουν λόγο των χρονικών εξαρτήσεων μεταξύ των υπόλοιπων σημείων. Η συγκεκριμένη ομάδα θα πρέπει να αναλυθεί γιατί μπορεί να ένα σύνολο σημείων να θεωρείται ανωμαλία ως προς το σχήμα, την εποχικότητα και την τάση.

Shapelet (βάση του σχήματος): Ως προς το σχήμα οι συλλογικά ακραίες τιμές, θεωρούνται ομάδες ακολουθιακών στοιχείων που ακολουθούν ένα μοτίβο το οποίο διαφέρει σχετικά με υπόλοιπα στο σύνολο δεδομένων χρονικά και δεν είναι το συνηθισμένο ή το προβλεπόμενο που θα περιμέναμε. Η ανίχνευσή τους προκύπτει συγκρίνοντας υποσύνολα δεδομένων μεταξύ τους ώστε να βρούμε πόσο διαφορετικά είναι από το συνηθισμένο. Η σύγκριση αυτή γίνεται μέσω μιας συνάρτησης και ενός ορίου που δείχνει πόσο ασυνήθιστο είναι το υποσύνολο και διαφέρει σε σχέση με το προβλεπόμενο.

Seasonal (βάση της εποχικότητας): Πρόκειται για υποσύνολα τιμών που δεν διαφέρουν ως προς το σχήμα αλλά βάσης της εποχικότητας που έχει η συνολική χρονοσειρά και ανιχνεύονται μόνο βάση αυτού του κριτηρίου.

Trend (βάση της τάσης): Η συγκεκριμένη ομάδα ακραίων τιμών αποτελείται από υποσύνολα στοιχείων που βρίσκονται στα σημεία καμπής της χρονοσειράς, δηλαδή σε σημεία που αλλάζει απότομα η τάση της χρονοσειράς. Με ένα παράδειγμα για να γίνει πιο κατανοητό, έστω ότι ο μέσος όρος των σημείων σε μια χρονοσειρά μέχρι την συγκεκριμένη χρονική περίοδο είναι 2, έπειτα από το υποσύνολο που αναφέρουμε υπάρχει αλλαγή στην χρονοσειρά και από εκεί και μετά αλλάζει ο μέσος όρος των σημείων δηλαδή είτε γίνεται μεγαλύτερος είτε μικρότερο ανάλογα με την τάση, δηλαδή η χρονοσειρά έχει είτε θετική είτε αρνητική τάση από εκείνο το σημείο.

Η παρακάτω φωτογραφία θα μας βοηθήσει να κατανοήσουμε καλύτερα τις παραπάνω έννοιες



(είδη pattern outliers, εικόνα από [23])

(ΥΠΟΚΕΦΑΛΑΙΟ 4.2 ΚΑΤΗΓΟΡΙΕΣ ΑΛΓΟΡΙΘΜΩΝ ΑΝΙΧΝΕΥΣΗΣ ΑΚΡΑΙΩΝ ΤΙΜΩΝ ΣΕ ΧΡΟΝΙΚΑ ΜΕΤΑΒΑΛΛΟΜΕΝΑ ΔΕΔΟΜΕΝΑ)

Ως προς την ανίχνευση των ακραίων τιμών και την κατανόηση συμπεριφοράς των ακολουθιακών στοιχείων στις χρονοσειρές, διότι όπως έχει αναφερθεί σε προηγούμενες ενότητες οι ακραίες τιμές εκτός από το να δυσχεραίνουν την επίδοση του μοντέλου μπορεί να δίνουν και σημαντικές πληροφορίες ως προς την συμπεριφορά των δεδομένων, θα μπορούσαμε να χωρίσουμε την λειτουργία των αλγορίθμων σε τρεις κατηγορίες. Η πρώτη βάση της διαφοράς των αναμενόμενων τιμών από τις πραγματικές που προκύπτουν από την [23]συνολική συμπεριφορά των στοιχείων της χρονοσειράς. Η δεύτερη βάση της πλειοψηφίας, πόσο δηλαδή διαφέρουν οι τιμές που ελέγχονται βάση της πλειοψηφίας των δεδομένων και τέλος βάση της διαφοράς των υποσυνόλων.

Απόφαση βάση διαφοράς προς την αναμενόμενη τιμή: Οι αλγόριθμοι αυτής της ομάδας διαχωρίζουν τις ανώμαλες από τις κανονικές παρατηρήσεις ανάλογα από το πόσο διαφέρουν τα δεδομένα του συνόλου από τις αντίστοιχες προβλεπόμενες τιμές

τις συγκεκριμένες χρονικές περιόδους. Οι προβλεπόμενες τιμές προκύπτουν από μαθηματικά μοντέλα, είτε μέσω αναπαράστασης των δεδομένων. Ένας αλγόριθμος που βασίζεται σε αυτή την μέθοδο (αναπαράστασης) και θα χρησιμοποιηθεί από το πρόγραμμα μας είναι ο LSTM-autoencoder και η λειτουργία του θα αναλυθεί παρακάτω στην εργασία.

Απόφαση βάση της πλειοψηφίας: Η συγκεκριμένη μέθοδος θεωρεί πως τα στοιχεία μπορούν να αναπαρασταθούν χωρικά ώστε να δημιουργείται κάτι σαν μία συστάδα που θα περιέχεται η πλειοψηφία του συνόλου δεδομένου και από αυτή την ομάδα θα μπορούν με κάποιο όριο να χωριστούν με τα ανώμαλα δεδομένα δηλαδή τα στοιχεία που θα συγκεντρώνονται πέρα από τα όρια της πλειοψηφίας των δεδομένων δηλαδή των κανονικών. Δύο χαρακτηριστικοί αλγόριθμοι αυτής της ομάδας που έχουν παρόμοια λειτουργία με αυτήν που περιγράψαμε μόλις (ειδικά ο πρώτος) είναι ο OCSVM και ο IFOREST και θα χρησιμοποιηθούν και αυτοί στο πρόγραμμα μας και θα αναλυθούν παρακάτω στην εργασία.

Απόφαση βάση της διαφοράς των υποσυνόλων: Αυτή η μέθοδος χωρίζει την χρονοσειρά σε υποσύνολα και τα συγκρίνει μεταξύ του προσπαθώντας να βρει τα πιο διαφορετικά σε σχέση με τα υπόλοιπα υποσύνολα. Βρίσκει κυρίως pattern outliers όπως τους αναφέραμε προηγουμένως στην ενότητα. Η σύγκριση μπορεί να γίνει με αλγόριθμους συσταδοποίησης και εμείς στο πρόγραμμά μας θα χρησιμοποιήσουμε έναν τέτοιο το cblof αλλά και με τους αλγορίθμους της παραπάνω ομάδας που αναφέραμε iforest, ocsvm.

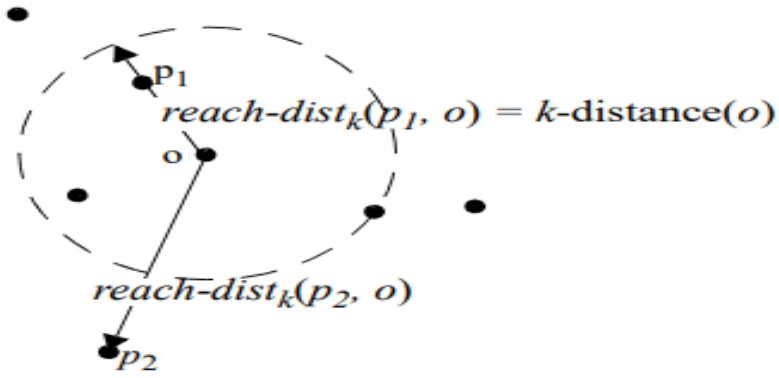
ΚΕΦΑΛΑΙΟ 5 ΑΝΑΛΥΣΗ ΑΛΓΟΡΙΘΜΩΝ ΓΙΑ ΕΥΡΕΣΗ ΑΝΩΜΑΛΙΩΝ

Στην παρούσα εργασία θα χρησιμοποιηθούν αλγόριθμοι για την εύρεση ανωμαλιών σε στατικά δεδομένα, η υλοποίηση και η ανάλυση τους θα γίνει μέσω της βιβλιοθήκης **PyOD,** μία από τις πιο εξειδικευμένες βιβλιοθήκες python για ανίχνευση ανωμαλιών σε στατικά δεδομένα πολλών χαρακτηριστικών. Όλοι οι αλγόριθμοι που θα υλοποιηθούν θα είναι με μη επίβλεψη (unsupervised) όπως και ο αλγόριθμος βαθιάς μάθησης (deep learning) LSTM autoencoder που θα χρησιμοποιηθεί για ανίχνευση ανωμαλιών σε χρονικά μεταβαλλόμενα δεδομένα.

(Υποκεφάλαιο 5.1 Local Outlier Factor (LOF))

Local Outlier Factor (LOF)

Στους περισσότερους αλγόριθμους ανίχνευσης ανωμαλιών [11] οι ανωμαλίες διαχωρίζονται δυαδικά, δηλαδή είναι ανώμαλη τιμή ή όχι και όχι σε ποσοστό, δηλαδή πόσο πολύ θεωρείται ακραία μια τιμή. Ο συγκεκριμένος αλγόριθμος δεν κάνει αυτό το διαχωρισμό μόνο αλλά δίνει και έναν βαθμό για το πόσο θεωρείται ένα σημείο ακραία τιμή, αυτός ο βαθμός είναι ο **Local Outlier Factor (LOF)** του σημείου και δείχνει κατά πόσο είναι απομακρυσμένο από τα υπόλοιπα κοντινά στοιχεία. Για να κατανοήσουμε πως υπολογίζεται ο **Local Outlier Factor** είναι σημαντικό να εξηγήσουμε κάποιους ορισμούς. Σαν (**k-distance**) ορίζουμε την απόσταση του σημείου που ελέγχουμε με το K πιο κοντινό, δηλαδή αν το (**k-distance**) k=n θα βρίσκαμε την απόσταση του n πιο κοντινού σημείου από το στοιχείο που εξετάζουμε και το (**k-distance**) και προκύπτει από τον μέσο όρο των αποστάσεων των σημείων. Θα είναι μία ακτίνα γύρο από το σημείο και προφανώς μικρότερη από την απόσταση του n σημείου. Έπειτα η **k-distance neighborhood** ορίζεται από τα σημεία που έχουν μικρότερη ή ίση απόσταση από το σημείο εξέτασης σε σχέση με την απόσταση **k-distance** και το **reachability distance** που μετράει την προσβασιμότητα των σημείων σαν απόσταση δεν θα έδινε την πραγματική απόσταση αλλά θα την εξομάλυνε σαν την **k-distance** των σημείων της γειτονιάς και όποιο σημείο είχε μεγαλύτερη απόσταση από αυτή τότε το **reachability distance** θα έδινε την πραγματική του απόσταση από το σημείο όπως θα φανεί καλύτερα στην παρακάτω εικόνα.



(παράδειγμα υπολογισμού της reach distance, εικόνα από [11])

Είναι εύκολο να καταλάβουμε ότι όσο μεγαλύτερο είναι το k τόσο περισσότερα στοιχεία θα έχουν το ίδιο **reachability distance,** άρα το να βρούμε το βέλτιστο k είναι αρκετές φορές δύσκολο και χρειάζεται αρκετές δοκιμές. Σε συνδυασμό με την υψηλή πολυπλοκότητα του αλγορίθμου που είναι $O(n^2)$ τον κάνει αρκετά κοστοβόρο στην εφαρμογή του. Σε αλγόριθμους συσταδοποίησης με βάση την πυκνότητα υπάρχουν δύο παράμετροι που δείχνουν την πυκνότητα μια περιοχής σημείων. Ο **MinPts** που είναι ο ελάχιστος αριθμός γειτονικών στοιχείων που χρειάζεται για να θεωρηθούν οι γειτονιές και το **volume** που είναι για το πόσο χώρο να καλύπτει η κάθε γειτονιά. Μέσω αυτών μπορούμε να υπολογίσουμε το όριο για την πυκνότητα, δηλαδή πόσο κοντά πρέπει να είναι τα σημεία σε ένα συγκεκριμένο χώρο ώστε να θεωρούνται ‘γειτονιά’. Στον συγκεκριμένο αλγόριθμο δεν μπορούμε να έχουμε ένα συγκεκριμένο όριο πυκνότητας για όλες ‘γειτονιές’ οπότε με το **MinPts** δίνουμε έναν αριθμό για το πόσα σημεία

χρειάζονται για να θεωρηθεί μία ΄γειτονιά' και το κατάλληλο **volume** βρίσκεται χρησιμοποιώντας το **reachability distance των MinPts** κάθε περιοχής ώστε να βρούμε πόσο 'πυκνή' είναι. Εαν αυτό είναι μεγάλο τότε η περιοχή δεν είναι τόσο πυκνή και εκεί πιθανόν θα υπάρχουν ακραίες τιμές. Το **(local reachability density)** είναι το αντίστροφο κλάσμα του αθροίσματος των **reachability distance των MinPts** προς τον αριθμό των **MinPts,** δηλαδή το αντίστροφο του μέσου όρου των **reachability distance των MinPts.**

1.

$$lrd_{MinPts}(p) = 1 / \left(\frac{\sum_{o \in N_{MinPts}(p)} reach\text{-}dist_{MinPts}(p, o)}{|N_{MinPts}(p)|} \right)$$

(τύπος local reachability distance, εικόνα από [11])

Έχοντας κατανοήσει όλους τους προηγούμενους όρους μπορούμε να υπολογίσουμε το LOF για το σημείο που εξετάζουμε. Το οποίο είναι ο μέσος όρος του λόγου του lrd (που εξηγήσαμε προηγουμένως) του σημείου που εξετάζουμε με το lrd των υπόλοιπων **MinPts** της ΄γειτονιάς΄. Συγκρίνουμε δηλαδή το το lrd του σημείου που εξετάζουμε με τα γειτονικά lrd των σημείων. Όσο πιο πολύ διαφέρει το lrd του σημείου με τα γειτονικά τόσο μεγαλύτερο το LOF του σημείου θα είναι και τόσο πιο πιθανό το σημείο αυτό να είναι ακραία τιμή Οι φυσιολογικές τιμές LOF σε ένα σημείο πρέπει να είναι κοντά στην μονάδα.

2.

$$LOF_{MinPts}(p) = \frac{\sum_{o \in N_{MinPts}(p)} \frac{lrd_{MinPts}(o)}{lrd_{MinPts}(p)}}{|N_{MinPts}(p)|}$$

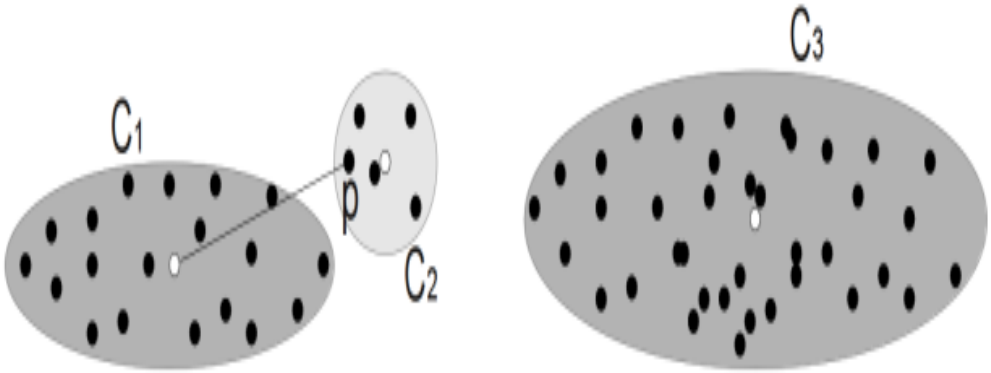
(τελικός τύπος local outlier factor, εικόνα από [11])

Από το όνομα του και μόνο αλλά και από τον τρόπο λειτουργίας δηλαδή ο έλενχος γίνεται μόνο με γειτονικά ή αλλιώς τοπικά (local) στοιχεία καταλαβαίνουμε ότι αυτός ο αλγόριθμος είναι ισχυρός στην ανίχνευση τοπικών ακραίων τιμών (local anomaly).

(Υποκεφάλαιο 5.2 Cluster-Based Local Outlier Factor (CBLOF))

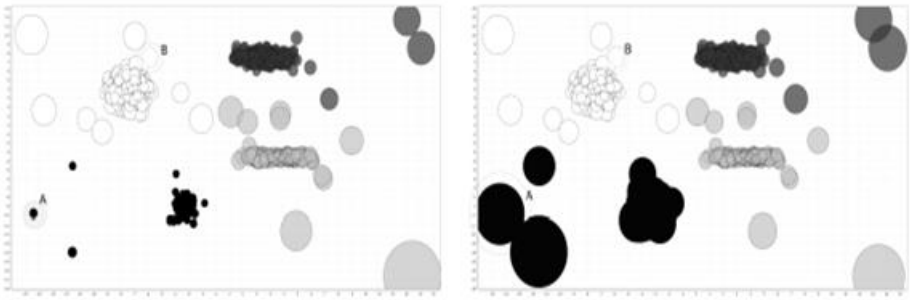
Cluster-Based Local Outlier Factor (CBLOF)

Ο συγκεκριμένος αλγόριθμος προσπαθεί να ανιχνεύσει τις ακραίες τιμές μέσω της[12] συσταδοποίησης. Όπως είχε αναφερθεί παραπάνω αυτή η μέθοδος για να βρει ακραίες τιμές ομαδοποιεί σύνολα δεδομένων που συσχετίζονται ή είναι όμοια με κάποιον τρόπο σε διάφορες συστάδες. ‘Οσα δεδομένα δεν ανήκουν σε κάποια συστάδα ή δημιουργούν μία συστάδα με ελάχιστα στοιχεία με χαμηλή πυκνότητα μακριά από το κέντρο της συστάδας θεωρούνται ακραίες τιμές. Ο CBLOF χρησιμοποιεί το αποτέλεσμα τέτοιων αλγορίθμων για την εύρεση ανωμαλιών, ο πιο δημοφιλής είναι ο k-means (συνήθως η πολυπλοκότητα του αλγορίθμου CBLOF υπολογίζεται κοντά σε $O(n^2)$ και εξαρτάται από τον αλγόριθμο συσταδοποίησης). Ο αλγόριθμος χωρίζει τις συστάδες σε μεγάλες και μικρές και έπειτα υπολογίζει το score του CBLOF για κάθε σημείο που προκύπτει από το μέγεθος της συστάδας στην οποία βρίσκεται επί την απόσταση του σημείου από την πιο κοντινή μεγάλη συστάδα και εδώ προκύπτει ένα όριο που αν ξεπεραστεί θα θεωρηθεί η εξεταζόμενη τιμή ως ανωμαλία.



(παράδειγμα υπολογισμού CBLOF, εικόνα από [12])

Τέλος υπάρχει άλλη μία προσέγγιση που δεν λαμβάνει υπόψιν το μέγεθος της συστάδας όταν υπολογίζει το score του CBLOF (unweighted-CBLOF) γιατί σε κάποιες περιπτώσεις όπως στο παράδειγμα παρακάτω επηρεάζει αρνητικά τα αποτελέσματα σε μικρές συστάδες. Έστω ότι έχουμε δύο συστάδες προς εξέταση η μία είναι αρκετά πιο μακριά συγκριτικά με την άλλη απλά επειδή η δεύτερη βρίσκεται κοντά σε μεγάλη συστάδα πολλαπλασιάζεται με το μέγεθος της και έτσι το score του CBLOF δίνει μη επιθυμητά αποτελέσματα.

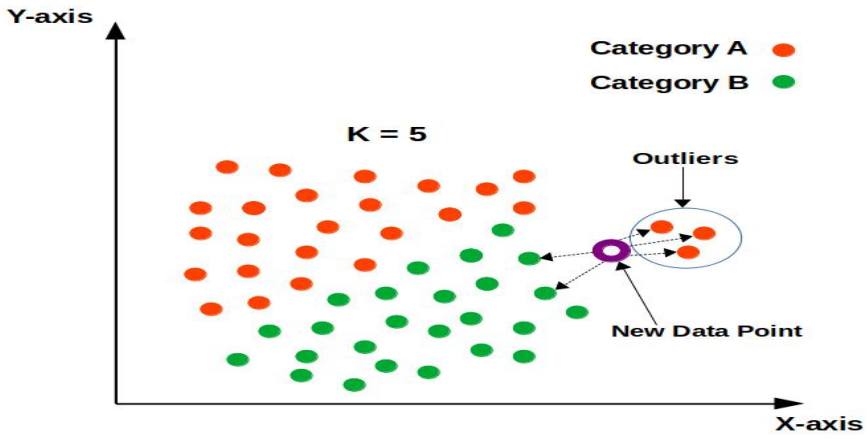


(αριστερά παράδειγμα CBLOF και δεξιά unweighted CBLOF , εικόνα από [12])

(Υποκεφάλαιο 5.3 K –NEAREST-NEIGHBORS)

K –NEAREST-NEIGHBORS

Είναι ένας χαρακτηριστικός αλγόριθμος ανίχνευσης καθολικών ανωμαλιών[13] (global anomaly). Λειτουργεί διαλέγοντας ένα K αριθμό κοντινότερων γειτόνων ενός σημείου και εξετάζει χωρικά βάση της απόστασης, με τον μέσο όρο των αποστάσεων του σημείου προς όλους τους γείτονές του. Αυτός ο μέσος όρος υπολογίζεται για κάθε σημείο του συνόλου δεδομένου και οι μεγαλύτεροι μέσοι όροι αποστάσεων σημείων που ξεπερνούν κάποιο όριο που ορίζουμε θεωρούνται ως ακραία σημεία. Η απόσταση μπορεί να υπολογιστεί με διάφορους τρόπους, η πιο γνωστή είναι η ευκλείδεια απόσταση αλλά μπορεί να χρησιμοποιηθεί και mahalanobis distance που είναι ισχυρή σε δεδομένα πολλών διαστάσεων ή και manhattan distance σε συγκεκριμένες περιπτώσεις. Έχει σχετικά μικρότερη πολυπλοκότητα με άλλους αλγόριθμους ανίχνευσης ανωμαλιών, δηλαδή μεγαλύτερη ταχύτητα. Η δυσκολία είναι να βρεθεί κατάλληλο αριθμό K ώστε να είναι αποτελεσματικός, όσο μεγαλύτερο το K τόσο πιο αργός γίνεται ο αλγόριθμος αφού υπολογίζει περισσότερες αποστάσεις γειτονικών σημείων. Πάλι όπως και σε κάποιους προηγούμενους αλγόριθμους είναι δύσκολο να βρούμε το βέλτιστο K και θέλει κάποιες δοκιμές ώστε να αποφασίσουμε. Αν το K έχει μεγάλες τιμές τότε υπάρχει μεγαλύτερη ακρίβεια στον διαχωρισμό κανονικής και ακραίας τιμής αλλά είναι πιο χρονοβόρο, αν το K έχει μικρή τιμή έχει χαμηλότερη ακρίβεια και μπορεί να δώσει λάθος αποτελέσματα. Επίσης δεν χρειάζεται να γνωρίζει την κατανομή για να κάνει προβλέψεις και να εντοπίσει ακραίες τιμές, ο διαχωρισμός γίνεται μόνο χωρικά βάση των αποστάσεων.



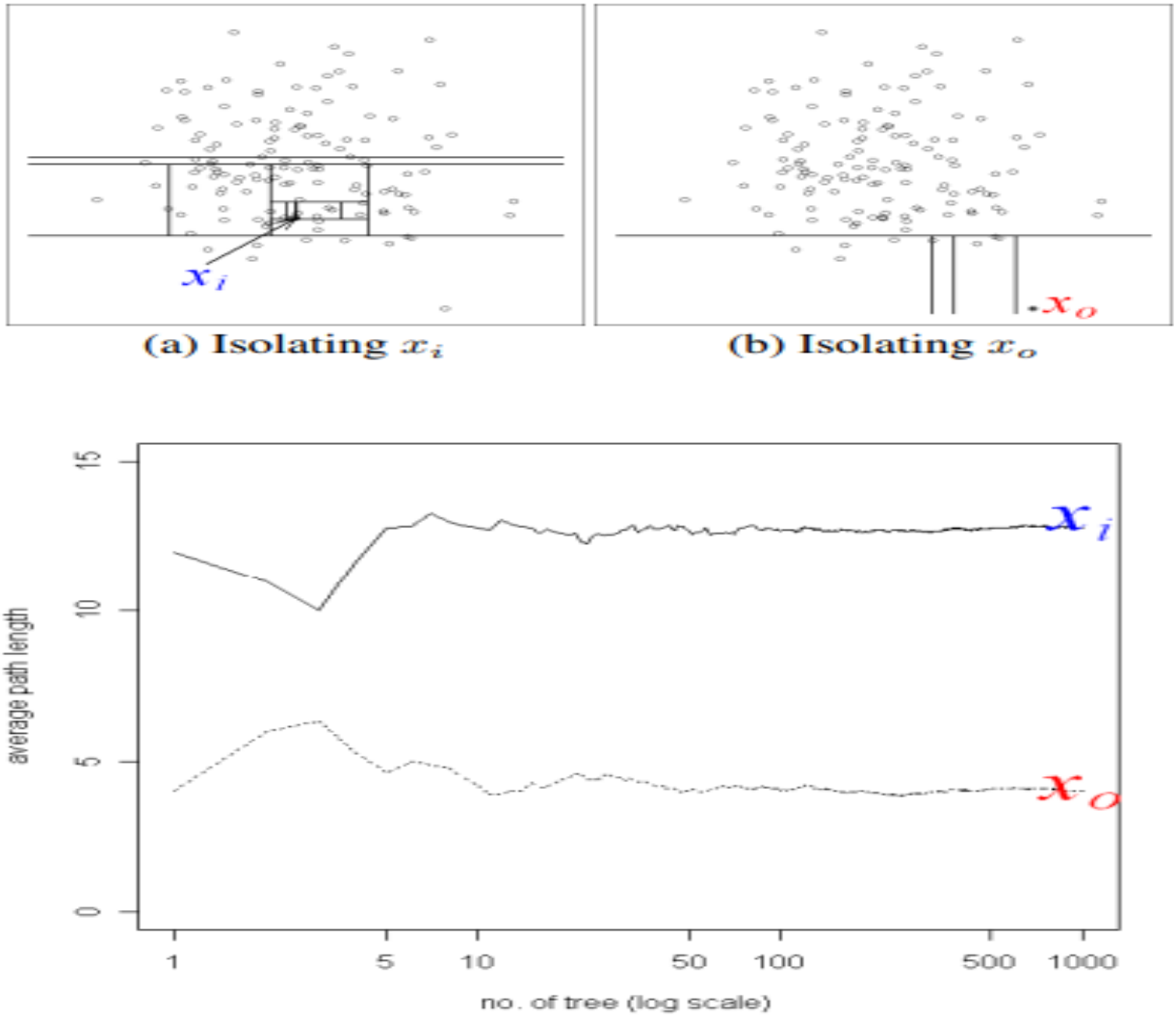
(παράδειγμα λειτουργίας knn, εικόνα από [20]))

(Υποκεφάλαιο 5.4 Isolation Forest)

Isolation Forest

Είναι ένας αλγόριθμος που για να επιτύχει την ανίχνευση ανωμαλιών βασίζεται [14]σε μέθοδο model-based αλλά και graph-based. Η κύρια διαφορά με τους υπόλοιπους αλγόριθμους τέτοιου είδους είναι ότι ενώ οι άλλοι μαθαίνουν πως είναι η κανονική τιμή και βάση αυτής διαχωρίζουν τις ακραίες τιμές, κάτι το οποίο δεν είναι η βέλτιστη τακτική αφού εξειδικεύονται στο να φτιάξουν ένα προφίλ για τις κανονικές τιμές, αλλά όχι στο να ανιχνεύουν τις ακραίες, κάτι που τους κάνει λιγότερο αποτελεσματικούς. Τέλος, απευθύνονται σε σύνολα δεδομένων με λίγες διαστάσεις και μικρό πλήθος λόγω της μεγάλης πολυπλοκότητας τους που δεν τους κάνει εφαρμόσιμους πρακτικά σε άλλα πιο σύνθετα σύνολα δεδομένων. Το Isolation Forest λύνει αυτά τα προβλήματα μιας και είναι ένας αλγόριθμος που απομονώνει τις ακραίες τιμές και είναι ικανός να χειρίζεται μεγάλα σύνολα δεδομένων με πολλά χαρακτηριστικά έχοντας χαμηλή πολυπλοκότητα (περίπου O(n)) συγκριτικά με άλλους model-based αλγορίθμους. Ο λόγος της χαμηλής πολυπλοκότητας οφείλεται στο ότι δεν χρησιμοποιεί μεθόδους density-based ή distance-based που μπορεί να είναι κοστοβόρες για να επιτύχει την ανίχνευση. Αυτό που εκμεταλλεύεται και χρησιμοποιεί στην ανίχνευση των ανωμαλιών είναι δύο κύρια χαρακτηριστικά τέτοιων τιμών, είναι πολύ λίγα συγκριτικά με τις κανονικές τιμές σε ένα σύνολο και διαφέρουν αρκετά από τις υπόλοιπες τιμές του συνόλου. Αξιοποιώντας αυτά τα χαρακτηριστικά ο αλγόριθμος απομονώνει τα στοιχεία του συνόλου αναπαριστώντας τα σε δενδρική δομή. Βάση όσων αναφέραμε προηγουμένως οι ακραίες τιμές θα είναι πιο εύκολο να απομονωθούν, δηλαδή δεν θα χρειάζονται αρκετοί διαχωρισμοί στα δεδομένα οπότε οι ακραίες τιμές θα βρίσκονται κοντά στην ρίζα κάθε δέντρου. Οι κανονικές τιμές που θα έχουν πολύ μεγαλύτερο μήκος διαδρομής στο δέντρο. Στην παρακάτω εικόνα φαίνεται πόσο εύκολο είναι να απομονωθεί μία κανονική τιμή (αριστερά) και για μία ακραία τιμή (δεξιά) και πόσο είναι το μήκος της διαδρομής του δέντρου απομόνωσης για κάθε

σημείο αντίστοιχα. Το δέντρο που κατασκευάζεται για να απομονωθεί η ακραία τιμή κατά μέσο όρο (έπειτα από παραγωγή 1000 δένδρων) έχει μήκος διαδρομής 4 ενώ η κανονική 12 και αντιλαμβανόμαστε ότι το μήκος διαδρομής του δέντρου ισούται με τον αριθμό των διαχωρισμών που χρειάζονται ώστε να απομονωθεί πλήρως ένα σημείο από το υπόλοιπο σύνολο στον χώρο.



(τρόπος λειτουργίας isolation forest, εικόνα από [14])

Έχοντας εξηγήσει τι κάνει το iforest θα εξηγήσουμε πως επιτυγχάνεται αυτό μέσω των isolation trees. Επιλέγοντας μια υποομάδα του συνόλου δεδομένων αρχίζουμε και τα χωρίζουμε διαλέγοντας αυθαίρετα χαρακτηριστικά του συνόλου δεδομένου με μία συγκεκριμένη τιμή σαν όριο για τον διαχωρισμό. Κάθε κόμβος μπορεί να έχει μέχρι δύο 'παιδιά' δηλαδή δυαδικό δέντρο. Ο διαχωρισμός κάθε κόμβου επαναλαμβάνεται μέχρι το μήκος του δέντρου να φτάσει ένα όριο που έχουμε θέσει, μείνει μόνο μία εγγραφή από την υποομάδα ή όλες οι εγγραφές στην υποομάδα έχουν τις ίδιες τιμές. Έπειτα για κάθε εγγραφή μετράμε πόσοι διαχωρισμοί έγιναν για να απομονωθεί από την υποομάδα της ή αλλιώς το μήκος του isolation tree. Μετά ταξινομούμε κάθε εγγραφή ανάλογα το μήκος του isolation tree το οποίο θα είναι το σκορ του αλγορίθμου (h(x)) για το πόσο θεωρείται ανωμαλία, οι ακραίες τιμές θα έχουν τα χαμηλότερα μήκη. Έχουμε το μέσο μήκος όλων των δέντρων που συμβολίζεται ως

E(h(x)) όμως για να υπολογιστεί το μέσο μήκος πρέπει να κανονικοποιήσουμε τα h(x)) ώστε να είναι πιο συγκρίσιμα γιατί κάποια έστω και αν είναι λίγα μπορεί να έχουν μεγάλες τιμές.
Δανείζεται μία μέθοδο κανονικοποίησης από τα δυαδικά δέντρα

1. $$c(n) = 2H(n-1) - (2(n-1)/n),$$

και το σκορ για την ανωμαλία προκύπτει από τον παρακάτω τύπο.
2.

$$s(x,n) = 2^{-\frac{E(h(x))}{c(n)}}$$

Βάση του τύπου, αν το S τείνει το 1 τότε σίγουρα είναι ανωμαλία, αν είναι μικρότερο του 0,5 τότε θεωρείται κανονική τιμή, αν οι τιμές από το δείγμα είναι όλες κοντά στο 0,5 τότε μπορεί να θεωρηθεί ότι το δείγμα δεν έχει κάποια ακραία τιμή.

(Υποκεφάλαιο 5.5 ONE-CLASS SUPPORT VECTOR MACHINES (OCSVM))

Βασίζεται στην μέθοδο **SVM** και μπορεί να έχει δύο διαφορετικές [15]λειτουργίες είτε regression είτε κατηγοριοποίηση με επίβλεψη (supervised classification) αλγορίθμου. Διαχωρίζει διαφορετικά δεδομένα μεταξύ τους βρίσκοντας ένα όριο ή 'γραμμή' ώστε να μεγιστοποιεί την απόσταση μεταξύ των δύο ομάδων χωρικά και να είναι όσο το δυνατόν πιο μακριά γίνεται από το κοντινότερο σημείο κάθε ομάδας. Σε περίπτωση που τα δεδομένα δεν μπορούν να διαχωριστούν γραμμικά το

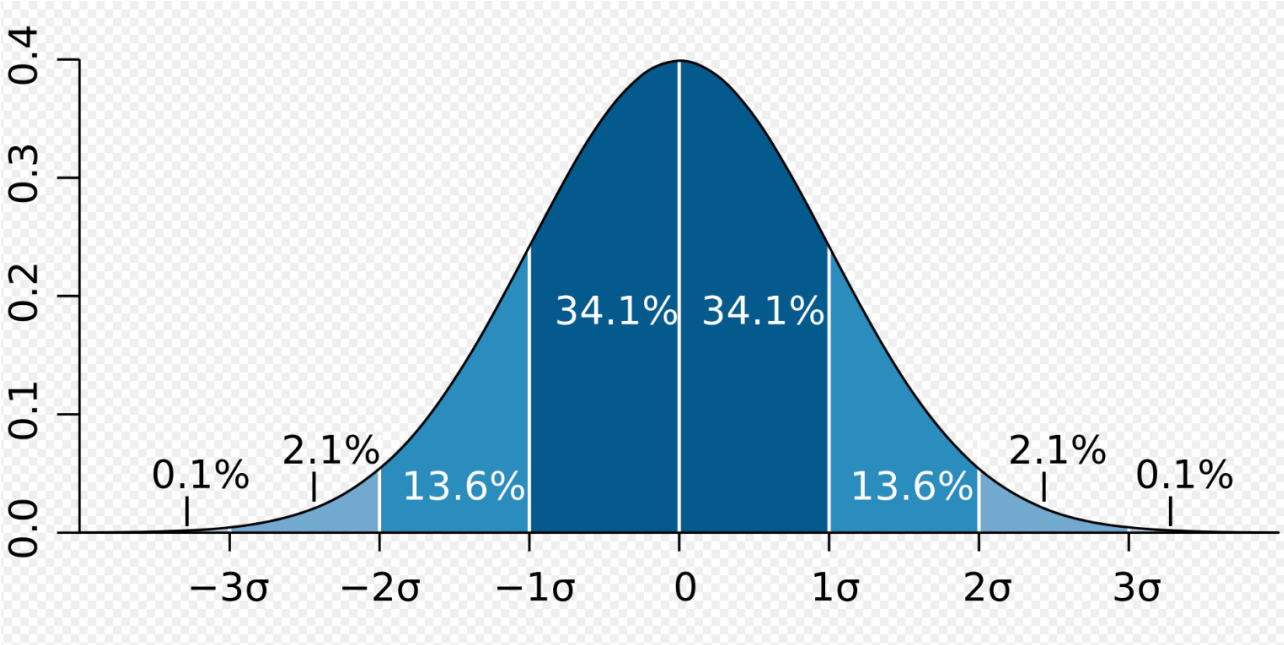
SVM έχει βρει έναν τρόπο Kernel method ώστε να αναπαριστά τα δεδομένα σε περισσότερες διαστάσεις κάνοντας πιο εύκολο να διαχωριστούν Με αυτόν τον έξυπνο τρόπο κάνει το **SVM** ισχυρό στον διαχωρισμό δύσκολων μοτίβων δεδομένων. Ο αλγόριθμος που θα ασχοληθούμε δηλαδή ο **ONE-CLASS SUPPORT VECTOR MACHINES (OCSVM)** που είναι μία παραλλαγή του **SVM** που χρησιμοποιείται όμως για διαφορετικό σκοπό. Αντί να διαχωρίζει τα δεδομένα διαφορετικών ομάδων μεταξύ τους, ο σκοπός του είναι να ανιχνεύει ακραίες τιμές. Είναι αλγόριθμος unsupervised learning και χρησιμοποιεί **kernel-based classifier** όπως στο **SVM** για να ξεχωρίσει τις δύο ομάδες με ένα όριο. Στην ανίχνευση ακραίων τιμών η μία ομάδα είναι οι κανονικές τιμές και όσα σημεία δεν ταιριάζουν στην ομάδα θεωρούνται ακραίες τιμές. Υπάρχουν δύο προσεγγίσεις στην οριοθέτηση μεταξύ των σημείων. Η πρώτη σαν όριο έχει σφαιρικό σχήμα και περικλείει την πλειοψηφία των δεδομένων 'κανονικά' και ότι βρίσκεται εξωτερικά είναι ακραία τιμή. Έχει μία παράμετρο που καθορίζει το μέγεθος του κύκλου που περικλείει τα δεδομένα, όσο πιο μεγάλο τόσο περισσότερες ακραίες τιμές θα ομαδοποιηθούν μαζί με τα κανονικά δεδομένα. Η δεύτερη το όριο το θέτει γραμμικά σαν μια ευθεία μεταξύ των σημείων και της αρχής των συντεταγμένων που διαχωρίζει ένα μέρος των ανωμαλιών με τα υπόλοιπα δεδομένα. Η πολυπλοκότητα του αλγορίθμου δεν είναι γνωστή με ακρίβεια καθώς υπάρχουν διάφοροι παράμετροι ανάλογα με την περίπτωση όπως οι διαστάσεις των δεδομένων ή η μέθοδος που θα χρησιμοποιήσουμε, γενικά είναι ένας αργός αλγόριθμος.

(Υποκεφάλαιο 5.6 Empirical Cumulative Distribution Functions (ECOD)

Empirical Cumulative Distribution Functions (ECOD)

Ο αλγόριθμος που θα αναλύσουμε **Empirical Cumulative Distribution Functions [16] (ECOD)** δημιουργήθηκε λαμβάνοντας υπόψιν κάποια

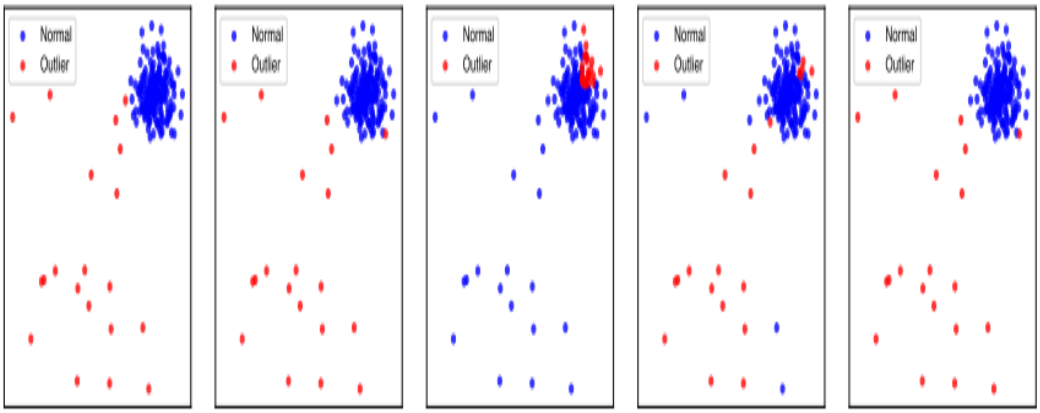
χαρακτηριστικά, δυσκολίες και αδυναμίες άλλων αλγορίθμων. Οι περισσότεροι αλγόριθμοι χρειάζονται να βρεθούν οι κατάλληλες τιμές για τις μεταβλητές τους και η επίδοση τους κρίνεται αρκετά από αυτό, κάτι που δεν είναι εύκολο και θέλει αρκετές δοκιμές οπότε και χρόνο στην εκτέλεση τους. Αλγόριθμοι εύρεσης ανωμαλιών βάση των χαρακτηριστικών τους και των τρόπων λειτουργίας τους αντιμετωπίζουν δυσκολίες σε συγκεκριμένα σύνολα δεδομένων όπως για παράδειγμα αυτοί που ακολουθούν μεθόδους πυκνότητας (density -based). Οι συγκεκριμένοι αλγόριθμοι ή και αλγόριθμοι που υπολογίζουν αποστάσεις όταν μεγαλώνει ο αριθμός των διαστάσεων (curse of dimensionality) και συνεπώς την αύξηση των δεδομένων γίνονται πιο κοστοβόροι χρονικά και υπολογιστικά και χάνουν την ακρίβεια τους. Ο **ECOD** εκμεταλλευόμενος τα χαρακτηριστικά των ακραίων τιμών στα σύνολα δεδομένων, δηλαδή ότι είναι σπάνια φαινόμενα και αυτά θα βρεθούν κυρίως στα άκρα των κατανομών. Όπως για παράδειγμα την κανονική κατανομή δηλαδή στις τρεις τυπικές αποκλίσεις και μετά θα περιμένουμε να βρίσκονται τα σπάνια φαινόμενα και οι ανωμαλίες.



(παράδειγμα κανονικής κατανομής, εικόνα από [17])

Η προσέγγιση από τον αλγόριθμο δεν γίνεται έπειτα από υπόθεση ότι το σύνολο δεδομένων ακολουθεί κανονική κατανομή. Όταν υπάρχουν πολλά χαρακτηριστικά στα δεδομένα και κάνουν δύσκολο τον υπολογισμό του **ECOD** για να το αντιμετωπίσει υπολογίζει για κάθε διάσταση ξεχωριστά και όχι για το σύνολο δεδομένων συνολικά και έπειτα για να βρει ένα συνολικό outlier score πολλαπλασιάζει το κάθε σκορ από κάθε διάσταση θεωρώντας ότι οι διαστάσεις δεν είναι εξαρτημένες μεταξύ τους (κάτι που δεν ισχύει πάντα στα σύνολα δεδομένων). Τέλος, για να υπολογίσει την πιθανότητα κατά πόσο μία τιμή είναι ακραία, υπολογίζει για κάθε διάσταση του συνόλου δεδομένων τις πιο αριστερές και δεξιές τιμές τιμές στην ‘ουρά’ της κατανομής και μέσω αυτών μπορεί να διαπιστώσει πια τιμή θεωρείται ακραία. Έπειτα για κάθε στοιχείο του συνόλου για κάθε διάσταση συναθροίζει τις πιθανότητες των αριστερών και δεξιών τιμών για να προκύψει ένα σκορ για αυτήν το στοιχείο. Όσο μεγαλύτερο είναι το σκορ τόσο μεγαλύτερη η πιθανότητα να είναι ακραία τιμή. Με έναν έξυπνο

τρόπο ο αλγόριθμος για κάθε διάσταση διαλέγει είτε την αριστερή είτε την δεξιά ακραία πιθανότητα ανάλογα με την κυρτότητα της κατανομής των δεδομένων και δεν διαλέγει αυθαίρετα, με αποτέλεσμα να είναι πιο αποτελεσματικός και πιο γρήγορος από το να διαλέγουμε και τα δύο για κάθε διάσταση.



(παράδειγμα υπολογισμού κάθε διάστασης ανάλογα με την αριστερή ή δεξία κυρτότητα της κατανομής των δεδομένων από τον ECOD, εικόνα από [16])

Κλείνοντας συμπεραίνουμε ότι ο παραπάνω αλγόριθμος αν και απλός και εύκολος στην εκτέλεση. Είναι αρκετά αποτελεσματικός και γρήγορος (πολυπλοκότητα (O(nd) όπου n αριθμός στοιχείων και d αριθμός διαστάσεων) αφού δεν χρειάζεται βελτιστοποίηση στις παραμέτρους του και για την εύρεση των ανωμαλιών δεν χρειάζονται κοστοβόροι υπολογισμοί όπως αναφέρθηκαν στην αρχή της παραγράφου. Βάση όλων όσων αναφέρθηκαν προκύπτει ότι είναι μια καλή επιλογή για σύνθετα και μεγάλα σύνολα δεδομένων με πολλές διαστάσεις.

(Υποκεφάλαιο 5.7 Histogram-based Outlier Score (HBOS))

Histogram-based Outlier Score (HBOS)

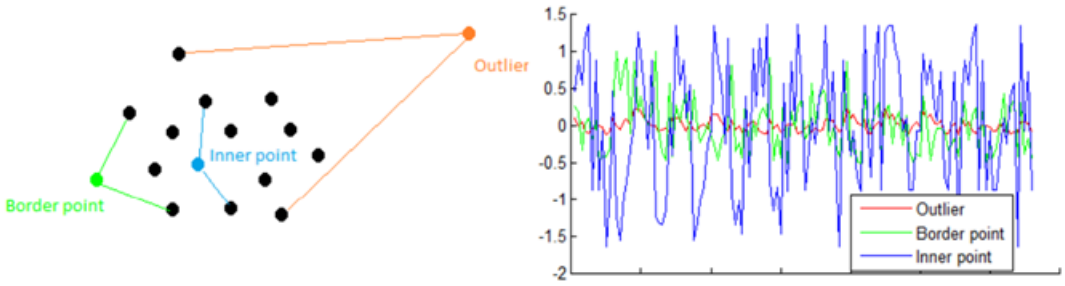
Είναι ένας αλγόριθμος που ακολουθεί στατιστική μέθοδο για την ανίχνευση ακραίων[18] τιμών. Είναι γρήγορος και με χαμηλή πολυπλοκότητα άρα και κατάλληλος για μεγάλα σύνολα δεδομένων. Χειρίζεται μονοδιάστατα τα δεδομένα (χωρίς να θεωρεί ότι συσχετίζονται μεταξύ τους, κάτι που σε κάποια σύνολα δεδομένων δεν ισχύει με αποτέλεσμα να χάνεται πληροφορία). Για κάθε ένα χαρακτηριστικό φτιάχνει ένα ‘καλάθι’ (ιστόγραμμα) ξεχωριστά. Μπορεί να χειριστεί

και κατηγορικά δεδομένα για κάθε χαρακτηριστικό μετρώντας πόσο συχνά εμφανίζεται η κάθε μια από τις ομάδες του χαρακτηριστικού, δηλαδή την συχνότητα της. Όταν τα χαρακτηριστικά των δεδομένων είναι αριθμητικά τότε υπάρχουν δύο τρόποι στην λειτουργία του **HBOS** είτε το 'καλάθι' να είναι συγκεκριμένου μεγέθους είτε μεταβλητού. Ο πρώτος τρόπος έχει έναν αριθμό 'καλαθιών' ίσου μεγέθους (π.χ. εύρος αριθμών 0-15,16-30 κ.ο.κ.) και η συχνότητα των αριθμών που πηγαίνει σε κάθε 'καλάθι' μας δίνει το τελικό μέγεθος του κάδου. Ο δεύτερος τρόπος ταξινομεί τα στοιχεία σε αύξουσα σειρά και διαιρεί τον αριθμό τον στοιχείων προς τον αριθμό των 'καλαθιών' που θέλουμε να έχουμε, έτσι κάθε 'καλάθι' θα έχει ισόποσο αριθμό στοιχείων. Με αυτόν τον τρόπο τα 'καλάθια' που έχουν μεγαλύτερο εύρος τιμών θα μπορούν να θεωρηθούν ως σημεία χαμηλής πυκνότητας. Εάν υπάρχουν αρκετές ίδιες τιμές που αριθμός τους είναι μεγαλύτερος από τον αριθμό των ΄καλαθιών΄ τότε στο συγκεκριμένο 'καλάθι' που θα εκχωρηθούν τα στοιχεία θα επιτρέπεται να τοποθετηθούν περισσότερα στοιχεία ώστε να αντιπροσωπευτικό. Οι δύο μέθοδοι υπάρχουν για να χειριστούν διαφορετικές κατανομές στα δεδομένα αν π.χ. υπάρχουν μεγάλες διαφορές στις τιμές των στοιχείων τότε ο δεύτερος τρόπος είναι πιο αποτελεσματικός. Επειδή σε σύνολα δεδομένων οι ακραίες τιμές συνήθως διαφέρουν πολύ από τις κανονικές συνιστάται να χρησιμοποιείται ο δεύτερος τρόπος. Σε περίπτωση που η κατανομή δεν είναι γνωστή και είναι δύσκολο να βρεθεί ένας αριθμός για τα ΄καλάθια' μια καλή λύση είναι να χρησιμοποιούμε ως αριθμό 'καλαθιών' την ρίζα του αριθμού του συνόλου δεδομένων. Έπειτα για οποιονδήποτε από τους τρόπους που χρησιμοποιούμε φτιάχνουμε ένα ιστόγραμμα για κάθε χαρακτηριστικό του συνόλου δεδομένων που δείχνει για κάθε 'καλάθι' τον αριθμό των δεδομένων που υπάρχουν(ύψος), κανονικοποιούμε το κάθε 'καλάθι' ώστε στο ιστόγραμμα που ανήκει η κάθε διάσταση να έχει την ίδια βαρύτητα στον υπολογισμό και το HBOS κάθε στοιχείου υπολογίζεται από τον πολλαπλασιασμό του αντίστροφου κλάσματος του κανονικοποιημένου ύψους των ιστογραμμάτων στο οποίο ανήκει μεταξύ τους. Τέλος όσο μεγαλύτερο είναι το σκορ ενός σημείου τόσο πιο πιθανό είναι αυτό να είναι ακραία τιμή.

(Υποκεφάλαιο 5.8 Angle-based Outlier Detection (ABOD))

Ο ABOD για να εντοπίσει τις ακραίες τιμές υπολογίζει για κάθε σημείο πόσο[19] μεγάλο είναι το φάσμα της γωνίας του και το συγκρίνει με των άλλων σημείων, όσο μικρότερο είναι συγκριτικά με τα υπόλοιπα τόσο πιο πιθανόν είναι να είναι ακραία τιμή. Η συγκεκριμένη μέθοδος δεν επηρεάζεται τόσο πολύ από τις πολλές διαστάσεις στον χώρο όπως π.χ. οι μέθοδοι που βασίζονται στην απόσταση και έτσι την καθιστά έναν αποτελεσματικό αλγόριθμο για την εύρεση ανωμαλιών σε δεδομένα πολλών διαστάσεων. Λόγο όμως της μεγάλης πολυπλοκότητας που έχει ο αλγόριθμος O(dn3) όπου d αριθμός των χαρακτηριστικών του συνόλου δεδομένων και n ο αριθμός των

στοιχείων, δημιουργήθηκαν παραλλαγές του αλγορίθμου για να μειώσει τον χρόνο εκτέλεσης ώστε να είναι λειτουργικός και για μεγάλα σύνολα δεδομένων πολλών διαστάσεων. Από αυτές η μία παραλλαγή που επιτυγχάνει την μικρότερη πολυπλοκότητα σχεδόν γραμμική O(n log n(d + log n)) όπου d αριθμός των χαρακτηριστικών του συνόλου δεδομένων και n ο αριθμός των στοιχείων.



(υπολογισμός της διακύμανσης της γωνίας για κάθε σημείο, εικόνα από [19])

Στην εικόνα παραπάνω φαίνεται πως λειτουργεί ο αλγόριθμοςΑπό μια υποομάδα στοιχείων από το σύνολο εξετάζουμε την γωνία κάθε στοιχείου και την συγκρίνουμε με τις άλλες. Τα στοιχεία με μεγάλη διακύμανση θεωρούνται κανονικές τιμές ενώ όσα έχουν μικρή ακραίες. Οπτικά γίνεται πιο κατανοητό όλο αυτό. Είναι λογικό τα στοιχεία που βρίσκονται σε περιοχές με υψηλή πυκνότητα περιτριγυρισμένα από άλλα στοιχεία σαν συστάδες να έχουν μεγάλη διακύμανση. Οι γωνίες δηλαδή μεταξύ τους είναι σε πολλά σημεία άρα θα έχουν μεγάλη διαφορά μεταξύ τους αφού θα έχουν σημεία προς πολλές κατευθύνσεις ενώ οι ακραίες τιμές που είναι απομακρυσμένες από τις υπόλοιπες τιμές το αντίθετο. Αυτό που περιγράψαμε είναι η παραλλαγή του αλγορίθμου που έχει χαμηλότερη πολυπλοκότητα από την αρχική που ήταν κυβική. Υπολογίζει για κάθε σημείο το φάσμα της γωνίας από το συνημίτονό της επηρεαζόμενο από την απόσταση κάτι που ήταν εξαιρετικά κοστοβόρο και δεν ήταν τόσο αποτελεσματικό όσο αυξανόντουσαν οι διαστάσεις του συνόλου δεδομένων. Η παραλλαγή του αλγορίθμου και είναι πιο γρήγορη και δεν χρειάζεται βελτιστοποίηση στις παραμέτρους αφού δεν χρειάζεται κάποια στον υπολογισμό κάτι που είναι πολύ χρήσιμο ειδικά σε περιπτώσεις με μέθοδο χωρίς επίβλεψη που είναι δύσκολο να βρεθούν οι βέλτιστες παράμετροι.

(Υποκεφάλαιο 5.9 LONG SHORT-TERM MEMORY AUTO ENCODER)

LONG SHORT-TERM MEMORY AUTOENCODER

Το Long Short-Term Memory (LSTM) είναι πιο εξελιγμένο μοντέλο που βασίστηκε [24] στο Recurrent Neural Networks (RNN). Δημιουργήθηκε ώστε να διορθώσει το πρόβλημα vanishing/exploding gradient που προκύπτει την διάρκεια της εκπαίδευσης του νευρωνικού δικτύου. Κυριότερα όμως και ειδικά για το παράδειγμα μας για την

αποθήκευση και ανάκτηση πληροφοριών τόσο μακροπρόθεσμα όσο και βραχυπρόθεσμα. Σε αντίθεση με το RNN που είχε την ικανότητα αποθήκευσης μνήμης μόνο για σύντομο χρονικό διάστημα. Το LSTM είναι πολύ πιο αποτελεσματικό στην ανάλυση χρονοσειρών αφού είναι ικανό να να κρατάει την πληροφορία των προηγούμενων στοιχείων και έτσι να βγουν συμπεράσματα μέσω της αλληλοσχυσχέτισης μεταξύ τους όσο μακριά και αν βρίσκονται χρονικά. Αυτό επιτυγχάνεται μέσω μιας πιο σύνθετης δομής που αποτελείται το συγκεκριμένο νευρωνικό δίκτυο και θα αναληθεί παρακάτω. Για να απλοποιηθεί η ανάλυση του νευρωνικού δικτύου θα το χωρίσουμε σε δύο μέρη το πρώτο είναι το LSTM και το δεύτερο το encoder-decoder.

ΔΟΜΗ LSTM

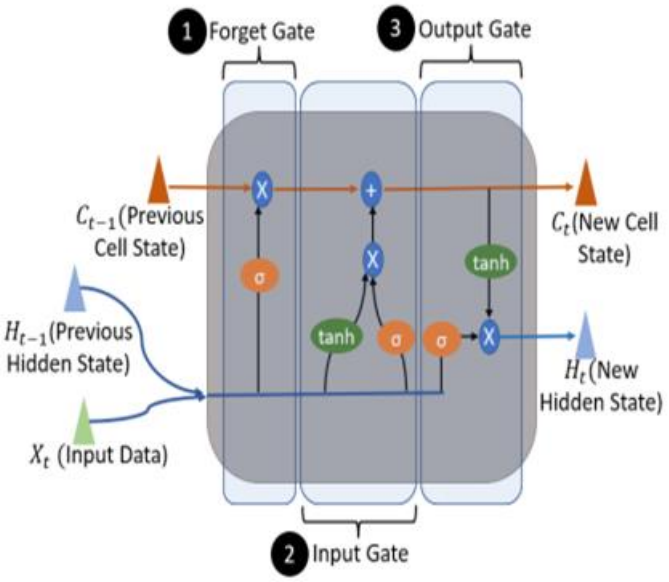
Αρχικά στο LSTM υπάρχει το cell state το οποίο διατηρεί την πληροφορία από τα προηγούμενα στοιχεία δηλαδή (long-term memory). Το previous hidden state που διατηρεί την πληροφορία από το αμέσως προηγούμενο στοιχείο (short-term memory). Την είσοδο που είναι για κάθε στοιχείο που εξετάζεται την συγκεκριμένη χρονική στιγμή. Επίσης για τον έλεγχο των πληροφοριών, δηλαδή ποιες είναι χρήσιμες και θα διατηρούνται και ποιες όχι. υπάρχουν τρεις ‘πύλες’. Κάθε μία θα ελέγχει την πληροφορία που μεταφέρεται μέσα στο LSTM.

Η πρώτη είναι η forget gate που σαν είσοδο δέχεται το στοιχείο που είναι στην είσοδο εκείνη την χρονική στιγμή και το previous hidden state. Τα περνάει σε ένα νευρωνικό δίκτυο το οποίο χρησιμοποιεί σιγμοειδή (sigmoid) συνάρτηση ενεργοποίησης (activation function). Η συνάρτηση ενεργοποίησης δέχεται σαν είσοδο μία μεταβλητή του άξονα χ και παράγει μία τιμή στον άξονα y. Στην σιγμοειδή συνάρτηση η τιμή που θα παράγεται θα έχει εύρος από 0 εώς 1. Με αυτόν τον τρόπο εξετάζει αν η πληροφορία που δέχτηκε σαν είσοδο είναι χρήσιμη και παράγει τιμές κοντά στο 1 αλλιώς τιμές κοντά στο 0 και τις πολλαπλασιάζει με το hidden state (short-term memory) έτσι προκύπτει ποια ‘μέρη’ πληροφορίας από το cell state δηλαδή την long-term memory θα διατηρηθούν.

Η δεύτερη είναι η input gate κάνει δύο λειτουργίες. Δέχεται το previous hidden state και το στοιχείο που είναι στην είσοδο εκείνη την χρονική στιγμή και τα συνδυάζει πολλαπλασιάζοντας τις τιμές από τα βάρη (weights) που καθορίζουν την βαρύτητα του κάθε χαρακτηριστικού σε μία συνάρτηση ενεργοποίησης (activation function) την λειτουργία της οποίας εξηγήσαμε στην προηγούμενη παράγραφο όμως η συγκεκριμένη η tanh παράγει τιμές από -1 έως 1 και η τιμή προστίθεται με ένα “bias” που χρησιμεύει στην μετατόπιση της συνάρτησης ενεργοποίησης (activation function). Αν η τελική τιμή είναι αρνητική τότε μειώνεται η επίδραση του στοιχείου στο cell state αλλιώς το αντίθετο. Η δεύτερη λειτουργία είναι παρόμοια με του forget gate χρησιμοποιώντας την ίδια συνάρτηση ενεργοποίησης (σιγμοειδή που δίνει αποτελέσματα από 0 εώς 1) περνώντας όμως τις τιμές πολλαπλασιάζοντας τις με weights και προσθέτοντας το αποτέλεσμα της σιγμοειδούς με bias. Έτσι αν είναι χρήσιμη η πληροφορία θα έχει τιμή κοντά στο 1 αλλιώς στο 0. Τέλος οι 2 τιμές που προκύπτουν από τις λειτουργίες πολλαπλασιάζονται μεταξύ τους και προστίθενται στο cell state δηλαδή long-term memory του LSTM. Με λίγα λόγια το input gate αποφασίζει αν η καινούρια πληροφορία είναι χρήσιμη και πόση από αυτήν πρέπει να διατηρηθεί.

Η Τρίτη η output gate πάλι περνάει το το previous hidden state και το στοιχείο που είναι στην είσοδο εκείνη την χρονική στιγμή και τα συνδυάζει πολλαπλασιάζοντας τις τιμές από τα βάρη (weights) που καθορίζουν την βαρύτητα του κάθε χαρακτηριστικού σε μία σιγμοειδή (που δίνει αποτελέσματα από 0 έως 1) συνάρτηση ενεργοποίησης (activation function) και στο αποτέλεσμα της συνάρτησης προστίθεται ένα bias. Επίσης περνάει το cell state (long-term memory) μέσα από μία συνάρτηση ενεργοποίησης tanh (που παράγει τιμές από -1 έως 1). Τέλος οι δύο τιμές που προκύπτουν από τους υπολογισμούς, δηλαδή η τιμή από τον πρώτο υπολογισμό και το καινούριο cell state πολλαπλασιάζονται μεταξύ τους και η τιμή τους διαμορφώνει το καινούριο hidden state. Με λίγα λόγια το output gate υπολογίζει το καινούριο hidden state που θα είναι το previous hidden state στο επόμενο στοιχείο της χρονοσειράς και αλλάζει την τιμή του cell state που θα είναι το previous cell state στο επόμενο στοιχείο της χρονοσειράς. Ότι έχουμε περιγράψει μέχρι στιγμής θα επαναληφθεί μέχρι το τελευταίο στοιχείο υπό εξέταση από το LSTM.

Η εικόνα παρακάτω αναπαριστά την δομή ενός LSTM, ότι δηλαδή περιγράψαμε προηγουμένως.

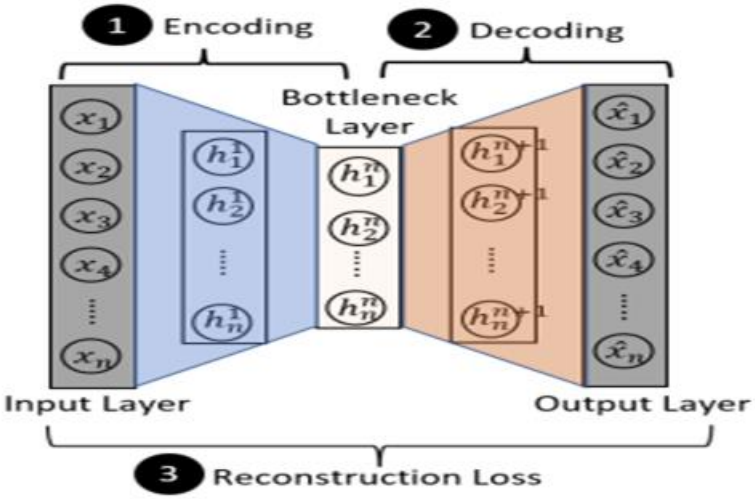


(δομή LSTM, εικόνα από [24])

ΔΟΜΗ AUTOENCODER

Είναι ένα νευρωνικό δίκτυο που βασίζεται στην μέθοδο μη επιβλεπόμενης μάθησης. Έχει τρεις φάσεις στην λειτουργία του. Η πρώτη που είναι η κωδικοποίηση περνάει τα δεδομένα σε αναπαράσταση λιγότερων διαστάσεων πολλαπλασιάζοντας κάθε στοιχείο με βάρη, περνώντας το μέσα από μία συνάρτηση ενεργοποίησης και τέλος προσθέτοντας ένα bias στην τιμή. Έπειτα γίνεται η αποκωδικοποίηση παίρνοντας τα στοιχεία που παράχθηκαν από την κωδικοποίηση και περνώντας τα αφού τα έχουν πολλαπλασιάσει με άλλα βάρη (σε σχέση με του κωδικοποιητή) σε μία συνάρτηση ενεργοποίησης και προσθέτοντας ένα διαφορετικό bias (σε σχέση με του

κωδικοποιητή) στην τιμή. Τέλος στην Τρίτη φάση μέσω μιας συνάρτησης που υπολογίζει την διαφορά του input, δηλαδή των πραγματικών στοιχείων με τα στοιχεία που παράχθηκαν από τον decoder στην δεύτερη φάση. Ο σκοπός το autoencoder είναι να ελαχιστοποιήσει αυτήν την διαφορά της συνάρτησης. Βάση αυτής της διαφοράς παίρνεται η απόφαση για την ανίχνευση ανωμαλιών στην χρονοσειρά.

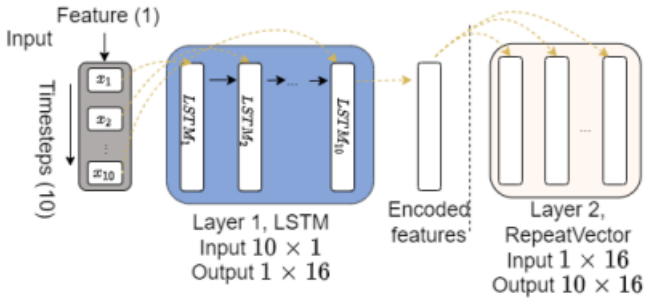


(δομή AUTOENCODER, εικόνα από [24])

ΛΕΙΤΟΥΡΓΙΑ ΚΑΙ ΔΟΜΗ LSTM AUTOENCODER

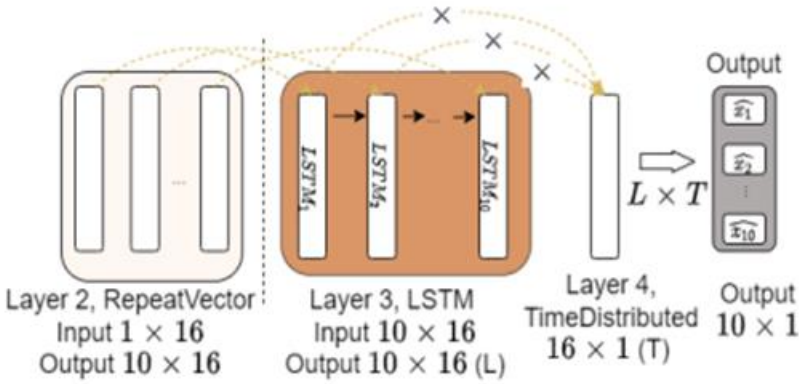
Έχοντας εξηγήσει την λειτουργία του **LSTM** και του **AUTOENCODER** θα εξηγήσουμε πως ο συνδυασμός τους οδηγεί στην λειτουργία του τελικού προγράμματος, δηλαδή την ανίχνευση ανωμαλιών σε χρονοσειρές μέσω των δύο διαδικασιών που εξηγήσαμε προηγουμένως**.** Πάλι θα χωρίσουμε την λειτουργία του προγράμματος σε καταστάσεις όπως στις προηγούμενες ενότητες.

Αρχικά στην είσοδο χωρίζουμε την χρονοσειρά σε μέρη σταθερού μεγέθους ανά χρονικές περιόδους. Έπειτα τα περνάμε σε ένα πίνακα με σειρές όσες τα σημεία της υποομάδας της χρονοσειράς και στήλες όσες τα χρονικά μέρη που χωρίσαμε την χρονοσειρά. Από την είσοδο αυτά τα δεδομένα από τον πίνακα θα εξετάζονται για κάθε μέρος της χρονοσειράς ξεχωριστά. Θα περνάνε στον LSTM encoder που θα περιέχει ένα LSTM layer για κάθε στοιχείο που θα συνδέεται με το επόμενο LSTM layer για το επόμενο στοιχείο κ.ο.κ. όπως περιγράψαμε στην δομή του LSTM. Όπως εξηγήσαμε στην προηγούμενη ενότητα κάθε LSTM περιέχει την long-term memory και επιλέγει αν θα συμπεριληφθεί η πληροφορία από το προηγούμενο LSTM δηλαδή του προηγούμενου στοιχείου και η πληροφορία του στοιχείου που εξετάζεται εκείνη την στιγμή στην long-term memory που θα περάσει στο επόμενο στοιχείο μέχρι το τελευταίο από μέρος της χρονοσειράς που εξετάζεται εκείνη την στιγμή. Όταν θα φτάσουμε στο τελευταίο στοιχείο της υποομάδας της χρονοσειράς που εξετάζεται εκείνη την στιγμή το αποτέλεσμα του long-term memory που θα περιέχει πληροφορία όλων των προηγούμενων στοιχείων της υποομάδας, αυτό θα είναι η έξοδος του encoder. Η έξοδος αποθηκεύεται σε έναν πίνακα μίας γραμμής με στήλες όσες του αριθμού των χαρακτηριστικών της κωδικοποίησης της υποομάδας του encoder. Αυτός ο πίνακας θα φτιαχτεί για κάθε υποομάδα της χρονοσειράς. Οπότε οι διαστάσεις το πίνακα θα είναι n x m, όπου n είναι ο αριθμός των υποομάδων και m των χαρακτηριστικών της κωδικοποίησης της υποομάδας του encoder.



(δομή LSTM ENCODER, εικόνα από [24])

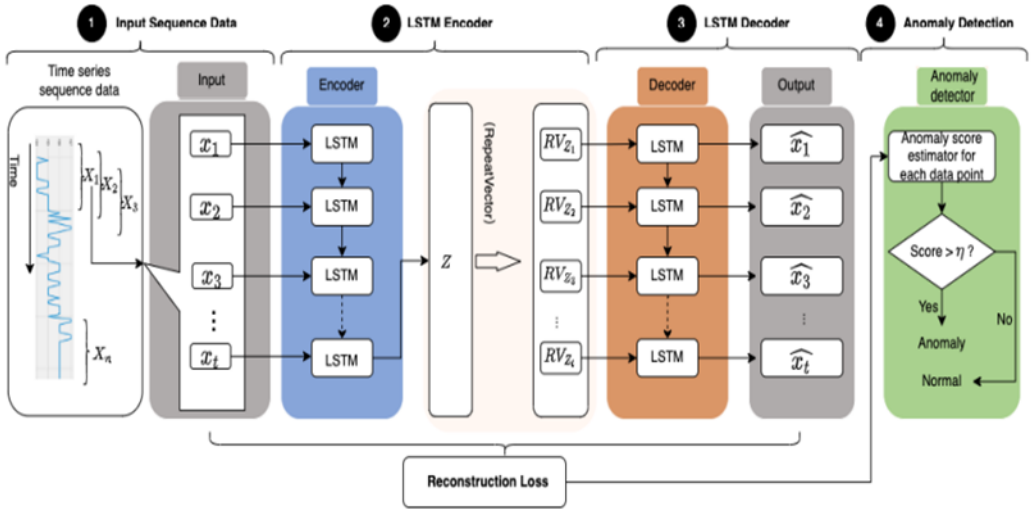
Μετά από τον encoder παίρνει σειρά ο decoder που σαν είσοδο θα δέχεται ξεχωριστά κάθε γραμμή του πίνακα n x m και θα περνάει σε κάθε LSTM ξεχωριστά. Τα LSTM θα είναι όσες οι υποομάδες της χρονοσειράς, άρα θα είναι ίσα με τα LSTM του encoder. Κάθε έξοδος από τα LSTM περιέχει την πληροφορία που υπολογίστηκε τα χαρακτηριστικά της κωδικοποίησης της υποομάδας encoder και πολλαπλασιάζεται με έναν πίνακα που ονομάζεται time distribution layer. Αυτό βοηθάει στην ανακατασκευή των δεδομένων της υποομάδας της χρονοσειράς ώστε να έχει τις ίδιες διαστάσεις με αυτές στην είσοδο στην αρχη του μοντέλου. Προκύπτει και μία δεύτερη έξοδος από τα LSTM που δείχνει τι πληροφορία έχει περάσει από το ένα LSTM στο επόμενο.



(δομή LSTM DECODER, εικόνα από [24])

Τέλος, ο έλεγχος για τα ακραία στοιχεία επιτυγχάνεται με την σύγκριση των πραγματικών τιμών με τις τι ανακατασκευασμένες τιμές, δηλαδή το πόσο διαφέρουν μεταξύ τους. Όταν η διαφορά είναι μεγαλύτερη από κάποιο όριο που θα θέσουμε τότε αυτή η τιμή θα θεωρείται ακραία. Το όριο προκύπτει μετά το πέρας της φάσης της εκπαίδευσης. Κάθε υποομάδα που έχει χωριστεί η χρονοσειρά και εξετάζεται από τον LSTM enocoder και decoder έχει ένα ποσοστό διαφοράς μεταξύ των πραγματικών τιμών και αυτών που παράγει το πρόγραμμα. Το μέγιστο ποσοστό διαφοράς από όλες τις υποομάδες θα τεθεί ως όριο για το αν μια υποομάδα θεωρείται ανώμαλη.

Η παρακάτω εικόνα δείχνει όλη την δομή και λειτουργία του LSTM AUTOENCODER



(Δομή LSTM AUTOENCODER, εικόνα από [24])

ΚΕΦΑΛΑΙΟ 6 ΣΥΝΟΛΑ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΓΙΑ ΤΗΝ ΕΠΙΛΟΓΗ ΜΕΘΟΔΩΝ ΕΝΤΟΠΙΣΜΟΥ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ

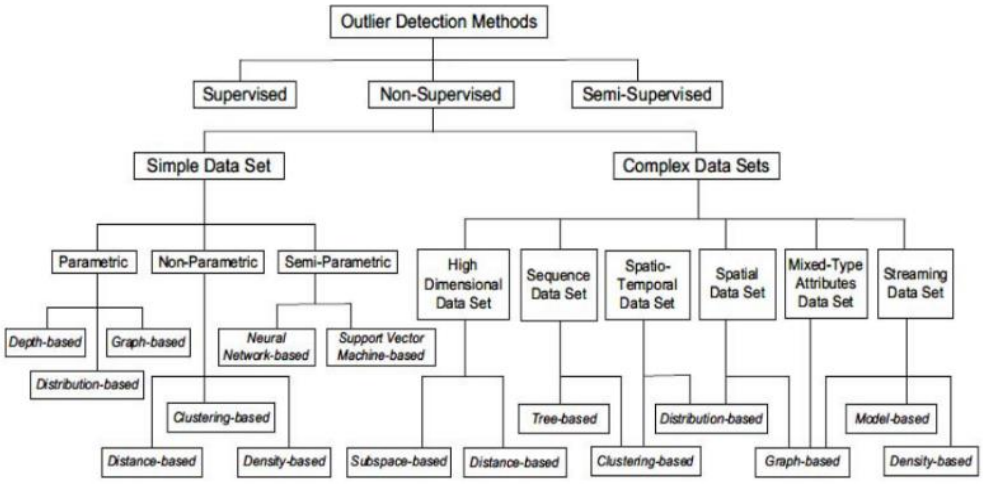
(πληροφορίες στο κεφάλαιο 6 από 8, 9, 21, 22)

Έχοντας αναλύσει τον ορισμό των ακραίων τιμών ή αλλιώς ανωμαλιών στα δεδομένα, τις κατηγορίες τους, τις διάφορες μεθόδους για τον εντοπισμό τους. Τους αλγορίθμους που θα χρησιμοποιηθούν από το πρόγραμμα μας που θα ανήκουν στην κάθε μέθοδο. Θα προσπαθήσουμε να συνδέσουμε όλα όσα έχουν αναφερθεί προηγουμένως. Βάση κάποιον χαρακτηριστικών του συνόλου δεδομένων θα διαλέγουμε την κατάλληλη μέθοδο και αλγόριθμο για την ανίχνευση ακραίων τιμών. Το πρόγραμμα μας ανάλογα των χαρακτηριστικών των συνόλων δεδομένων που θα αναφερθούν σε αυτό το κεφάλαιο και είναι σημαντικά για επιλογή συγκεκριμένων αλγορίθμων και για κάθε μέθοδο εντοπισμού, θα κατατάξει το σύνολο στην κάθε ομάδα. Έπειτα θα διαλέγει τους κατάλληλους αλγορίθμους που αντιστοιχούν σε αυτήν την ομάδα βάση της μεθόδου εντοπισμού που ακολουθούν και βάση την αποτελεσματικότητα τους σε συγκεκριμένη κατηγορία ανωμαλιών, δηλαδή κάποιοι είναι ισχυροί σε τοπικές ανωμαλίες άλλοι σε καθολικές κ.ο.κ..

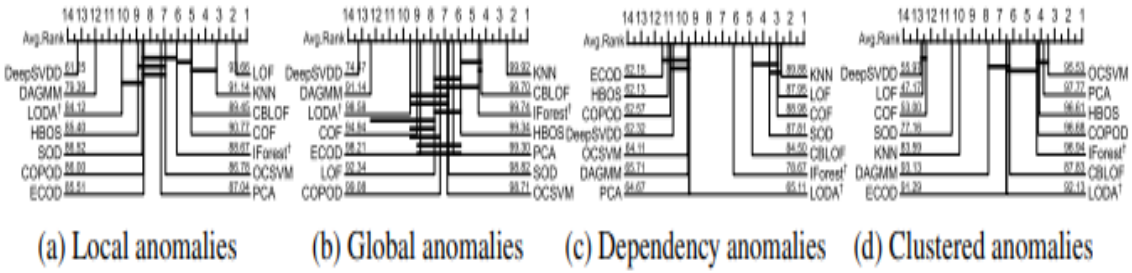
ENSEMBLE TECHINQUES KAI VOTING CLASSIFIER

Για κάθε ομάδα θα αντιστοιχούν τρεις αλγόριθμοι οι οποίοι θα βασίζονται σε διαφορετικές μεθόδους που θα είναι κατάλληλες για την συγκεκριμένη ομάδα δεδομένων. Οι αλγόριθμοι κάθε ομάδας εκτός της διαφοράς τους ως προς την μέθοδο και την λειτουργία έχουν διαφορετικά επίπεδα αποτελεσματικότητας σε συγκεκριμένες κατηγορίες αλγορίθμων. Αυτό γίνεται σε περίπτωση που κανένας από τους αλγορίθμους δεν έχει το επιθυμητό roc score (θέτοντας ένα όριο στην συνάρτηση) τότε θα καλείται ένας voting classifier που θα συνδυάζει τα αποτελέσματα των τριών αλγορίθμων. Αυτό έχει ως αποτέλεσμα ο voting classifier να είναι πιο αποτελεσματικός αφού συνδυάζει αλγορίθμους που είναι διαφορετικοί στην λειτουργία τους (σημαντικό όταν συνδυάζουμε αλγορίθμους με ensemble techniques) και είναι αποτελεσματικοί σε ανίχνευση διαφορετικών κατηγοριών ακραίων τιμών. Κάνοντας το μοντέλο μας να είναι ισχυρό στις προβλέψεις σε διάφορα προβλήματα ανίχνευσης ανωμαλιών. Γενικότερα στην ανίχνευση απέναντι σε οποιοδήποτε είδος ακραίας τιμής ανιχνεύοντας μεγαλύτερο εύρος ανωμαλιών και δίνοντας μεγαλύτερη ευελιξία στο πρόγραμμά μας.

Κατά κύριο λόγο η λειτουργία της συνάρτησης του προγράμματος είναι επηρεασμένη από δύο άρθρα. Το πρώτο [9] πάνω στη βιβλιοθήκη που χρησιμοποιούμε pyod για τον εντοπισμό ανωμαλιών αναφέροντας τις επιδόσεις κάθε μοντέλου βάση του roc score για κάθε είδος ανωμαλίας (καθολική ανωμαλία (global outlier), τοπική ανωμαλία (local anomaly), συμφραζόμενα ακραίες τιμές (Contextual-conditional outliers) (dependency στην φωτογραφία), ανωμαλίες συστάδων (Collective Outliers) (clustered στην φωτογραφία)**)** σε 30 datasets και το δεύτερο [8] έχει να κάνει με τις μεθόδους που επιλέγουμε ανάλογα με χαρακτηριστικά από το σύνολο των δεδομένων που φαίνεται από το σχήμα παρακάτω.



(επιλογή αλγορίθμου βάση των χαρακτηριστικών του συνόλου δεδομένων,εικόνα από[8])



(μέσος όρος επιδώσεων αλγορίθμων μη επίβλεψης στα σύνολα δεδομένων της βιβλιοθήκης PYOD, εικόνα από[9])

(Υποκεφάλαιο 6.1 ΑΝΑΛΥΣΗ ΤΩΝ ΟΜΑΔΩΝ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΣΥΝΑΡΤΗΣΗΣ)

Οι αλγόριθμοι που επιλέχθηκαν είναι όλοι με προσέγγιση με μη επίβλεψη (unsupervised learning) επειδή είναι πιο ρεαλιστικό σενάριο τα περισσότερα προβλήματα ανίχνευσης ανωμαλιών να μην έχουν labeled δεδομένα ώστε να χρησιμοποιηθούν supervised προσεγγίσεις. Η συνάρτηση κάνει ότι φαίνεται στο σχήμα στην φωτογραφία [8] προσπαθεί να χωρίσει το dataset σε απλό ή σύνθετο και έπειτα σε άλλες υποομάδες. Στο απλό dataset υπάρχουν δύο ομάδες για παραμετρικά και μη παραμετρικά σύνολα ή αλλιώς σύνολα που ακολουθούν κανονική κατανομή ή όχι. Στα σύνθετα έχουμε είτε πολλών διαστάσεων (βάση του προγράμματος, σύνολο δεδομένων άνω των 15 διαστάσεων) είτε κανονικών (βάση του προγράμματος, σύνολο δεδομένων μεγαλύτερο των 5 διαστάσεων και μικρότερο των 15 και ο πολλαπλασιασμός των διαστάσεων επί των στοιχείων του θα έχει αριθμό μεγαλύτερο του 18000). Υπάρχει μία

ξεχωριστή κατηγορία για τα χρονικά μεταβαλλόμενα (time series) η οποία δεν θα έχει κάποια υποομάδα επειδή κατά κύριο λόγο η δομή της εργασίας απευθύνεται σε στατικά δεδομένα και δεν θα εμβαθύνουμε τόσο στην ανάλυση χρονοσειρών.

Θεωρούμε ένα dataset ως απλό στο συγκεκριμένο πρόγραμμα αν έχει μικρό αριθμό χαρακτηριστικών ή αλλιώς διαστάσεων και εάν έχει μικρό αριθμό εγγραφών. Στις διαστάσεις έχουμε θέσει ως όριο τις 15, αν τα δεδομένα μας έχουν περισσότερες τότε θεωρούνται σύνθετα και θα μπουν στην υποομάδα των multidimensional, εάν έχουν λιγότερες αλλά ο αριθμός των εγγραφών τους επί των αριθμό των διαστάσεων είναι μεγαλύτερος των 18000 τότε θα μπουν στην υποομάδα σύνθετων αλλά κανονικών διαστάσεων. Οπότε προκύπτουν πέντε ομάδες που μπορεί να κατηγοριοποιηθεί ένα σύνολο δεδομένων.

ΧΡΗΣΗ ΣΥΝΑΡΤΗΣΕΩΝ ΓΙΑ ΤΗΝ ΚΑΤΗΓΟΡΙΟΠΟΙΗΣΗ

Χρησιμοποιούνται δύο συναρτήσεις η μία για να ελέγχει εάν το σύνολο δεδομένων ακολουθεί κανονική κατανομή. Μέσω της βιβλιοθήκης stats υπολογίζει εάν ένα σύνολο μίας διάστασης ακολουθεί κανονική κατανομή μέσω του Kolmogorov-Smirnov test και της τιμής όπου a αν το αποτέλεσμα του τεστ είναι μικρότερο από 0,005 τότε τα δεδομένα ακολουθούν κανονική κατανομή. Σε multivariate δεδομένα δηλαδή πολλών μεταβλητών δεν είναι εύκολο να βρεθεί με ακρίβεια αν ακολουθούν κανονική κατανομή μέσω κάποιων τεστ παρά εμπειρικά και μέσω οπτικοποίησης των δεδομένων κάτι που δεν μπορεί να γίνει αυτοματοποιημένα μέσω μιας συνάρτησης.

```
def normal_distribution(dataset, a=0.05):

    distribution = stats.norm

    _,value = stats.kstest(dataset, distribution.cdf)

    if value < a :
        return True
    else:
        return False
```

(συνάρτηση για τον έλεγχο κανονικής κατανομής)

Η δεύτερη κάνει έλεγχο αν είναι χρονικά μεταβαλλόμενα τα δεδομένα (time series). Η συνάρτηση δέχεται σαν όρισμα το σύνολο δεδομένων και εξετάζει μέσω της συνάρτησης isinstance(dataframe.index, (pd.DatetimeIndex, pd.PeriodIndex)) εξετάζει εάν ο δείκτης των γραμμών του πίνακα είναι είτε τύπου είτε datetime είτε period, δηλαδή είναι ημερομηνία η χρονική περίοδο. Σε περίπτωση που ισχύει ένα από τα δύο γυρνάει true αλλιώς false. Αυτή η Boolean τιμή δηλαδή true or false περνάει στην μεταβλητή time_index και αυτή επιστρέφεται από την συνάρτηση για να ολοκληρώσει τον έλεγχο και να κατατάξει το σύνολο δεδομένων σε ανάλογη ομάδα.

```
def time_series_check(dataframe):

    time_index = isinstance(dataframe.index, (pd.DatetimeIndex, pd.PeriodIndex))

    return time_index
```

(συνάρτηση για έλεγχο χρονικά μεταβαλλόμενων δεδομένων)

Αφού γίνουν οι έλεγχοι από το πρόγραμμα και το σύνολο κατανεμηθεί σε μία κατηγορία τότε θα χρησιμοποιούνται οι κατάλληλοι αλγόριθμοι για την συγκεκριμένη κατηγορία δεδομένων. Οι υπόλοιπες κατηγορίες δεδομένων προκύπτουν από τους περιορισμούς που αναφέρθηκαν στις προηγούμενες παραγράφους. Όλοι οι έλεγχοι που γίνονται για την κατηγοριοποίηση είναι απλοί στην εφαρμογή τους και δεν είναι κοστοβόροι χρονικά. Στην επίδοση ενός προγράμματος είναι σημαντική η χρονική πολυπλοκότητα και δεν θα ήταν αποτελεσματικό για την ανάλυση των δεδομένων να επιβαρυνόταν το πρόγραμμα μας από σύνθετες συναρτήσεις. Η κατηγοριοποίηση γίνεται από ουσιαστικές και απλές συναρτήσεις ή thresholds δηλαδή κατώφλια ή αλλιώς όρια που είναι σημαντικά για να κατατάξουμε τα δεδομένα με ελάχιστο χρονικό κόστος.

(Υποκεφαλαιο 6.2 ΑΝΑΛΥΣΗ ΤΩΝ ΟΜΑΔΩΝ ΚΑΙ ΕΠΙΛΟΓΗ ΑΛΓΟΡΙΘΜΩΝ)

Η πρώτη ομάδα ανήκει στα απλά σύνολα δεδομένων είναι για την περίπτωση που το σύνολο δεδομένων είναι μίας μεταβλητής και ακολουθεί κανονική κατανομή. Οι προτεινόμενοι μέθοδοι ανίχνευσης ακραίων τιμών για αυτήν την κατηγορία είναι graph-based, depth-based,distribution-based. Ο αλγόριθμος που βασίζεται σε μέθοδο graph-based είναι ο isolation forest. Ο local outlier factor (lof) βασίζεται σε μέθοδο depth-based (μπορεί να θεωρηθεί και σαν density-based) Ο HBOS ανήκει σε μέθοδο

distribution-based, δηλαδή μέθοδο που θεωρεί ότι τα δεδομένα ακολουθούν μια συγκεκριμένη κατανομή που στην περίπτωση μας και στου αλγορίθμου αυτή είναι η κανονική κατανομή.

```
parametric= {'Local Outlier Factor ': LOF(
        contamination=outliers_fraction),'Isolation Forest': IForest(contamination=outliers_fraction,
        random_state=random_state), 'Histogram-base Outlier Detection': HBOS(contamination=outliers_fraction)}
```

(ομάδα αλγορίθμων για τα σύνολα δεδομένων που ακολουθούν κανονική κατανομή)

Η δεύτερη ομάδα ανήκει και αυτή στα απλά σύνολα δεδομένων αλλά σε δεδομένα που δεν ακολουθούν κανονική κατανομή ή γενικά κάποια συγκεκριμένη κατανομή. Οι προτεινόμενοι μέθοδοι είναι clustering-based, distance-based, densinty-based. Ο Cluster-based Local Outlier Factor (CBLOF) ανήκει στην πρώτη κατηγορία, ο k-nearest-neighbor (knn) ανήκει στην distance-based μέθοδο και ο local outlier factor (cblof) ανήκει στην density-based. Όπως είχε αναφερθεί σε προηγούμενη παράγραφο και ισχύει και για την παραπάνω ομάδα και για τις επόμενες, οι αλγόριθμοι που διαλέγουμε εκτός ότι ανήκουν στις προτεινόμενες μεθόδους προσπαθούμε να είναι στους βέλτιστους όσον αφορά το roc score σε διαφορετικά είδη ακραίων τιμών (από τα αποτελέσματα στην εικόνα από [9]). Από [9] συμπεραίνουμε ότι στατιστικά κανένας αλγόριθμος δεν μπορεί να έχει υψηλά ποσοστά επιτυχίας σε όλες τις κατηγορίες αλγορίθμων οπότε θα ήταν καλύτερο σαν ιδέα ο συνδυασμός αλγορίθμων που έχουν υψηλά ποσοστά σε διαφορετικές κατηγορίες. Στην συγκεκριμένη ομάδα έχουμε τον knn που έχει το μεγαλύτερο σκορ σε καθολικές ανωμαλίες (global outliers) και τον lof που έχει το μεγαλύτερο σκορ σε τοπικές (local outliers) και το δεύτερο καλύτερο σκορ (Contextual-conditional outliers)).

```
non_parametric={'K Nearest Neighbors': KNN(contamination=outliers_fraction), 'Cluster-based Local Outlier Factor':
    CBLOF(contamination=outliers_fraction,check_estimator=False, random_state=random_state),'Local Outlier Factor ':
            LOF(contamination=outliers_fraction)}
```

(ομάδα αλγορίθμων για σύνολα δεδομένων που ανήκουν στην κατηγορία non_parametric)

Η Τρίτη ομάδα είναι για τα σύνθετα σύνολα δεδομένων πολλών διαστάσεων. Οι προτεινόμενοι μέθοδοι είναι distance-based και subspace-based. Στην συγκεκριμένη περίπτωση δεν θα δωθεί τόσο μεγάλη βάση στις μεθόδους αλλά θα χρησιμοποιηθούν αλγόριθμοι που δεν επηρεάζονται από δεδομένα πολλών διαστάσεων(curse of dimensionality) και δεν είναι μεγάλης πολυπλοκότητας. Όσο μεγαλώνουν οι διαστάσεις και ο αριθμός των δεδομένων τόσο πιο κοστοβόρο είναι για το πρόγραμμα μας μεγάλες πολυπλοκότητες αλγορίθμων. Γενικά μέθοδοι όπως distance-base και clustering-base έχουν μεγάλη πολυπλοκότητα. Ο πρώτος αλγόριθμος που θα

χρησιμοποιήσουμε θα είναι ο ECOD όπως είχαμε αναφέρει στην ανάλυση του είναι αρκετά αποτελεσματικός και γρήγορος (πολυπλοκότητα (O(nd) όπου n αριθμός στοιχείων και d αριθμός διαστάσεων). Εφόσον δεν χρειάζεται βελτιστοποίηση στις παραμέτρους του και για την εύρεση των ανωμαλιών δεν χρειάζονται κοστοβόροι υπολογισμοί κάνοντας τον καλή επιλογή για σύνθετα και μεγάλα σύνολα δεδομένων με πολλές διαστάσεις. Ο iforest χρησιμοποιείται για τον ίδιο λόγο, δηλαδή της χαμηλής του πολυπλοκότητας και ότι δεν επηρεάζεται η απόδοση του από τις πολλές διαστάσεις. Είναι ο αλγόριθμος που θα βρίσκεται στις πιο πολλές ομάδες επειδή είναι πολύ ισχυρός σε διάφορα είδη συνόλων δεδομένων και αποτελεσματικός σε διάφορες κατηγορίες ακραίων τιμών. Ο τρίτος που θα χρησιμοποιηθεί είναι ο ABOD, ο τρόπος λειτουργίας του τον κάνει να μην επηρεάζεται η επίδοση του από τις πολλές διαστάσεις, υπολογίζοντας το φάσμα της γωνίας κάθε σημείου δεν υπολογίζει αποστάσεις που επηρεάζονται από τις πολλές διαστάσεις. Η παραλλαγή του αλγορίθμου που χρησιμοποιούμε και είναι πιο γρήγορη και δεν χρειάζεται βελτιστοποίηση στις παραμέτρους, κάτι που είναι πολύ χρήσιμο χρονικά αλλά και ειδικά σε περιπτώσεις με μέθοδο χωρίς επίβλεψη.

```
high_dim={'Angle-based Outlier Detector':ABOD(contamination=outliers_fraction),'Isolation Forest':
    IForest(contamination=outliers_fraction,random_state=random_state),'Empirical Cumulative Distribution Functions':
    ECOD(contamination=outliers_fraction)}
```

(ομάδα αλγορίθμων για τα σύνολα δεδομένων που ανήκουν στη κατηγορία high_dim)

Η τέταρτη ομάδα είναι για σύνθετα δεδομένα όχι πολλών διαστάσεων. Οι προτεινόμενοι μέθοδοι είναι distribution-based, graph-based και θα χρησιμοποιήσουμε και μεθόδους συσταδοποίησης και συγκεκριμένα το αλγόριθμο CBLOF. Ο αλγόριθμος iforest θα χρησιμοποιηθεί για graph-based και ο HBOS για distribution-based.

```
complex_non_high_dim={'Isolation Forest': IForest(contamination=outliers_fraction, random_state=random_state),
                      'Cluster-based Local Outlier Factor':CBLOF(contamination=outliers_fraction,
                        check_estimator=False, random_state=random_state),
                    'Histogram-base Outlier Detection': HBOS(contamination=outliers_fraction)}
```

(ομάδα αλγορίθμων για σύνολο δεδομένων που ανήκουν στην κατηγορία high_dim)

Η τελευταία ομάδα είναι ξεχωριστή και αφορά τα χρονικά μεταβαλλόμενα δεδομένα. Η συγκεκριμένη κατηγορία θα μπορούσε να έχει και άλλες υποομάδες αλλά όπως έχουμε αναφέρει κατά κύριο λόγο η εργασία αφορά τα στατικά δεδομένα και δεν θα εμβαθύνουμε τόσο σε αυτήν την κατηγορία δεδομένων, επίσης η βιβλιοθήκη που χρησιμοποιούμε για τους αλγορίθμους απευθύνεται σε στατικά δεδομένα. Θα παρουσιάσουμε ενδεικτικά κάποιες μεθόδους και αλγορίθμους και για την καλύτερη αντιμετώπιση του συγκεκριμένου προβλήματος θα χρησιμοποιήσουμε έναν αλγόριθμο

deep learning τον LSTM-autoencoder που είναι αποτελεσματικός στην ανίχνευση ανωμαλιών σε χρονοσειρές. Οι αλγόριθμοι που θα χρησιμοποιηθούν θα είναι ο OCSVM που θα βασίζεται σε μέθοδο model-based, ο iforest με μέθοδο graph-based και ο lof με μέθοδο density based.

```
time_series={'One-class SVM (OCSVM)': OCSVM(contamination=outliers_fraction),'Isolation Forest':
     IForest(contamination=outliers_fraction,random_state=random_state),'Local Outlier Factor ': LOF(
       contamination=outliers_fraction)}
```

(ομάδα αλγορίθμων για σύνολο δεδομένων που ανήκει στην κατηγορία time_series)

(Υποκεφαλαιο 6.3 ΑΝΑΛΥΣΗ ΣΥΝΑΡΤΗΣΗΣ ΕΚΤΕΛΕΣΗΣ ΑΛΓΟΡΙΘΜΩΝ)

Μέσα στην τελική συνάρτηση καλείται η συνάρτηση που θα εκτελούνται οι αλγόριθμοι των ομάδων που αναφέραμε προηγουμένως. Η συνάρτηση καλείται έπειτα από κάθε έλεγχο που γίνεται στο σύνολο των δεδομένων ώστε να καταταχθεί σε κάποια κατηγορία από όσες έχουμε ορίσει. Δημιουργήθηκε για να μην επαναλαμβάνονται ίδια σημεία μέσα στην τελική συνάρτηση και να βελτιωθεί οπτικά ο κώδικας, επειδή έπειτα από κάθε έλεγχο που κάναμε εκτελούνταν ακριβώς ο ίδιος κώδικας και το μόνο που άλλαζε ήταν η λίστα με τις ομάδες των αλγορίθμων. Αυτός είναι ο λόγος που το μόνο όρισμα που έχει η συνάρτηση είναι το όνομα για κάθε λίστα αλγορίθμων.

```
def algo_exec(list_name):
    i = 0
    max_f1score = 0
    algorithms = [key for key in list_name if '' in key]

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,random_state=random_state)
    test_scores = np.zeros([X_test.shape[0],3])
    roc_scores = []
    precision_scores = []
    time_complexity = []
    for clf_name, clf in list_name.items() :

        t0=time()
        clf.fit(X_train)
        test_scores_i = clf.decision_function(X_test)
        test_scores[:, i] =test_scores_i
        t1 = time()
        duration = round(t1 - t0, ndigits=4)

        roc = round(roc_auc_score(y_test, test_scores_i), ndigits=4)
        precision = round(precision_n_scores(y_test, test_scores_i), ndigits=4)
        roc_scores.append(roc)
        precision_scores.append(precision)
        time_complexity.append(duration)
        print('{clf_name} ROC:{roc}, precision @ rank n:{precision}, '
        'execution time: {duration}s'.format(
        clf_name=clf_name, roc=roc, precision=precision, duration=duration))
        i += 1
        if roc>max_f1score:
             max_f1score=roc
```

```
  if max_f1score<0.8:

      comb_by_average = average(test_scores)
      evaluate_print('Combination by Average', y_test,comb_by_average)
      comb_by_maximization = maximization(test_scores)
      evaluate_print('Combination by Maximization', y_test, comb_by_maximization)
      comb_by_aom = aom(test_scores, 3)
      evaluate_print('Combination by AOM', y_test, comb_by_aom)
      comb_by_moa = moa(test_scores, 3)
      evaluate_print('Combination by MOA', y_test, comb_by_moa)


  plt.bar(range(len(roc_scores)), roc_scores, tick_label=algorithms,width=0.6, color=['red', 'green', '
  plt.xticks(fontsize=7)
  plt.xlabel('algorithms')
  plt.ylabel('roc score')
  plt.show()


  plt.bar(range(len(precision_scores)), precision_scores, tick_label=algorithms,width=0.6, color=['red'
  plt.xticks(fontsize=7)
  plt.xlabel('algorithms')
  plt.ylabel('presicion score')
  plt.show()

  plt.bar(range(len(time_complexity)), time_complexity, tick_label=algorithms,width=0.6, color=['red',
  plt.xticks(fontsize=7)
  plt.xlabel('algorithms')
  plt.ylabel('time_complexity')
  plt.show()
```

(συνάρτηση εκτέλεσης των αλγορίθμων)

Όπως εξηγήσαμε σαν όρισμα στην συνάρτηση έχουμε το όνομα της λίστας. Όταν θα ξεκινάει το for loop θα καλείται το πρώτο στοιχείο της λίστας που θα είναι ο αλγόριθμος που θα εκτελείται και έπειτα ο επόμενος μέχρι να ολοκληρωθεί η λίστα. Στις επόμενες δύο γραμμές αρχικοποιείται το i και το max_roc_score και τα δύο με την τιμή 0 εκτός του for loop. Στις επόμενες γραμμές γίνεται ο διαχωρισμός στα δεδομένα στο train και test στα X και Y αντίστοιχα το 70% των δεδομένων θα χρησιμοποιηθεί για το train και το υπόλοιπο 30% για το testing. Μετά αρχικοποιείται ο πίνακας

test_scores με όλα τα στοιχεία του να είναι 0. Σαν γραμμές θα έχει το πλήθος του τεστ δείγματος του συνόλου δεδομένων δηλαδή το 30% του συνολικού δείγματος (το πλήθος δίνεται από το X_test.shape[0]) που θα είναι μεταβλητό ανάλογα του πλήθος του συνόλου δεδομένων. Το πλήθος των στηλών θα είναι πάντα 3 αφού τόσοι είναι οι αλγόριθμοι που καλούνται για κάθε σύνολο και κάθε στήλη θα έχει τα αποτελέσματα για κάθε στοιχείο. Σε περίπτωση που δεν χρησιμοποιούσαμε ensemble τεχνικές δεν θα χρειαζόταν αυτός ο πίνακας απλά θα εμφανίζαμε το test_score_i που είναι το αποτέλεσμα για κάθε αλγόριθμο

Το i θα αυξάνεται όταν τελειώνει το train και το testing των αλγορίθμων για να αποθηκεύονται τα αποτελέσματα στους αντίστοιχους πίνακες και έπειτα θα αυξάνεται η τιμή του κατά 1, με λίγα λόγια λειτουργεί σαν δείκτης για τους πίνακες ώστε να αποθηκεύονται οι τιμές στις σωστές θέσεις. Το max_roc_score θα υπάρχει για να αποθηκεύεται η μέγιστη τιμή roc και έπειτα να γίνεται ο έλεγχος εκτός for loop αν αυτή θα ξεπερνά το όριο που έχουμε θέσει ώστε να μην χρειαστεί να χρησιμοποιήσουμε ensemble methods. Ξεκινώντας το for loop το όρισμα που δώσαμε στην συνάρτηση περνάει σαν το όνομα της λίστα που θα τρέχει το for loop για να διαβάζει τους αλγορίθμους. Οι μεταβλητές t0, t1 μέσω της συνάρτησης time() από την βιβλιοθήκη time μετράνε την διάρκεια που χρειάζεται για την εκτέλεση ο κάθε αλγόριθμος της λίστας. Η t0 πριν την αρχή εκτέλεσης δηλώνει το ξεκίνημα και η t1 αμέσως μετά το τέλος την λήξη. Στο τέλος αφαιρούμε την t0 από την t1 και την αποθηκεύουμε στην μεταβλητή duration. Ανάμεσα στο t0 και t1 γίνεται πρώτα η εκπαίδευση του μοντέλου στα δεδομένα για εκπαίδευση και έπειτα στην μεταβλητή test_scores_i αποθηκεύονται τα αποτελέσματα του αλγορίθμου στα δεδομένα που είναι για το testing και έπειτα η μεταβλητή αποθηκεύεται σε πίνακα test_scores όπου από εκεί θα πάρουμε τα σκορ των μοντέλων σε περίπτωση που χρησιμοποιήσουμε ensemble method. Ο αλγόριθμος εκπαιδεύεται σε διαφορετική ομάδα δεδομένων και αξιολογεί την επίδοση του σε διαφορετική. Αυτό όπως είναι γνωστό γίνεται για να μην απομνημονεύσει το μοντέλο τα δεδομένα αλλά να μάθει να κάνει προβλέψεις σε νέα άγνωστα δεδομένα και να γενικοποιήσει τις προβλέψεις του. Ο διαχωρισμός των δεδομένων γίνεται εκτός αυτής της συνάρτησης και εκτός της μεγάλης συνάρτησης, αλλά στο τελικό πρόγραμμα όπου θα αναφερθεί παρακάτω.
Αφού έχει τελειώσει ο αλγόριθμος αποθηκεύονται στις μεταβλητές roc και precision τα roc σκορ και το precision του αλγορίθμου στην αντίστοιχα από τις παρακάτω βιβλιοθήκες.

```
from pyod.utils.utility import precision_n_scores
from sklearn.metrics import roc_auc_score
```

(δήλωση συναρτήσεων για την χρήση και τον υπολογισμό του precision και του roc)

Στη συνέχεια εμφανίζουμε τα αποτελέσματα, αυξάνουμε την τιμή στην μεταβλητή i για να αποθηκευτεί καινούριο σκορ για τον επόμενο αλγόριθμο και αφού τελειώσει το for loop γίνεται ο έλεγχος για το roc σκορ, αν για κανένα από τους 3 αλγορίθμους δεν είναι μεγαλύτερο του 0,8 χρησιμοποιούμε ensemble methods.

Μέσω της βιβλιοθήκης pyod.models.combination γίνεται χρήση των ensemble methods. Στη συνάρτηση χρησιμοποιούμε όλες τις διαφορετικές τεχνικές ώστε να παρατηρήσουμε τα αποτελέσματα τους και να τις αναλύσουμε. Το AOM (Average of Maximums) για κάθε αλγόριθμο υπολογίζει την μέση τιμή των μέγιστων σκορ ακραίων τιμών. Για κάθε σημείο ξεχωριστά υπολογίζει την μέση τιμή από τους διαφορετικούς αλγορίθμους και έπειτα επιλέγει την μέγιστη μέση βαθμολογία. Το MOA (Maximum of Averages) υπολογίζει την μέγιστη μέση τιμή βαθμολογίας. Για κάθε σημείο δεδομένων σε κάθε μοντέλο υπολογίζει την μέση τιμή για κάθε σημείο και στο τέλος διαλέγει την μέγιστη μέση βαθμολογία. Το Average υπολογίζει την μέση τιμή σκορ ανωμαλίας για κάθε σημείο δεδομένων από όλα τα μοντέλα. Κάθε αλγόριθμος έχει το ίδιο 'βάρος' ή αλλιώς σημασία στον υπολογισμό. Το maximization επιλέγει το μέγιστο outlier score από όλους τους αλγόριθμους για κάθε μοντέλο και κρατάει αυτό για κάθε σημείο στην τελική απόφαση. Το median υπολογίζει την διάμεσο από τα σκορ ανωμαλιών για κάθε σημείο σε όλους τους αλγορίθμους. Η διαφορά του με το Average είναι ότι δεν επηρεάζεται τόσο από τις μεμονωμένες τιμές κάποιων αλγορίθμων αλλά δίνεται περισσότερη έμφαση στο συνολικό αποτέλεσμα των αλγορίθμων.

(Υποκεφάλαιο 6.4 Η ΤΕΛΙΚΗ ΣΥΝΑΡΤΗΣΗ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ)

Έχοντας αναλύσει όλες τις συναρτήσεις που γίνονται για τους ελέγχους και την εκτέλεση των μοντέλων, τις ομάδες αλγορίθμων και τους περιορισμούς για την κατηγοριοποίηση των συνόλων δεδομένων στις αντίστοιχες ομάδες μπορούμε να αναλύσουμε την τελική συνάρτηση που αποτυπώνει όλη την λογική του προγράμματος μας. Έχοντας χρησιμοποιήσει συναρτήσεις για την εκτέλεση και τον έλεγχο μέσα στην τελική συνάρτηση η δομή της γίνεται πολύ απλή και κατανοητή. Η συνάρτηση σαν όρισμα παίρνει το σύνολο δεδομένων. Ο πρώτος έλεγχος γίνεται για να διαπιστωθεί αν το σύνολο δεδομένων είναι χρονοσειρά μέσω της συνάρτησης που αναφέραμε στα προηγούμενα κεφάλαιο και αν ισχύει καλούμε την συνάρτηση που εκτελεί τους αλγορίθμους με όρισμα το όνομα της λίστας που έχει τους κατάλληλους αλγορίθμους για την συγκεκριμένη ομάδα. Σε όλους τους υπόλοιπους ελέγχους θα έχουμε σαν προϋπόθεση να μην είναι χρονοσειρά ώστε να ισχύει η συνθήκη του if. Έπειτα με το col=df.shape [1] παίρνουμε τον αριθμό των διαστάσεων του συνόλου και αν είναι μεγαλύτερο από 15 τότε θα καταταχθεί στην ομάδα των πολλών διαστάσεων και θα καλεστεί ο αλγόριθμος με το αντίστοιχο όνομα της λίστας σαν όρισμα. Σε περίπτωση που δεν ισχύει γίνεται έλεγχος αν ο αριθμός των στοιχείων επί τον αριθμό των διαστάσεων είναι μεγαλύτερο του 180000 τότε θα ανήκει στην σύνθετη ομάδα δεδομένων αλλά όχι πολλών διαστάσεων. Τον αριθμό των δεδομένων τον γνωρίζουμε από το n=df.shape[0]. Αν δεν ισχύουν οι παραπάνω έλεγχοι τότε αν ο αριθμός διαστάσεων ισούται με 1 και η συνάρτηση δείξει ότι το σύνολο ακολουθεί κανονική κατανομή θα κάνουμε τις αντίστοιχες ενέργειες για την εκτέλεση της ομάδας των συγκεκριμένων αλγορίθμων. Τέλος αν δεν ισχύει τίποτα από τα παραπάνω τότε το σύνολο κατανέμεται στην ομάδα non_parametric.

```
def outlier_detection_systems(df):

    #ελενχος για time series
    if time_series_dataset(df):
        print("kalw algorithmo gia time series")
        algo_exec(time_series)



    #ελενχος για διαστασεις αν ειναι πανω απο 15 highdimensional
    col=df.shape[1]
    if col >= 15  and not time_series_dataset(df):
        print("kalw algorithmo gia high_dimentional")
        algo_exec(high_dim)



    else:
        #ελενχος για complex dataset διαστασεις * αριθμος στοιχείων
        n=df.shape[0]
        if n*col >= 180000 and not time_series_dataset(df):
            print("kalw algorithmo gia  complex_non_high_dimestional")
            algo_exec(complex_non_high_dim)




        else:

               #ελενχος για κανονικη κατανομη
            if n== 1 and normal_distribution(df) and not time_series_dataset(df):
                print("kalw algorithmo gia  parametric")
                algo_exec(parametric)



     else:

        if  not time_series_dataset(df):
        #αν δεν είναι τίποτα απο τα παραπάνω τοτε καλώ αλγόριθμους για simple_dataset_non_parametric
           print("kalw algorithmo gia  non_parametric\n")
           algo_exec(non_parametric)
```

(τελική συνάρτηση διαχωρισμού και κατηγοριοποίησης των συνόλων δεδομένων)

(Υποκεφάλαιο 6.5 ΤΕΛΙΚΟ ΠΡΟΓΡΑΜΜΑ)

Το τελικό πρόγραμμα περιέχει μία λίστα με δέκα σύνολα δεδομένων που θα εξετάσουμε μέσω της τελικής συνάρτηση. Ένα for loop θα τρέχει για κάθε στοιχείο της λίστας δηλαδή κάθε σύνολο και θα φορτώνεται στην μεταβλητή mat που έπειτα θα χωρίζεται στα χαρακτηριστικά με μεταβλητή X θα είναι τα ανεξάρτητα στοιχεία που

θα είναι input στα μοντέλα μας και την Y που θα είναι η εξαρτημένη μεταβλητή σαν το output. Έπειτα μετράμε το ποσοστό των outlier στα δεδομένα μας, όπως είχαμε αναφέρει επιλέχθηκαν τα συγκεκριμένα δεδομένα επειδή είναι labeled δηλαδή ξέρουμε στο συγκεκριμένο παράδειγμα πια είναι κανονικές τιμές και πια όχι στην στήλη των Y δηλαδή την εξαρτημένη μεταβλητή. Βάση αυτού, μπορούμε να υπολογίσουμε το ποσοστό των ακραίων τιμών στα δεδομένα που βοηθάει στη επίδοση των μοντέλων μας και φαίνεται στο outliers_fraction = np.count_nonzero(y) / len(y) δηλαδή στα Y τον αριθμό των τιμών διαφορετικών του 0 δηλαδή 1 (το 1 δείχνει ότι το στοιχείο είναι ανωμαλία) προς το συνολικό αριθμό Y και έπειτα υπολογίζει το ποσοστό outliers_percentage = round(outliers_fraction * 100, ndigits=4). Τέλος με την μεταβλητή X μέσω της βιβλιοθήκης pandas δημιουργούμε ένα dataframe και το περνάμε στην μεταβλητή df και έπειτα καλούμε την τελική συνάρτηση με όρισμα το df.

```
mat_file_list = ['arrhythmia.mat',
                 'cardio.mat',
                 'glass.mat',
                 'shuttle.mat',
                 'mammography.mat',
                 'thyroid.mat',
                 'wine.mat' ,
                 'lympho.mat',
                ]

random_state = np.random.RandomState(42)


for mat_file in mat_file_list:
    print("\n...εξέταση του σύνολου δεδομένων ", mat_file, '...')
    mat=loadmat(mat_file)
    X=mat['X']
    y=mat['y'].ravel()

    outliers_fraction = np.count_nonzero(y) / len(y)
    outliers_percentage = round(outliers_fraction * 100, ndigits=4)



    df=pd.DataFrame(X)

    #λίστα με τους αλγόριθμους που θα χρησιμοποιώ ανάλογα με την κατηγοριοποίηση που θα γινει για το συνολο δεδομένων απο την
    #συνάρτηση



    parametric= {'Local Outlier Factor ': LOF(
```

```
#λίστα με τους αλγόριθμους που θα χρησιμοποιώ ανάλογα με την κατηγοριοποίηση που θα γινει για το συνολο δεδομένων απο την
#συνάρτηση

parametric= {'Local Outlier Factor ': LOF(
        contamination=outliers_fraction),'Isolation Forest': IForest(contamination=outliers_fraction,
        random_state=random_state), 'Histogram-base Outlier Detection': HBOS(contamination=outliers_fraction)}

non_parametric={'K Nearest Neighbors': KNN(contamination=outliers_fraction), 'Cluster-based Local Outlier Factor':
    CBLOF(contamination=outliers_fraction,check_estimator=False, random_state=random_state),'Local Outlier Factor ':
                LOF(contamination=outliers_fraction)}


high_dim={'Angle-based Outlier Detector':ABOD(contamination=outliers_fraction),'Isolation Forest':
      IForest(contamination=outliers_fraction,random_state=random_state),'Empirical Cumulative Distribution Functions':
      ECOD(contamination=outliers_fraction)}

complex_non_high_dim={'Isolation Forest': IForest(contamination=outliers_fraction, random_state=random_state),
                      'Cluster-based Local Outlier Factor':CBLOF(contamination=outliers_fraction,
                        check_estimator=False, random_state=random_state),
                  'Histogram-base Outlier Detection': HBOS(contamination=outliers_fraction)}

time_series={'One-class SVM (OCSVM)': OCSVM(contamination=outliers_fraction),'Isolation Forest':
      IForest(contamination=outliers_fraction,random_state=random_state),'Local Outlier Factor ': LOF(
        contamination=outliers_fraction)}



outlier_detection_systems(df)
```

(τελικό πρόγραμμα χρήσης συναρτήσεων και έλεγχο των συνόλων δεδομένων)

(Υποκεφάλαιο 6.6 LSTM AUTOENCODER)

Όπως είχαμε αναφέρει για να καλύψουμε το μέρος της ανίχνευσης ακραίων τιμών σε χρονικά μεταβαλλόμενα δεδομένα ή αλλιώς χρονοσειρές θα εφαρμόσουμε ένα μοντέλο βαθιάς μάθησης (deep learning) LSTM autoencoder αποτελεσματικό σε τέτοιου είδους δεδομένα. Έχοντας αναλύσει το θεωρητικό κομμάτι της λειτουργίας του και της σύνθετης δομής του θα εξηγήσουμε πως εφαρμόζεται όλη αυτή η λογική πρακτικά μέσω κώδικα. Αρχικά θα εισάγουμε την βιβλιοθήκη tensorflow και μέσω αυτής το api keras ώστε να μπορούμε να δημιουργήσουμε και να εκπαιδεύσουμε το μοντέλο μας. Εφόσον έχουμε διαβάσει το σύνολο δεδομένων που θα επεξεργαστούμε, θα κανονικοποιούμε τα δεδομένα ώστε να εξισορροπήσουμε την επίδραση των μεταβλητών στο μοντέλο και να βελτιωθεί η απόδοση του αλγορίθμου μας. Έπειτα καλείται η συνάρτηση create_sequences που από το όνομα της καταλαβαίνουμε πως ο σκοπός της είναι να χωρίσει την χρονοσειρά σε ακολουθίες ώστε να τις επεξεργαστεί το μοντέλο μας, έχει σαν όρισμα το X και το look back που αρχικοποιείται με 16. Το πρώτο όρισμα είναι τα δεδομένα της χρονοσειράς και το δεύτερο είναι το μέγεθος από τις ακολουθίες που θα δημιουργηθούν. Μετά αρχικοποιούνται δύο λίστες που θα αποθηκεύονται στη πρώτη τα στοιχεία της ακολουθίας που θα είναι το input και στην δεύτερη η τιμή που προσπαθεί να προβλέψει η κάθε ακολουθία. Όταν ολοκληρώνεται η συνάρτηση επιστρέφει πίσω τις δύο λίστες.

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(df)

def create_sequences(X, look_back = 16):
    X_seq, y_seq = [], []
    for i in range(len(X) - look_back):
        X_seq.append(X[i: (i + look_back)])
        y_seq.append(X[i + look_back, -1:])

    return np.array(X_seq), np.array(y_seq)


X, y = create_sequences(X_scaled)
```

(συνάρτηση δημιουργίας ακολουθιών)

Στις επόμενες γραμμές δημιουργούμε ένα ακολουθιακό μοντέλο μέσω του Keras και μετά προσθέτουμε 128 LSTM layers στο μοντέλο και των αριθμό των ακολουθιών που χωρίστηκε και τον χαρακτηριστικών του input. Η βελτιστοποίηση θα γίνει με την μέθοδο ADAM και η loss function θα είναι η MAE (mean absolute error). Στην τελευταία γραμμή γίνεται η εκπαίδευση του μοντέλου όπου X είναι το input και y το είναι οι πραγματικές τιμές που προσπαθεί να προβλέψει η κάθε ακολουθία. Το epochs=100 είναι ο αριθμός που θα γίνει το forward και backward pass του συνόλου εκπαίδευσης στο νευρωνικό δίκτυο. Το batch size=64 είναι ο αριθμός των στοιχείων που θα γίνει βελτιστοποίηση των βαρών στο forward και backward pass. Το σύνολο των δεδομένων θα χωριστεί σε 80% για την εκπαίδευση και το 20% για επικύρωση των δεδομένων. Το callback δεν θα ήταν αναγκαίο να υπάρχει στο συγκεκριμένο μοντέλο,

χρησιμεύει ώστε να μην καθυστερεί το μοντέλο σε περίπτωση που δεν υπάρχει βελτίωση στην εκπαίδευση δηλαδή δεν μειωθεί η τιμή του loss function μετά από 6 epochs (κάνει ότι εξηγήσαμε προηγουμένως).

```
model = Sequential()
model.add(LSTM(units = 128, input_shape = (X.shape[1], X.shape[2])))
model.add(RepeatVector(X.shape[1]))
model.add(LSTM(128, return_sequences = True))
model.add(TimeDistributed(Dense(y.shape[-1])))
model.compile(optimizer = 'adam', loss = 'mae')



history = model.fit(X, y, epochs = 100, batch_size = 64, validation_split = 0.2,
                    callbacks=[keras.callbacks.EarlyStopping(monitor = 'val_loss', patience = 6, mode = 'min')], shuffle = False)
```

(εκπαίδευση LSTM AUTOENCODER)

Στο τελευταίο σημείο του κώδικα που γίνεται η ανίχνευση ανωμαλιών υπολογίζεται η MAE (loss function που αναφέραμε και προηγουμένως) μεταξύ των τιμών των ακολουθιών και των τιμών που παρήγαγε σαν έξοδο ο decoder (στην θεωρία του αλγορίθμου έχουν εξηγηθεί όλα αυτά αναλυτικότερα). Υπολογίζουμε το όριο που αν ξεπερνιέται θα θεωρείτε ανωμαλία που θα είναι το 95% του max(loss) (αναφέραμε στην θεωρία του αλγορίθμου πως προκύπτει).

```
reconstructed = model.predict(X)
loss = np.mean(np.abs(reconstructed - X), axis = 1)

outliers = []
threshold = max(loss) * .95
for i in range(len(loss)):
    if loss[i] >= threshold:
        outliers.append(df.iloc[i, -1])

print(outliers)
```

```
6/6 [==============================] - 1s 11ms/step
[56, 71, 58]
```

(ανίχνευση ανωμαλιών)

(Υποκεφάλαιο 6.7 ΑΠΟΤΕΛΕΣΜΑΤΑ ΠΡΟΓΡΑΜΜΑΤΟΣ ΚΑΙ ΕΠΙΔΟΣΗ ΑΛΓΟΡΙΘΜΩΝ)

Για τον έλεγχο της συνάρτησης και την επίδοση των αλγορίθμων χρησιμοποιήθηκαν 10 σύνολα δεδομένων όπου τα στοιχεία είναι επισημασμένα (labeled) ώστε να είναι πιο αποτελεσματική η εκπαίδευση των αλγορίθμων αλλά και να είναι εφικτή η επικύρωση των μεθόδων ανίχνευσης και να μπορούμε να βγάλουμε συμπεράσματα για την επίδοση των μοντέλων και την αποτελεσματικότητας της συνάρτησης. Τα σύνολα δεδομένων παρέχονται από την ODDS library [21] .

Θα παρουσιάσουμε τα αποτελέσματα των αλγορίθμων ως [22]προς το roc score την ακρίβεια (precision) και την χρονική πολυπλοκότητα (time complexity). Για να γίνουν κατανοητές οι παραπάνω έννοιες πρέπει να αναφερθεί ο confusion matrix που μέσω αυτού προκύπτουν οι προηγούμενες μετρήσεις εκτός της χρονικής πολυπλοκότητας που είναι απλά η διάρκεια που χρειάστηκε για να εκτελεστεί ο κάθε αλγόριθμος. Σε περιπτώσεις που το πρόβλημα είναι η δυαδική ταξινόμηση, κατάταξη των στοιχείων μεταξύ δύο κατηγοριών όπως το πρόβλημα που αντιμετωπίζουμε στην παρούσα εργασία δηλαδή εάν ένα στοιχείο είναι ακραία τιμή ή όχι. Γίνεται χρήση του confusion matrix για να διαπιστώσουμε πόσο καλή είναι η επίδοση του αλγορίθμου που χρησιμοποιούμε.

	actual positive	actual negative
predicted positive	TP	FP
predicted negative	FN	TN

$$\text{Recall} = \frac{TP}{TP+FN}$$

$$\text{Precision} = \frac{TP}{TP+FP}$$

$$\text{True Positive Rate} = \frac{TP}{TP+FN}$$

$$\text{False Positive Rate} = \frac{FP}{FP+TN}$$

(εικόνα από [22])

(αριστερά confusion matrix, δεξιά τύποι για το precision , recall, true positive rate, false positive rate)

Όπως φαίνεται από την εικόνα ο confusion matrix έχεις 4 κατηγορίες.
TRUE POSITIVE: στοιχεία που ορθός έχουν καταταχθεί ως θετικά (ως ακραίες τιμές στην περίπτωση μας).
FALSE POSITIVE: στοιχεία που λανθασμένα έχουν καταταχθεί ως θετικά (ως ακραίες τιμές στην περίπτωση μας).

TRUE NEGATIVE: στοιχεία που σωστά έχουν καταταχθεί ως αρνητικά (ως κανονικές τιμές στην περίπτωση μας).
FALSE NEGATIVE: στοιχεία που λανθασμένα έχουν καταταχθεί ως αρνητικά (ως κανονικές τιμές στην περίπτωση μας).

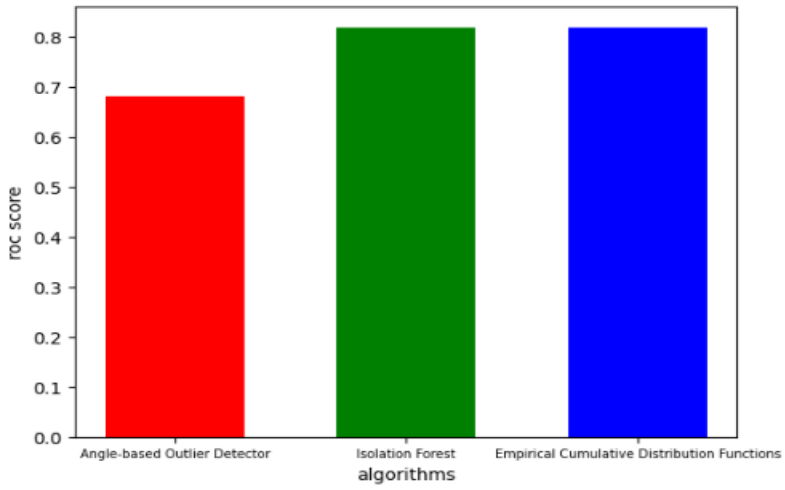
Μέσω τον παραπάνω κατηγοριών μπορεί να υπολογιστεί το roc score και το precision. Το πρώτο προκύπτει αν αναπαραστήσουμε χωρικά το (FPR) FALSE POSITIVE RATE που είναι το κλάσμα των FALSE POSITIVE διά το άθροισμα των FALSE POSITIVE συν TRUE NEGATIVE στον άξονα X και το (TPR) TRUE POSITIVE RATE που είναι το κλασμάτων TRUE POSITIVE διά TRUE POSITIVE συν το FALSE NEGATIVE στο άξονα Y και δημιουργείται μία καμπύλη που περιγράφει πόσο καλό είναι το μοντέλο στο να προβλέπει την θετική ομάδα (ακραία τιμή στην περίπτωση μας), όσο πιο ψηλά και αριστερά είναι αυτή η καμπύλη τόσο καλύτερη η επίδοση του αλγορίθμου. Το precision προκύπτει από το κλάσμα των TRUE POSITIVE προς το άθροισμα των TRUE POSITIVE συν FALSE POSITIVE περιγράφει την ακρίβεια των υπολογισμών των ακραίων τιμών στην περίπτωσή μας.

ΑΠΟΤΕΛΕΣΜΑΤΑ

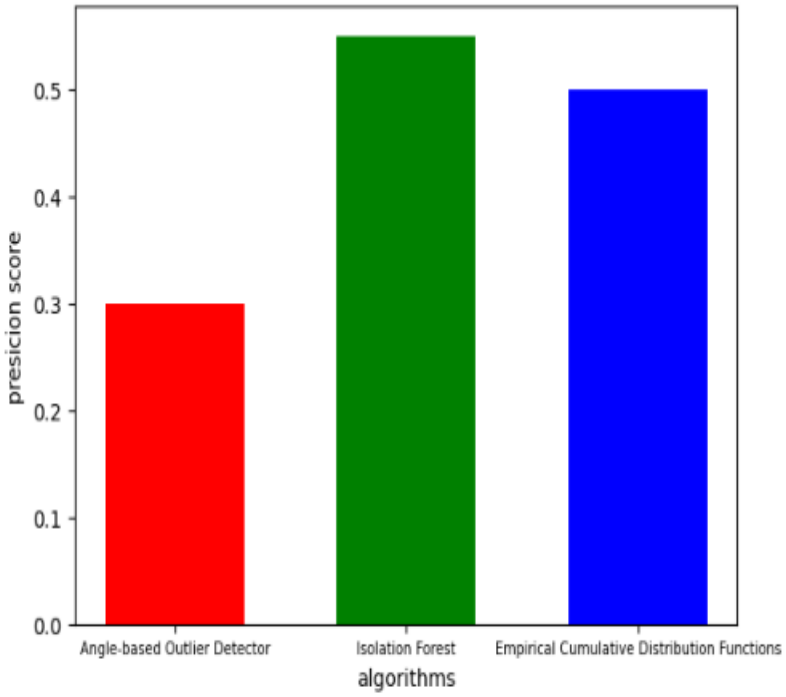
Εφόσον έχουμε εξηγήσει τους τρόπους αξιολόγησης της επίδοσης των αλγορίθμων μπορούμε να παρουσιάσουμε τα αποτελέσματα της συνάρτησης.

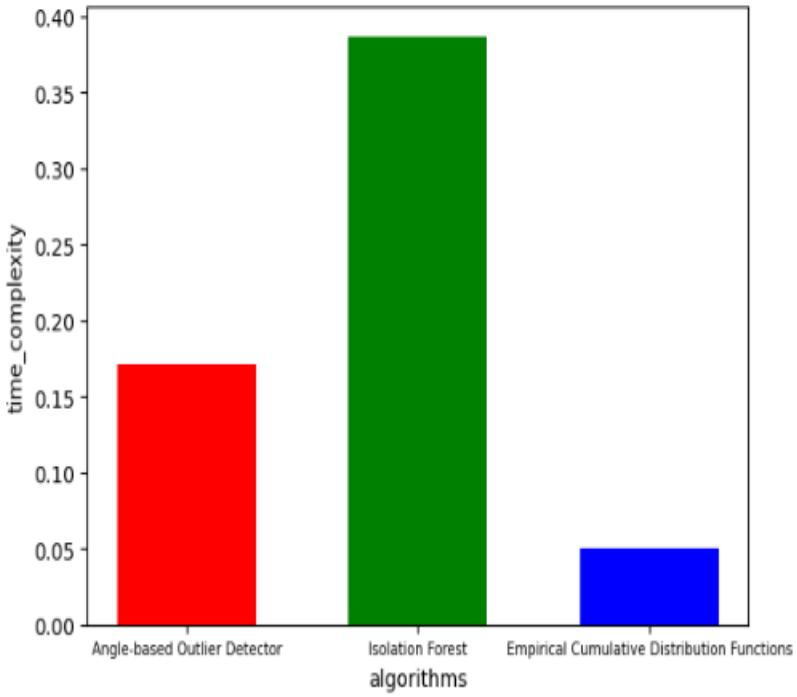
Το πρώτο σύνολο δεδομένων που εξετάζουμε έχει 452 στοιχεία και 274 διαστάσεις, και ποσοστό ακραίων τιμών 15% (66 ακραίες τιμές).
Στο συγκεκριμένο σύνολο καλύτερη επίδοση είχε ο IFOREST λόγω του roc score 0,819, precision 0,55, time complexity 0,3869 seconds.
Δεύτερη καλύτερη επίδοση ο ECDF λόγο roc score 0,8185, precision 0,5, time complexity 0,3869 seconds.
Χειρότερη επίδοση ο ABOD λόγο του roc score 0,6819, precision 0,3, time complexity 0,1715 seconds.

```
...εξέταση του σύνολου δεδομένων  arrhythmia.mat ...
kalw algorithmo gia high_dimentional
Angle-based Outlier Detector ROC:0.6819, precision @ rank n:0.3, execution time: 0.1715s
Isolation Forest ROC:0.819, precision @ rank n:0.55, execution time: 0.3869s
Empirical Cumulative Distribution Functions ROC:0.8185, precision @ rank n:0.5, execution time: 0.0504s
```



(αποτελέσματα roc_score συνόλου δεδομένων arrhythmia)

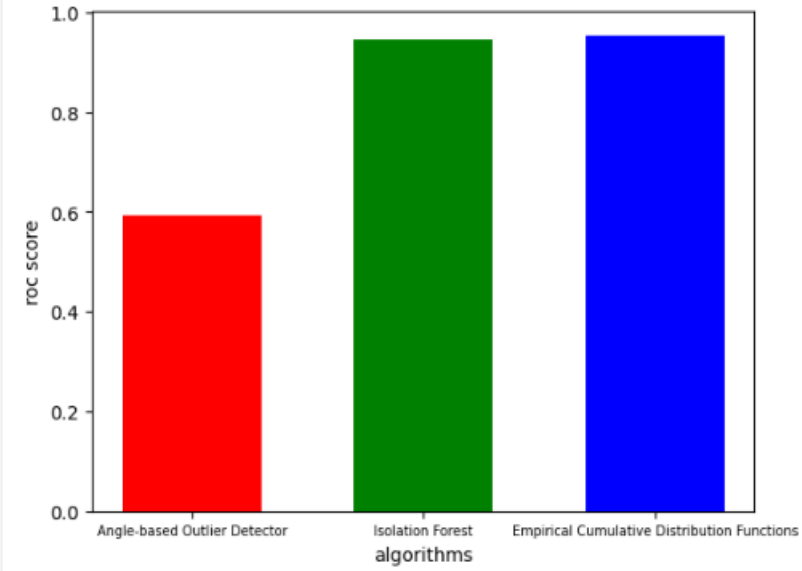




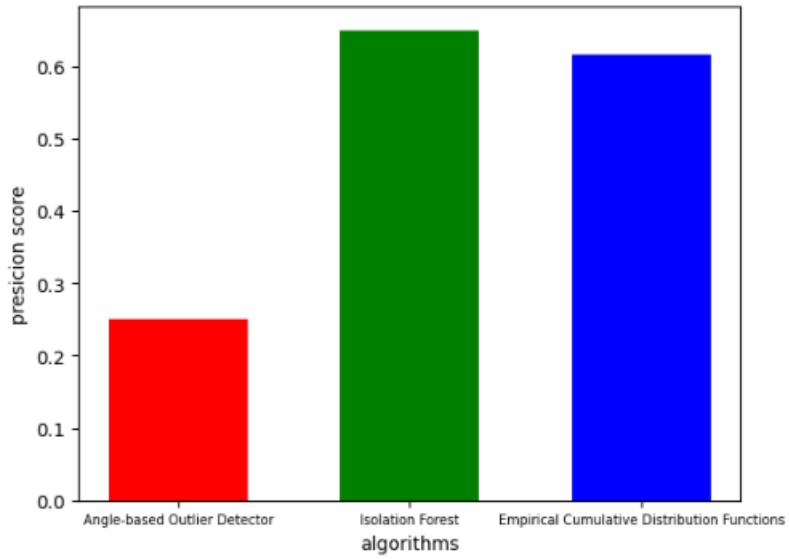
(αποτελέσματα precision (πάνω), time_complexity(κάτω) συνόλου δεδομένων arrhythmia)

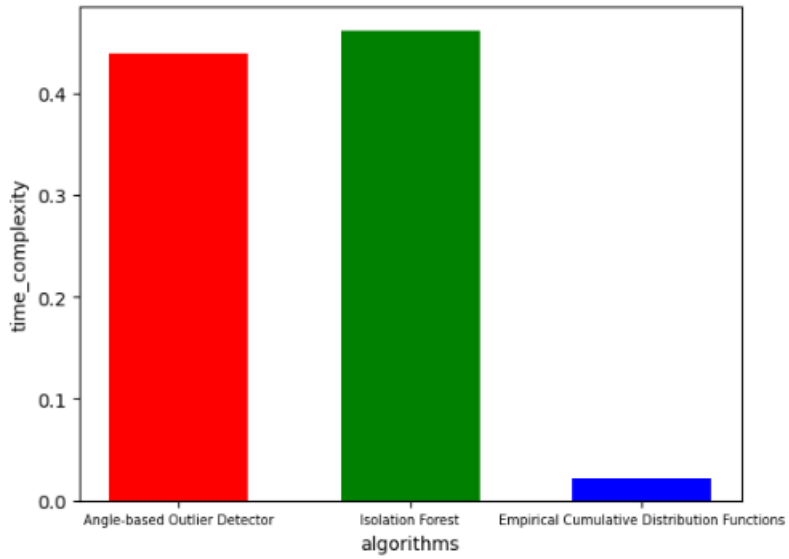
Το δεύτερο σύνολο δεδομένων που εξετάζουμε έχει 1831 στοιχεία και 21 διαστάσεις, και ποσοστό ακραίων τιμών 9.6% (176 ακραίες τιμές).
Στο συγκεκριμένο σύνολο καλύτερη επίδοση είχε ο ECDF λόγω του roc score 0,9543, precision 0,6167 time complexity 0,022 seconds.
Δεύτερη καλύτερη επίδοση ο IFOREST λόγω roc score 0,9467, precision 0,65, time complexity 0,4614 seconds.
Χειρότερη επίδοση ο ABOD λόγω του roc score 0,6819, precision 0,3, time complexity 0,1715 seconds.

```
...εξέταση του σύνολου δεδομένων  cardio.mat ...
kalw algorithmo gia high_dimentional
Angle-based Outlier Detector ROC:0.592, precision @ rank n:0.25, execution time: 0.4388s
Isolation Forest ROC:0.9467, precision @ rank n:0.65, execution time: 0.4614s
Empirical Cumulative Distribution Functions ROC:0.9543, precision @ rank n:0.6167, execution time: 0.022s
```



(αποτελέσματα roc_score συνόλου δεδομένων cardio)





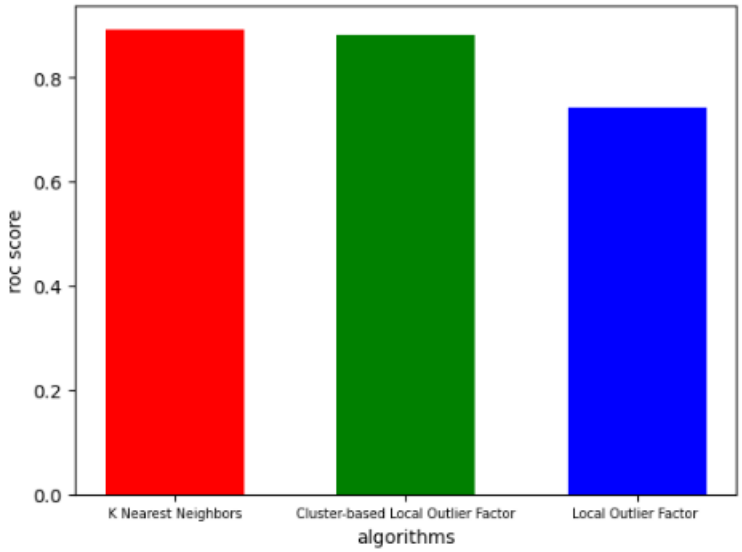
(αποτελέσματα precision (πάνω), time_complexity(κάτω) συνόλου δεδομένων cardio)

Το τρίτο σύνολο δεδομένων που εξετάζουμε έχει 214 στοιχεία και 9 διαστάσεις, και ποσοστό ακραίων τιμών 4.2% (9 ακραίες τιμές).

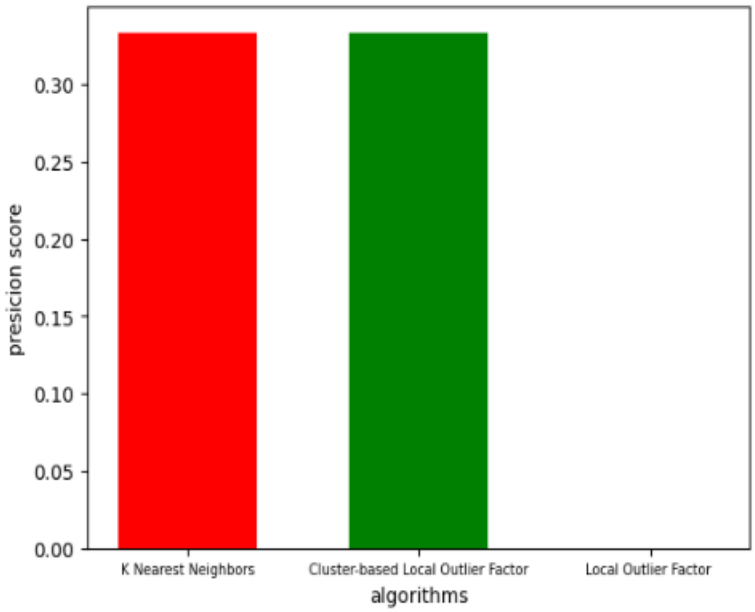
Στο συγκεκριμένο σύνολο καλύτερη επίδοση είχε ο KNN λόγω του roc score 0,89, precision 0,33, time complexity 0,0032 seconds.
Δεύτερη καλύτερη επίδοση ο CBLOF λόγω roc score 0,8817, precision 0,33, time complexity 2,38 seconds.
Χειρότερη επίδοση ο LOF λόγω του roc score 0,74, precision 0, time complexity 0,0059 seconds.(Το precision 0 προκύπτει λόγω του μικρού ποσοστού outlier στο σύνολο 4.2% και μικρού αριθμού στοιχείων στο σύνολο 214)

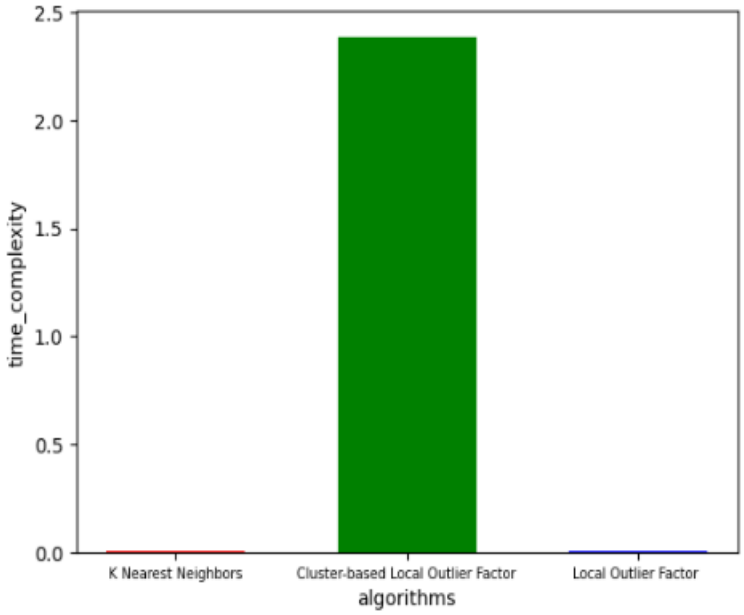
```
...εξέταση του σύνολου δεδομένων  glass.mat ...
kalw algorithmo gia  non_parametric

K Nearest Neighbors ROC:0.8925, precision @ rank n:0.3333, execution time: 0.0032s
Cluster-based Local Outlier Factor ROC:0.8817, precision @ rank n:0.3333, execution time: 2.3873s
Local Outlier Factor  ROC:0.7419, precision @ rank n:0.0, execution time: 0.0059s
```



(αποτελέσματα roc_score συνόλου δεδομένων glass)

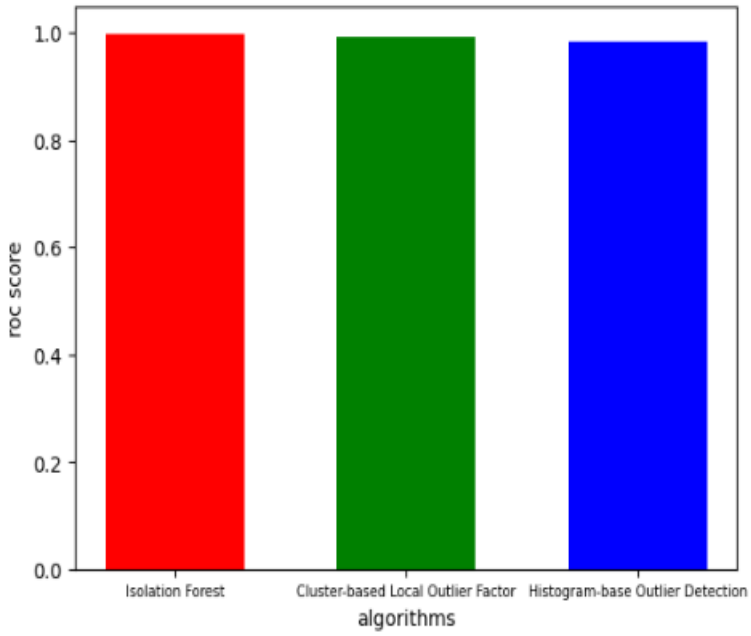




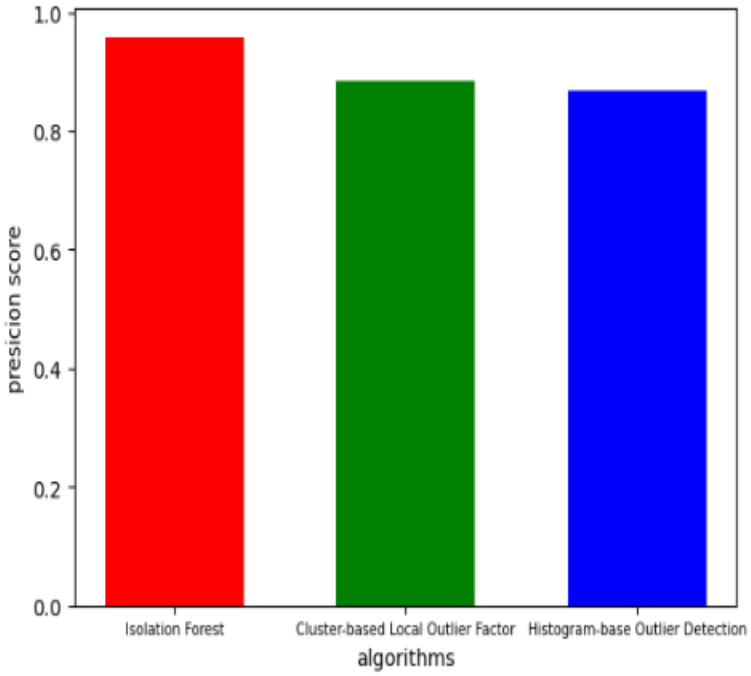
(αποτελέσματα precision (πάνω), time_complexity(κάτω) συνόλου δεδομένων glass)

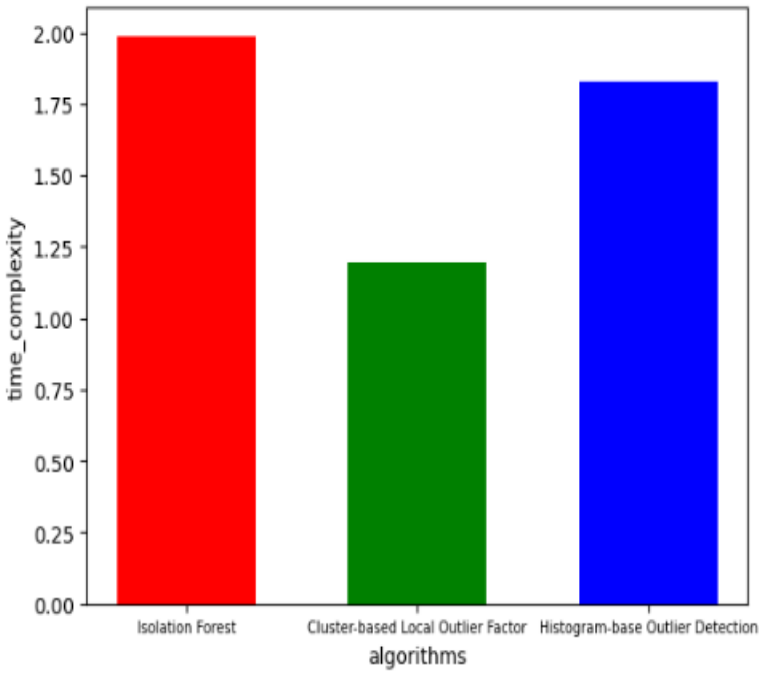
Το τέταρτο σύνολο δεδομένων που εξετάζουμε έχει 49097 στοιχεία και 9 διαστάσεις, και ποσοστό ακραίων τιμών 7% (3511 ακραίες τιμές).
Στο συγκεκριμένο σύνολο καλύτερη επίδοση είχε ο IFOREST λόγω του roc score 0,9981, precision 0,9586, time complexity 1,9875 seconds.
Δεύτερη καλύτερη επίδοση ο CBLOF λόγω roc score 0,9929, precision 0,, time complexity 1,1935 seconds.
Χειρότερη επίδοση ο HBOS λόγω του roc score 0,983, precision 0,8693, time complexity 1,8259 seconds.

```
...εξέταση του σύνολου δεδομένων  shuttle.mat ...
kalw algorithmo gia  complex_non_high_dimestional
Isolation Forest ROC:0.9981, precision @ rank n:0.9586, execution time: 1.9875s
Cluster-based Local Outlier Factor ROC:0.9929, precision @ rank n:0.8853, execution time: 1.1935s
Histogram-base Outlier Detection ROC:0.983, precision @ rank n:0.8693, execution time: 1.8259s
```



(αποτελέσματα roc_score συνόλου δεδομένων shuttle)



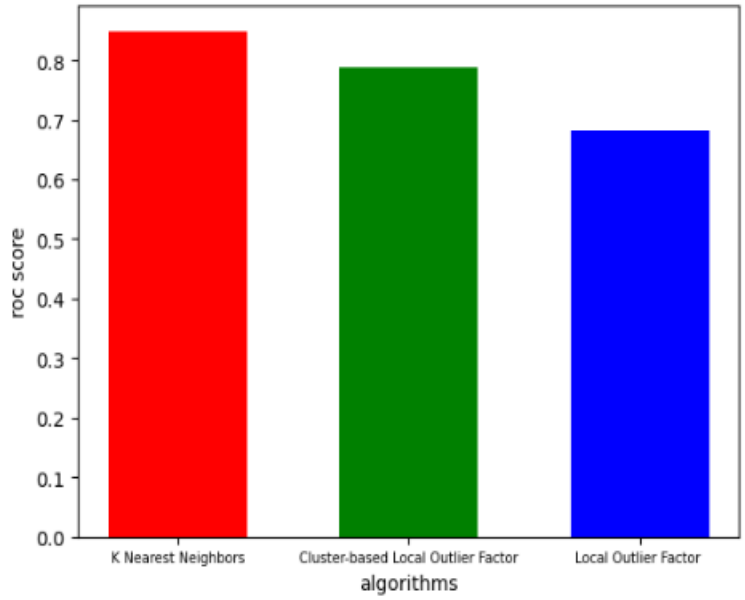


(αποτελέσματα precision (πάνω), time_complexity(κάτω) συνόλου δεδομένων shuttle)

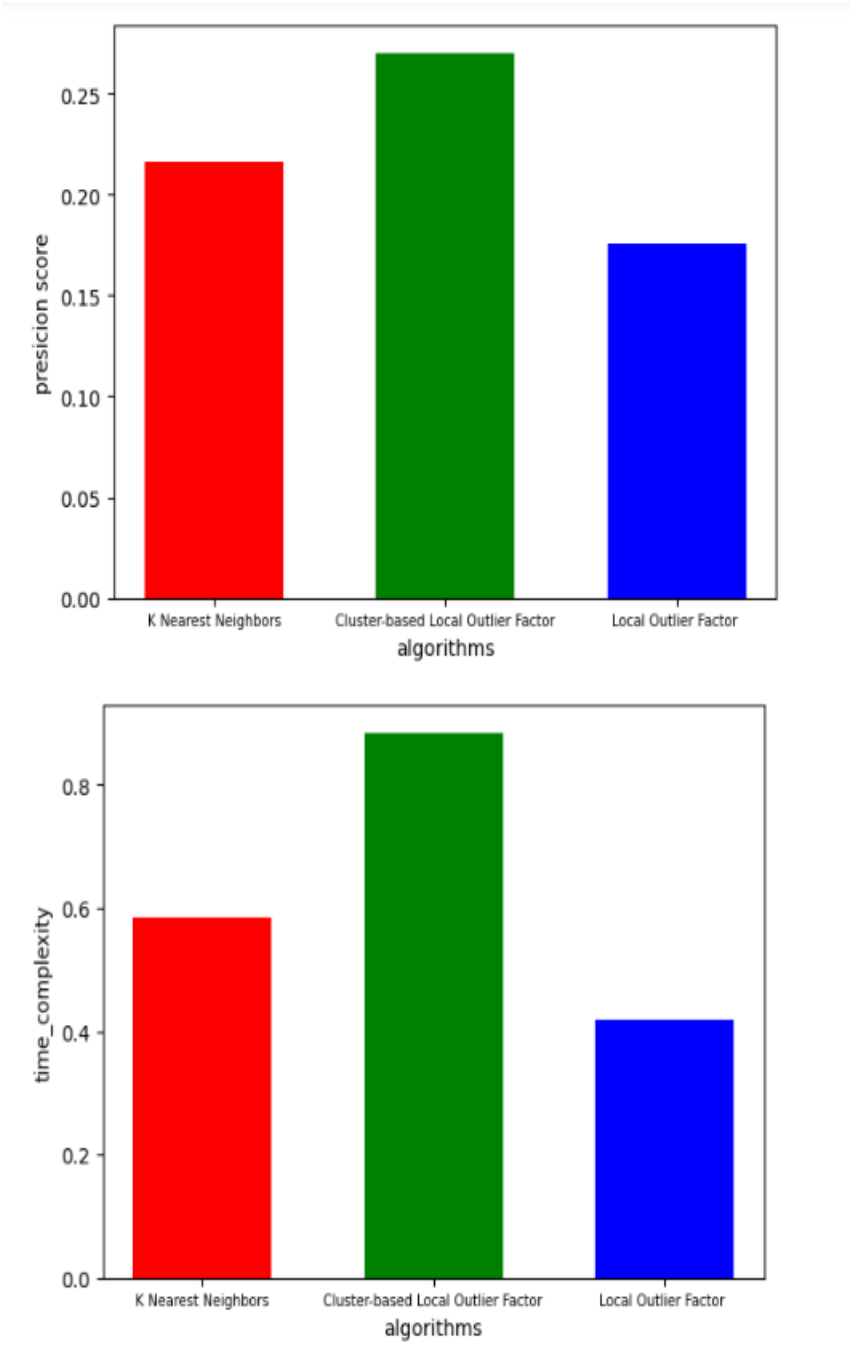
Το πέμπτο σύνολο δεδομένων που εξετάζουμε έχει 11183 στοιχεία και 6 διαστάσεις, και ποσοστό ακραίων τιμών 2.32% (260 ακραίες τιμές).
Στο συγκεκριμένο σύνολο καλύτερη επίδοση είχε ο KNN λόγω του roc score 0,8493, precision 0,2162, time complexity 0,5857 seconds.
Δεύτερη καλύτερη επίδοση ο CBLOF λόγω roc score 0,788, precision 0,2703, time complexity 0,8843 seconds.
Χειρότερη επίδοση ο LOF λόγω του roc score 0,6821, precision 0,1757, time complexity 0,4184 seconds.

```
...εξέταση του σύνολου δεδομένων  mammography.mat ...
kalw algorithmo gia  non_parametric

K Nearest Neighbors ROC:0.8493, precision @ rank n:0.2162, execution time: 0.5857s
Cluster-based Local Outlier Factor ROC:0.788, precision @ rank n:0.2703, execution time: 0.8843s
Local Outlier Factor  ROC:0.6821, precision @ rank n:0.1757, execution time: 0.4184s
```



(αποτελέσματα roc_score συνόλου δεδομένων mammography)

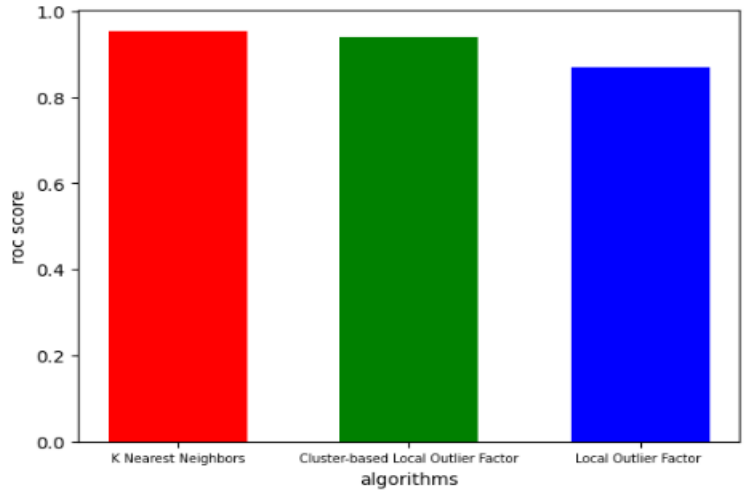


(αποτελέσματα precision (πάνω), time_complexity(κάτω) συνόλου δεδομένων mammography)

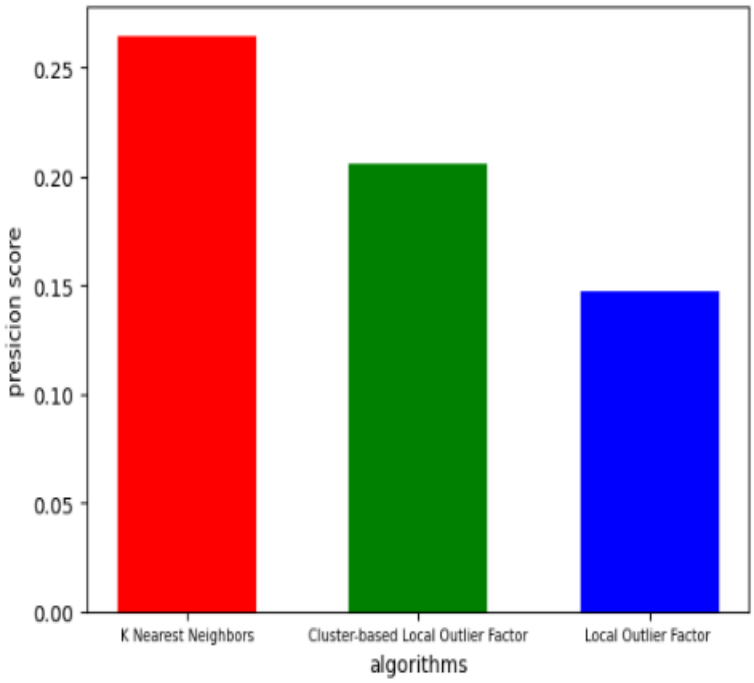
Το έκτο σύνολο δεδομένων που εξετάζουμε έχει 3772 στοιχεία και 6 διαστάσεις, και ποσοστό ακραίων τιμών 2.5% (93 ακραίες τιμές).
Στο συγκεκριμένο σύνολο καλύτερη επίδοση είχε ο KNN λόγω του roc score 0,9541, precision 0,2647, time complexity 0,1362 seconds.
Δεύτερη καλύτερη επίδοση ο CBLOF λόγω roc score 0,9402, precision 0,2059, time complexity 0,964 seconds.
Χειρότερη επίδοση ο LOF λόγω του roc score 0,869, precision 0,1471, time complexity 0,0974 seconds.

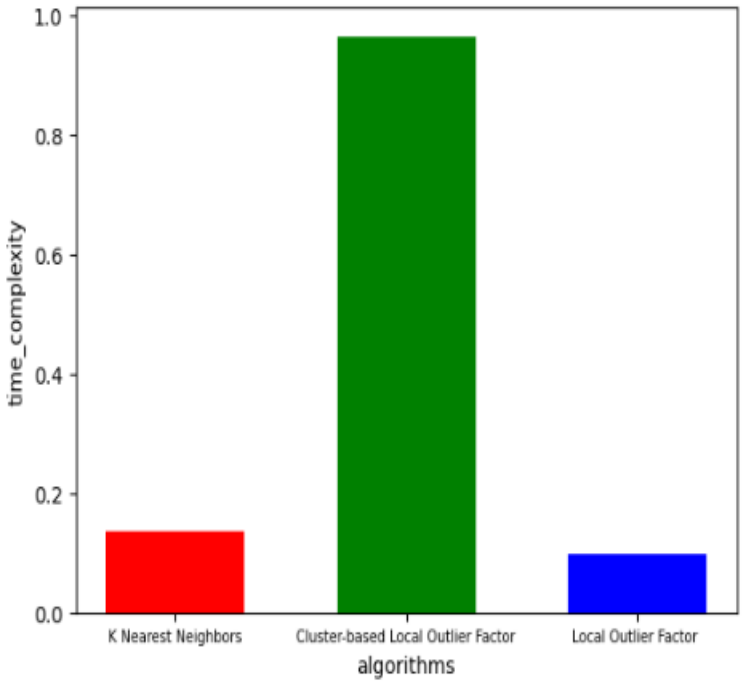
```
...εξέταση του σύνολου δεδομένων  thyroid.mat ...
kalw algorithmo gia  non_parametric

K Nearest Neighbors ROC:0.9541, precision @ rank n:0.2647, execution time: 0.1362s
Cluster-based Local Outlier Factor ROC:0.9402, precision @ rank n:0.2059, execution time: 0.9646s
Local Outlier Factor  ROC:0.869, precision @ rank n:0.1471, execution time: 0.0974s
```



(αποτελέσματα roc_score συνόλου δεδομένων thyroid)



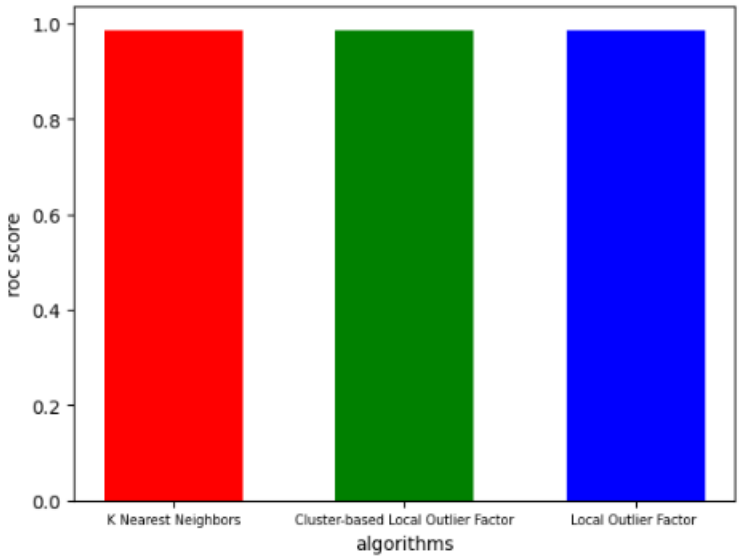


(αποτελέσματα precision (πάνω), time_complexity(κάτω) συνόλου δεδομένων thyroid)

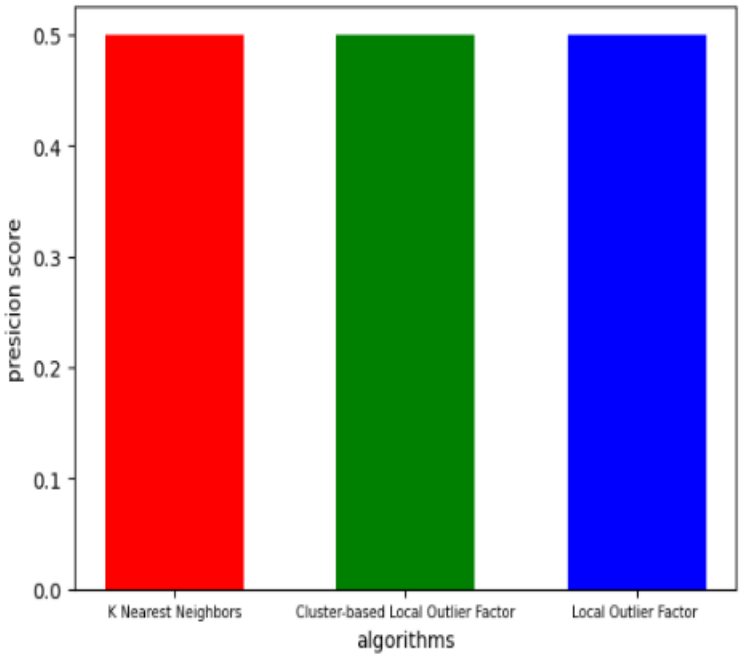
Το έβδομο σύνολο δεδομένων που εξετάζουμε έχει 129 στοιχεία και 13 διαστάσεις, και ποσοστό ακραίων τιμών 7,7% (10 ακραίες τιμές).
Στο συγκεκριμένο σύνολο καλύτερη επίδοση είχε ο LOF λόγω του roc score 0,9865, precision 0,5, time complexity 0,005 seconds.
Δεύτερη καλύτερη επίδοση ο KNN λόγω roc score 0,9865, precision 0,5, time complexity 0,0078 seconds.
Χειρότερη επίδοση ο CBLOF λόγω του roc score 0,9865, precision 0,5, time complexity 0,005 seconds.

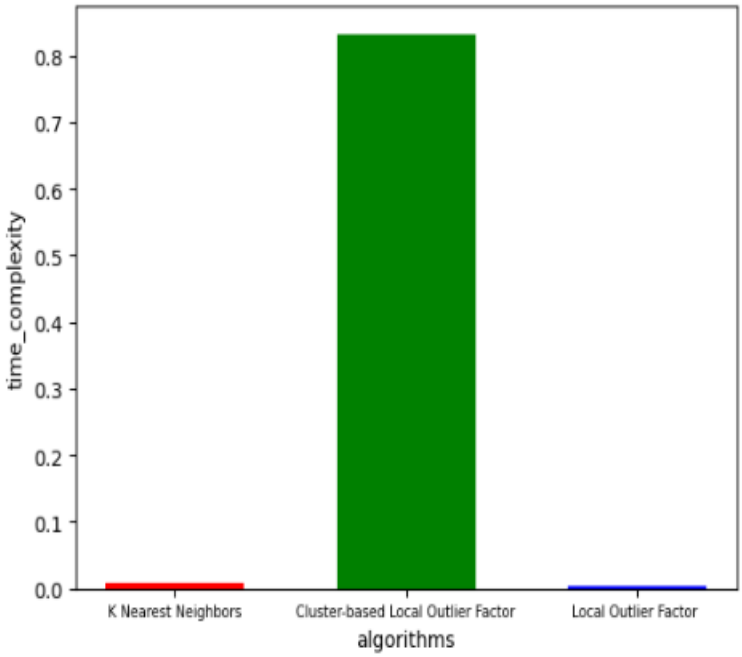
```
...εξέταση του σύνολου δεδομένων  wine.mat ...
kalw algorithmo gia  non_parametric

K Nearest Neighbors ROC:0.9865, precision @ rank n:0.5, execution time: 0.0078s
Cluster-based Local Outlier Factor ROC:0.9865, precision @ rank n:0.5, execution time: 0.8331s
Local Outlier Factor  ROC:0.9865, precision @ rank n:0.5, execution time: 0.005s
```



(αποτελέσματα roc_score συνόλου δεδομένων wine)

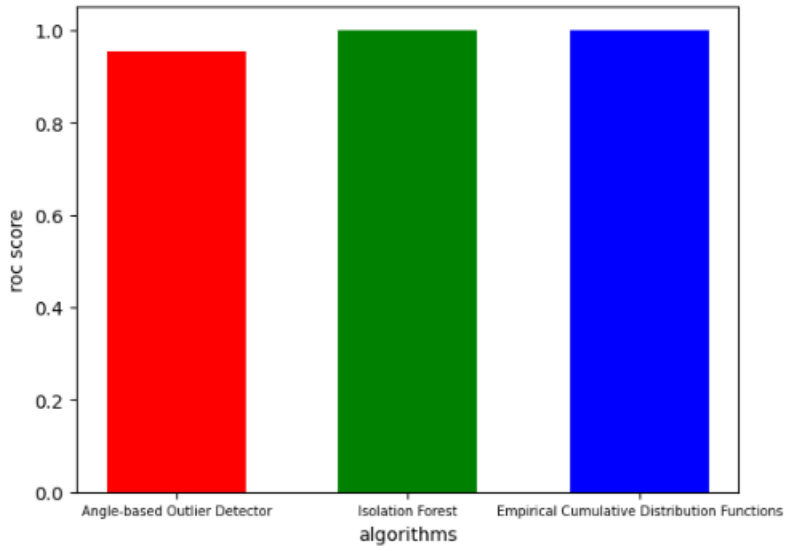




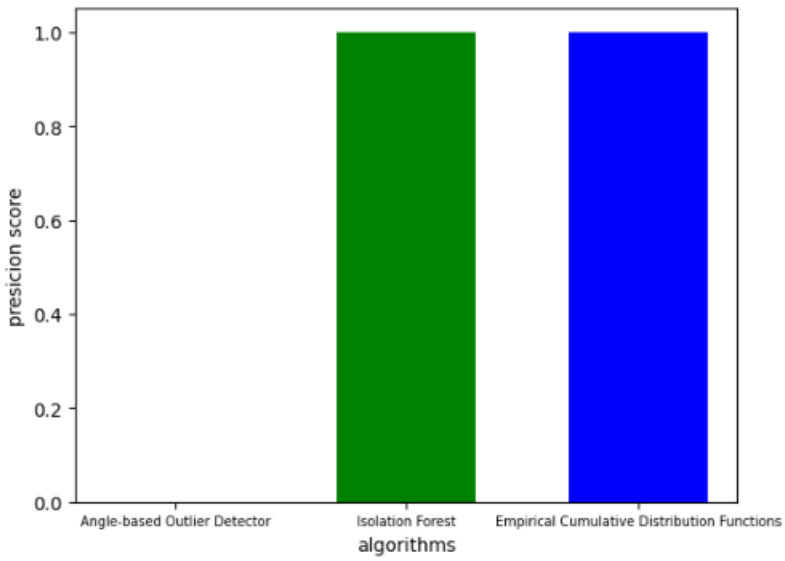
(αποτελέσματα precision (πάνω), time_complexity(κάτω) συνόλου δεδομένων wine)

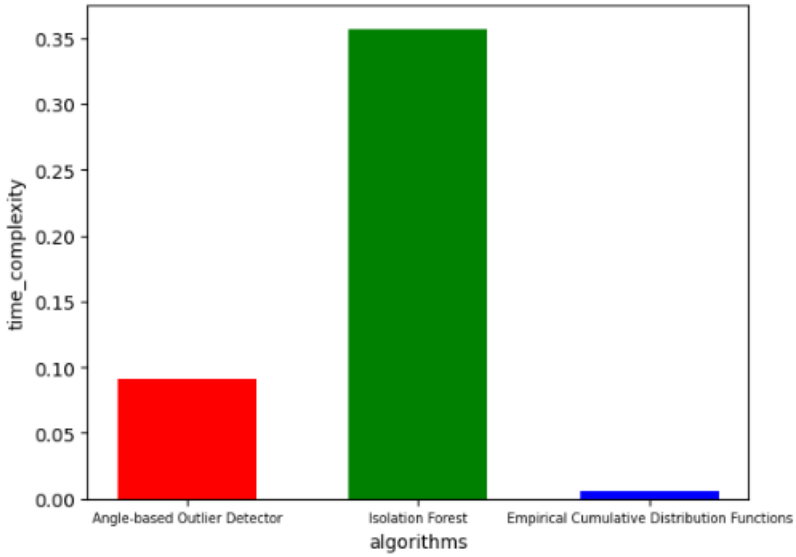
Το τελευταίο σύνολο δεδομένων που εξετάζουμε έχει 148 στοιχεία και 18 διαστάσεις, και ποσοστό ακραίων τιμών 4,1% (6 ακραίες τιμές).
Στο συγκεκριμένο σύνολο καλύτερη επίδοση είχε ο ECDF λόγω του roc score 1, precision 1, time complexity 0,0063 seconds.
Δεύτερη καλύτερη επίδοση ο IFOREST λόγω roc score 1, precision 1, time complexity 0,357 seconds.
Χειρότερη επίδοση ο ABOD λόγω του roc score 0,9545, precision 0, time complexity 0,0914 seconds (Το precision 0 προκύπτει λόγο του μικρού ποσοστού outlier στο σύνολο 4.1% και μικρού αριθμού στοιχείων στο σύνολο 148).

```
...εξέταση του σύνολου δεδομένων  lympho.mat ...
kalw algorithmo gia high_dimentional
Angle-based Outlier Detector ROC:0.9545, precision @ rank n:0.0, execution time: 0.0914s
Isolation Forest ROC:1.0, precision @ rank n:1.0, execution time: 0.3571s
Empirical Cumulative Distribution Functions ROC:1.0, precision @ rank n:1.0, execution time: 0.0063s
```



(αποτελέσματα roc_score συνόλου δεδομένων lympho)





(αποτελέσματα precision (πάνω), time_complexity(κάτω) συνόλου δεδομένων lympho)

ΣΧΟΛΙΑΣΜΟΣ ΑΠΟΤΕΛΕΣΜΑΤΩΝ

Τα αποτελέσματα της συνάρτησης ήταν αρκετά ενθαρρυντικά αν και το δείγμα των συνόλων δεδομένων δεν ήταν αρκετό (8 σύνολα δεδομένων) ώστε να βγουν ασφαλείς συμπεράσματα. Τα πειράματα έγιναν σε πραγματικά δεδομένα με προσθήκη τεχνητών ακραίων τιμών. Λόγο της φύσης του προβλήματος ήταν δύσκολο να βρεθούν "labeled" δεδομένα ώστε να μπορεί να γίνει αξιολόγηση της επίδοσης των μοντέλων και περαιτέρω ανάλυση των αποτελεσμάτων. Καταφέραμε να καλύψουμε τις περιπτώσεις των απλών συνόλων δεδομένων που δεν ακολουθούν κανονική κατανομή, των σύνθετων δεδομένων πολλών διαστάσεων (high dimensional) και των σύνθετων δεδομένων κανονικών διαστάσεων, δεν βρέθηκαν σύνολα δεδομένων που να ακολουθούν κανονική κατανομή και να έχουν ακραίες τιμές ή χρονοσειρές που να έχουν "labeled" δεδομένα ώστε να μπορεί να γίνει αξιολόγηση της επίδοσης των μοντέλων. Σε κάθε σύνολο υπήρχε τουλάχιστον ένας αλγόριθμος (στις περισσότερες περιπτώσεις περισσότεροι από έναν) που το roc score να ήταν μεγαλύτερο από το όριο που είχαμε θέσει ώστε να ήταν αποδεκτό και δεν χρειάστηκε να καλεστούν οι ταξινομητές ψηφοφορίας (voting classifiers) ώστε να έχουμε καλύτερες επιδόσεις στο τελικό πρόγραμμα. Λάβαμε υπόψιν και το precision και το time_complexity που ήταν και τα δύο σε αποδεκτά όρια, ειδικά η χρονική πολυπλοκότητα των αλγορίθμων ήταν αρκετά καλή. Σε δύο σύνολα δεδομένων παρατηρήσαμε ότι το precision έλαβε την τιμή 0 ενώ το roc score ήταν σε επιθυμητά πλαίσια. Στα συγκεκριμένα σύνολα δεδομένων υπήρχε πολύ μικρό ποσοστό ακραίων τιμών (περίπου 4%) και αρκετό μικρό αριθμό δεδομένων, δοθέντος και του τύπου υπολογισμού του δείκτη precision (TP/(TP+NP)), κάνοντας λογικό ένα τέτοιο αποτέλεσμα.

ΣΥΜΠΕΡΑΣΜΑ

Συνοψίζοντας, προσπαθήσαμε να καλύψουμε ένα πεδίο της ανίχνευσης ανωμαλιών που δεν είχε αναλυθεί αρκετά και λεπτομερώς σχετικά με άλλα πεδία εφαρμογής. Δίνοντας περισσότερη βάση στην ανάλυση των χαρακτηριστικών των δεδομένων, στις μεθοδολογίες των αλγορίθμων και στα χαρακτηριστικά των αλγορίθμων προτείνοντας τους βέλτιστους βάση των κριτηρίων που θεωρήσαμε σημαντικά.

Μέσω της μελέτης και ανάλυσης των μεθοδολογιών καταφέραμε να έχουμε μεγαλύτερη κατανόηση στον ορισμό των ακραίων τιμών. Στις μεθόδους αντιμετώπισής τους. Εμβαθύνοντας στην ανάλυση πολλαπλών αλγορίθμων δίνοντας έτσι ευελιξία στο πρόγραμμα μας έχοντας την δυνατότητα να χειριστεί διαφόρων ειδών στατικά δεδομένα. Είτε σύνθετα είτε απλά αλλά και χωρίς να επικεντρωθούμε αρκετά στα χρονικά μεταβαλλόμενα δεδομένα παρουσιάζοντας κάποιους ενδεδειγμένους αλγορίθμους και κάποιες εξελιγμένες προσεγγίσεις βαθιάς μάθησης όπως το νευρωνικό δίκτυο LSTM autoencoder να καλύψουμε εν μέρη και αυτό τον τομέα.

Η ανίχνευση ακραίων τιμών είναι ένα πολύ απαιτητικό σύνθετο πρόβλημα. Χρειάζεται ευρεία γνώση μαθηματικών, στατιστικής και πληροφορικής για την κατανόηση των δεδομένων, των αλγορίθμων και των μεθοδολογιών. Ένα συνεχώς μεταβαλλόμενο πεδίο με εφαρμογές σε πολλαπλούς τομείς που πρέπει να παρθούν αποφάσεις βάση των δεδομένων. Με την αύξηση του όγκου, της πολυπλοκότητας των δεδομένων αλλά και των τρόπων συλλογής τους κάνει αναγκαία την περαιτέρω έρευνα και ανάπτυξη νέων τεχνολογιών πάνω στο πεδίο.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Άρθρα:
1. A Survey of Outlier Detection Methodologies - Victoria J. Hodge, Jim Austin (https://eprints.whiterose.ac.uk/767/1/)
2. A critical overview of outlier detection methods - Abir Smiti (https://www.sciencedirect.com/science/article/abs/pii/S1574013720304068)
3. What is an Outlier? Definition and How to Find Outliers in Statistics - Dionysia Lemonaki (https://www.freecodecamp.org/news/what-is-an-outlier-definition-and-how-to-find-outliers-in-statistics)
4. Noise Versus Outliers – Cátia M. Salgado, Carlos Azevedo, Hugo Proença & Susana M. Vieira (https://link.springer.com/chapter/10.1007/978-3-319-43742-2_14)
5. Types of Outliers in Data Mining - geeksforgeeks (https://www.geeksforgeeks.org/types-of-outliers-in-data-mining/)
6. Outliers explained: a quick guide to the different types of outliers - Ira Cohen (https://towardsdatascience.com/outliers-analysis-a-quick-guide-to-the-different-types-of-outliers-e41de37e6bf6)
7. A Five Step Procedure for Outlier Analysis in Data Mining - V Ilango, R. Subramanian, V. Vasudevan (https://www.researchgate.net/publication/267247892_A_Five_Step_Procedure_for_Outlier_Analysis_in_Data_Mining)
8. Outlier Detection: A Novel Depth Approach - Yixin Chen, Xin Dang, Hanxiang Peng (https://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=0B6ECB3BDC518C5E0608755E3E4ED523?doi=10.1.1.585.9616&rep=rep1&type=pdf)
9. ADBench: Anomaly Detection Benchmark – Songqiao Han, Xiyang Hu, Hailiang Huang, Minqi Jiang, Yue Zhao (https://arxiv.org/pdf/2206.09426.pdf)
10. Graph-Based Anomaly Detection - Caleb C. Noble, Diane J. Cook (https://eecs.wsu.edu/~cook/pubs/kdd03.pdf)
11. LOF: Identifying Density-Based Local Outliers - Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, Jörg Sander (https://dl.acm.org/doi/pdf/10.1145/342009.335388)
12. Nearest-Neighbor and Clustering based Anomaly Detection Algorithms for RapidMiner - Mennatallah Amer and Markus Goldstein (https://www.goldiges.de/publications/Anomaly_Detection_Algorithms_for_RapidMiner.pdf)
13. k nearest neighbor outlier detection – The AI Blog (https://aiblog.co.za/technology/k-nearest-neighbor-outlier-detection-2)
14. Isolation Forest - Fei Tony Liu, Kai Ming Ting, Zhi-Hua Zhou (https://cs.nju.edu.cn/zhouzh/zhouzh.files/publication/icdm08b.pdf?q=isolation-forest)
15. ONE-CLASS SUPPORT VECTOR MACHINES APPROACH TO ANOMALY DETECTION - Maryamsadat Hejazi & Yashwant Prasad Singh (https://www.tandfonline.com/doi/pdf/10.1080/08839514.2013.785791)
16. ECOD: Unsupervised Outlier Detection Using Empirical Cumulative Distribution Functions - Zheng Li, Yue Zhao, Student Member Xiyang Hu, Nicola Botta, Cezar Ionescu and George H. Chen (https://arxiv.org/pdf/2201.00382.pdf)
17. Wikipedia (https://el.wikipedia.org/wiki/%CE%9A%CE%B1%CE%BD%CE%BF%CE%BD%CE%B9%CE%BA%CE%AE_%CE%BA%CE%B1%CF%84%CE%B1%CE%BD%CE%BF%CE%BC%CE%AE#/media/%CE%91%CF%81%CF%87%CE%B5%CE%AF%CE%BF:Standard_deviation_diagram.svg)

18. Histogram-based Outlier Score (HBOS): A fast Unsupervised Anomaly Detection Algorithm - Markus Goldstein and Andreas Dengel (https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.401.5686&rep=rep1&type=pdf)
19. A Near-linear Time Approximation Algorithm for Angle-based Outlier Detection in High-dimensional Data - Ninh Pham, Rasmus Pagh (https://pure.itu.dk/ws/files/38198304/rt613_pham.pdf)
20. K-Nearest Neighbour: The Distance-Based Machine Learning Algorithm - Shivam Sharma (https://www.analyticsvidhya.com/blog/2021/05/knn-the-distance-based-machine-learning-algorithm/)
22. The Relationship Between Precision-Recall and ROC Curves - Jesse Davis, Mark Goadrich - (https://www.biostat.wisc.edu/~page/rocpr.pdf)
23. Revisiting Time Series Outlier Detection: Definitions and Benchmarks – Kwei-Herng Lai, Daochen Zha, Junjie Xu, Yue Zhao, Guanchu Wang, Xia Hu (https://openreview.net/pdf?id=r8IvOsnHchr)
24. LSTM-Autoencoder based Anomaly Detection for Indoor Air Quality Time Series Data - Yuanyuan Wei, Julian Jang-Jaccard, Wen Xu, Fariza Sabrina, Seyit Camtepe, Mikael Boulic(https://arxiv.org/pdf/2204.06701.pdf)

Σύνολα δεδομένων :
21. Shebuti Rayana (2016). ODDS Library [https://odds.cs.stonybrook.edu]. Stony Brook, NY: Stony Brook University, Department of Computer Science.